Comprehensive Quality-Aware Automated Semantic Web Service Composition

by

Chen Wang

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Software Engineering.

Victoria University of Wellington
2020

Abstract

Automated web service composition is one of the ultimate goals of service-oriented computing. It loosely couples web services to accommodate users' complex requirements. Evolutionary Computation (EC) techniques combined with AI planning have been successfully proposed to efficiently produce composite services with near-optimal Quality of Semantic Matchmaking (QoSM) and/or Quality of Service (QoS), which measure the satisfaction of the functional and non-functional requirements from users, respectively. Despite some recent progress, both the effectiveness and efficiency of existing approaches need further improvement to enhance the competitive advantage of service providers. The overall goal of this thesis is to propose novel EC-based fully automated service composition approaches that can effectively and efficiently solve challenging single-objective, multi-objective, and dynamic service composition problems.

Firstly, this thesis proposes two novel Estimation of Distribution Algorithm (EDA) based approaches (called EDA-NHM and EDA-EHM) and one memetic EDA-based approach with four different local search operators to single-objective fully automated web service composition that jointly optimizes QoSM and QoS. EDA-NHM and EDA-EHM are proposed with novel permutation-based and DAG-based representations to model the distribution of composition solutions with respect to varied service composition workflows. Two sampling techniques are also studied in EDA-NHM and EDA-EHM to effectively and efficiently sample new promising permutations and functionally valid DAGs, respectively. These two EDA-based approaches are compared to state-of-the-art works. The comparisons reveal that EDA-NHM produces better-quality composite services than EDA-EHM and the state-of-the-art works. On the other

hand, EDA-EHM achieves the highest efficiency among all the competing EC-based methods, delivering moderate effectiveness. Furthermore, one proposed memetic approaches built upon EDA-NHM (called MEEDA-LOP) pushes the cutting-edge performance in terms of effectiveness and efficiency.

Secondly, this thesis studies two categories of multi-objective service composition problems: one category aims to generate a set of approximated Pareto optimal solutions for users to choose from, while the other category aims to generate multiple composite services for multiple user segments with distinctive preferences on QoSM. To effectively and efficiently handle the first category of problems, a memetic approach based on Non-dominated Sorting Genetic Algorithm II (NSGA-II), called MNSGA2-EDA, is proposed by enhancing NSGA-II with EDA-based local search. The novelty of this method lies in the innovative use of EDA for effective and efficient local improvements, rather than for global exploration. MNSGA2-EDA is compared to state-of-the-art multi-objective works, for studying its performance. We found that MNSGA2-EDA achieves much higher effectiveness and efficiency in finding Pareto optimal solutions. The second category of problems can be naturally treated as multitasking problems. Two novel multi-factorial evolutionary algorithms (called PMFEA and PMFEA-EDA) are proposed to effectively and efficiently solve this category of problems. These two algorithms implicitly or explicitly learn and share the knowledge of good solutions evolved so far for different tasks. We compare PMFEA and PMFEA-EDA with state-of-the-art works. We found that both PMFEA-EDA and PMFEA are performed at the cost of only a fraction of time compared to the single-tasking state-of-the-art works, which solve one task at a time. We also found that PMFEA-EDA yields solutions with the highest quality, confirming that learning and sharing knowledge explicitly is superior to learning and sharing knowledge implicitly.

Thirdly, this thesis studies a new dynamic service composition prob-

lem, focusing on handling stochastic service failures. We effectively handle this problem via two stages — the design stage and the execution stage. Particularly, two accurate robustness measures are proposed based on Monte Carlo sampling and a lower bound estimation, respectively. These robustness measures are utilized in two proposed GA-based approaches (called GA-MC and GA-RE) at the design stage, to generate baseline composite solutions with high robustness. These baseline solutions can cope with the stochastic service failures robustly via a repairing process that supports continued high-quality execution of a composite service at the execution stage. Meanwhile, we propose a GA-2Stage algorithm by introducing a new adaptive evolutionary control mechanism, which supports two sequential evolutionary stages with two different fitness evaluation methods. These approaches are compared to each other to determine the most suitable method. Our experimental comparisons reveal that GA-RE algorithm with lower bound estimation outperforms GA-MC algorithm with Monte Carlo sampling estimation in finding composition solutions with high robustness, regardless of the size of the service repositories. Besides, compared to GA-RE, GA-2Stage achieves the highest efficiency with a negligible impact on the effectiveness at the execution stage, regardless of the service repositories' size.

List of Publications

1. WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Comprehensive quality-aware automated semantic web service composition. In *AI 2017: Advances in Artificial Intelligence* (2017), Springer, pp. 195–207

2. WANG, C., MA, H., CHEN, G., AND HARTMANN, S. GP-based approach to comprehensive quality-aware automated semantic web service composition. In *Simulated Evolution and Learning* (2017), Springer, pp. 170–183

3. WANG, C., MA, H., AND CHEN, G. EDA-based approach to comprehensive quality-aware automated semantic web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2018), GECCO '18, ACM, pp. 147–148

4. WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Knowledge-driven automated web service composition — an EDA-based approach. In *Web Information Systems Engineering – WISE 2018* (2018), Springer, pp. 135–150

5. WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Towards fully automated semantic web service composition based on estimation of distribution algorithm. In *AI 2018: Advances in Artificial Intelligence* (2018), Springer, pp. 458–471

6. WANG, C., MA, H., CHEN, G., AND HARTMANN, S. A memetic NSGA-II with EDA-based local search for fully automated multi-

objective web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2019), GECCO '19, ACM, pp. 421–422

7. WANG, C., MA, H., AND CHEN, G. Using EDA-based local search to improve the performance of NSGA-II for multiobjective semantic web service composition. In *Database and Expert Systems Applications* (2019), Springer, pp. 434–451

8. WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Evolutionary multitasking for semantic web service composition. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), pp. 2490–2497

9. WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Towards robust web service composition with stochastic service failures based on a genetic algorithm. In *AI 2019: Advances in Artificial Intelligence* (2019), Springer, pp. 445–459

10. WANG, C., MA, H., CHEN, A., HARTMANN, S., AND ONG, Y.-S. Using an Estimation of Distribution Algorithm to achieve multitasking semantic web service composition. *ACM Transactions on Evolutionary Learning and Optimization* (Major revision)

11. WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Memetic EDA-based approaches to comprehensive quality-aware automated semantic web service composition. *IEEE Transactions on Services Computing* (Submitted)

12. WANG, C., MA, H., CHEN, A., HARTMANN, S., AND BRANKE, J. Robustness estimation and optimisation for semantic web service composition with stochastic service failures. *IEEE Transactions on Emerging Topics in Computational Intelligence* (Major revision)

Acknowledgments

I would like to express my deepest gratitude to my supervisors, A/Prof. Hui Ma, Dr. Gang Chen and Prof. Sven Hartmann (external supervisor) for their invaluable assistance and encouragement throughout my studies. My supervisors convincingly guided and encouraged me to be professional when the road got tough. Without their persistent help, the goal of my studies would not have been realized. I am thankful to other participants of my research works: Prof. Ong, Yew-Soon, and Prof. Jürgen Branke. I would also like to pay my special regards for the nurturing and supportive environment at Victoria University of Wellington, the Evolutionary Computation Research Group (ECRG) led by Prof. Mengjie Zhang and the Evolutionary Computation for Combinatorial Optimization (ECCO) subgroup.

I am deeply grateful to my family, in particular, my mother, Chunyun and my father, Jianxiang, for giving me the opportunity to pursue this degree at Victoria University of Wellington. I would also like to acknowledge the support and great love of my girlfriend, Wendy. They kept me going on, and this work would not have been possible without their encouragement. Last but not least, thank you to all my friends and colleagues, including those not explicitly listed, who assisted me in this journey: Dr. Alexandre Sawczuk da Silva, Atiya Masood, Boxiong Tan, Dr, Harith Al-Sahaf, Dr. Julian Mackay, Dr. John Park, Soheila Sadeghiram, Dr. Lily Chanida, Dr. Yi Mei, Dr, Yiming Pei and Dr. Ying Qu.

Contents

1 Introduction **1**

1.1 Problem Statement 1
1.2 Motivations 7
1.3 Research Goals 13
1.4 Major Contributions 19
1.5 Organization of Thesis 25

2 Background and Literature Review **27**

2.1 Background 27
2.1.1 Web Service 27
2.1.2 Web Service Composition 30
2.1.3 An Overview of EC Techniques 41
2.1.4 An Overview of AI Planning 53
2.2 Literature Review 56
2.2.1 Semi-Automated Web Service Composition 58
2.2.2 Fully Automated Web Service Composition 68
2.3 Summary 76

3 Single-Objective Fully Automated Web Service Composition **81**

3.1 Introduction 81
3.2 Chapter Organization 88
3.3 The WSC-CQ Problem 89
3.4 Pre-processing of A Service Repository 92

3.5 The EDA-NHM Algorithm 94
3.5.1 Outline of EDA-NHM 95
3.5.2 A novel permutation-based representation 97
3.5.3 Application of NHM Construction and NHBSA . . . 99
3.6 The EDA-EHM Algorithm 101
3.6.1 Outline of EDA-EHM 102
3.6.2 Discovery of Service Dependency 103
3.6.3 Application of EHM Construction 105
3.6.4 GEHBGSA for sampling 107
3.7 Experimental Evaluation 110
3.7.1 Comparing EDA-NHM and EDA-EHM with PSO, FL, SearchPath 111
3.8 A Memetic Algorithm Based on EDA-NHM 117
3.8.1 Outline of the Memetic Algorithm 118
3.8.2 Application of Uniform Distribution Schema 119
3.8.3 Stochastic Local Search Operators 121
3.9 Experimental Evaluation 126
3.9.1 Comparing Memetic EDA-based Methods with EDA-NHM and MEFL 127
3.10 Summary 134

4 Multi-Objective Fully Automated Web Service Composition 137
4.1 Introduction 137
4.1.1 Introduction to the WSC-MO Problem 138
4.1.2 Introduction to the WSC-MQP Problem 140
4.2 Chapter Organization 145
4.3 The WSC-MO Problem 145
4.4 The MNSGA2-EDA Algorithm 146
4.4.1 An overview of MNSGA2-EDA 147
4.4.2 Outline of MNSGA2-EDA 148
4.4.3 Genetic Operators 150

4.4.4 Identify a Cluster Representative of Each Cluster . . 151
4.4.5 Learn a NHM Based on Cluster Representatives . . . 152
4.5 Experimental Evaluation . . . 153
4.5.1 Parameters sensitivity . . . 154
4.5.2 Comparing MNSGA2-EDA with NSGA-II, Hybrid and Hybrid-L . . . 156
4.6 The WSC-MQP Problem . . . 162
4.7 The PMFEA Algorithm . . . 164
4.7.1 Outline of PMFEA . . . 165
4.7.2 Permutation-based representation . . . 167
4.7.3 Assortative Mating . . . 167
4.7.4 Task Selection for Evaluations . . . 168
4.8 The PMFEA-EDA Algorithm . . . 169
4.8.1 Outline of PMFEA-EDA . . . 171
4.8.2 NHMs Learning and Sampling Solutions . . . 173
4.8.3 NHBSA for Multitasking Evolutionary Search . . . 174
4.8.4 Skill Factor Transmission . . . 175
4.9 Experimental Evaluation . . . 176
4.9.1 Comparing PMFEAs with FL . . . 177
4.9.2 Comparing PMFEA-EDA with PMFEA-EDA-WTO, PMFEA, EDA-NHM, and FL . . . 182
4.10 Summary . . . 186

5 Evolving Robust Composite Services for Dynamic Semantic Web Service Composition 189
5.1 Introduction . . . 189
5.2 Chapter Organization . . . 195
5.3 The RWSC-SF Problem . . . 195
5.4 GA-MC Algorithm to RWSC-SF . . . 198
5.4.1 Robustness Estimation . . . 198
5.4.2 Outline of GA-MC . . . 199

5.4.3 Robustness Estimation based on Monte Carlo Sampling . . . 200
5.5 GA-2Stage Algorithm to RWSC-SF . . . 203
5.5.1 Robustness Estimation . . . 205
5.5.2 Outline of GA-2Stage . . . 206
5.5.3 Archive-based adaptive evolutionary control . . . 208
5.5.4 Robustness Estimation based on a Lower Bound . . 209
5.6 Experimental Evaluation . . . 213
5.6.1 Comparing GA-MC against FL . . . 213
5.6.2 Comparing GA-2Stage with GA-RE, GA-MC and FL 218
5.7 Summary . . . 228

6 Conclusions **231**
6.1 Achieved Objectives . . . 232
6.2 Main Conclusions . . . 236
6.2.1 Explicit Distribution Models for Fully Automated Service Composition . . . 236
6.2.2 Neighbourhood Structure of Composite services . . . 237
6.2.3 Local Improvements on Pareto Solutions Using EDA 238
6.2.4 Using EDA to Achieve Effective and Efficient WSC-MQP . . . 238
6.2.5 The Application of Two-stage Robust Service Composition . . . 239
6.3 Practical Guidelines . . . 240
6.4 Future Work . . . 241
6.4.1 Miscellaneous Distribution Models and Sampling Techniques . . . 241
6.4.2 Miscellaneous Decoding Strategy for Permutations . 241
6.4.3 Many-Objective Optimisation . . . 242
6.4.4 Robustness estimation . . . 243

List of Figures

1.1 Research objectives and sub-objectives. 14

2.1 Functional properties of a web service. 29
2.2 Semi-automated web service composition process [123]. . . . 31
2.3 An example of service composition for a travel agency. . . . 33
2.4 An example of a component service for demonstrating QoSM. 34
2.5 Input and output-related concepts and instances described for MapGeneration service in Fig. 2.4. 34
2.6 Sequence construct and calculation of its QoS [221]. 37
2.7 Parallel construct and calculation of its QoS [221]. 38
2.8 Choice construct and calculation of its QoS [221]. 38
2.9 Loop construct and calculation of its QoS [221]. 39
2.10 An example of a vector-based representation in GA. 43
2.11 Examples of crossover and mutation in GA. 43
2.12 Examples of probabilistic models in EDA [139]. 47
2.13 Crowding distance calculation based on points marked with filled circles [54]. 49
2.14 Fast non-dominated sorting strategy in NSGA-II [54]. 50
2.15 An overview of the literature review. 56

3.1 An example of pre-processing of service repository for a service request T. 94
3.2 A process of generating composite services as permutations. 98

3.3 A different permutation produced by a decoding and encoding process. 99
3.4 An example of labeled $\mathcal{O}$. 105
3.5 An example of a DAG generated by GEHBGSA. 109
3.6 A comparison of the convergence curves of EDA-NHM, EDA-EHM, PSO, FL over execution time on WSC08-6 (the left) and WSC09-5 (the right). 116
3.7 An example of a constrained one-point swap on $[1, 2, 3 \mid 0, 4]$. 122
3.8 An example of two-point swap on $[1, 2, 3 \mid 0, 4]$. 123
3.9 An example of one constrained block-swap on $[1, 2, 3 \mid 0, 4]$. . 123
3.10 An example of layer-based one-point swap operation on $[1, 2, 3 \mid 0, 4]$. 124
3.11 An example of layer order breached by constrained one swap operation. 126
3.12 A comparison of the average convergence rate of EDA-NHM, EDA-EHM, PSO, FL over execution time (the left) and generation (the right) for WSC09-2. 132
3.13 A comparison of the percentage of better neighbours produced by four memetic algorithms along generations over 30 runs for WSC08-03. 133

4.1 Two composite booking services produced by TripPlanner. . 141
4.2 Generation updates in MNSGA2-EDA. 147
4.3 Examples of crossover and mutation for parents. 150
4.4 An example of identifying two cluster representatives. . . . 151
4.5 Mean hypervolume over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right) (Note: the larger the hypervolume the better). 161
4.6 Mean IGD over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right) (Note: the smaller the IGD the better). 161

4.7 Pareto optimal solutions obtained for tasks WSC09-3 (left) and WSC09-5 (right). 162
4.8 An example of a DAG-based solution decoded from a given permutation. 168
4.9 An example of neighborhood structure over four tasks. . . . 169
4.10 Generation updates in PMFEA-EDA. 170
4.11 Mean fitness over generations for tasks 1-4, for WSC09-3 (Note: the larger the fitness the better). 181
4.12 Mean fitness over generations for tasks 1-4, for WSC09-2 (Note: the larger the fitness the better). 185

5.1 Two-stage robust web service composition system. 197
5.2 A new permutation produced based on a sampled scenario. 202
5.3 Generation updates with an adaptive evolutionary control in GA-2Stage. 204
5.4 A new permutation produced based on a sampled scenario. 212
5.5 Mean fitness values tested on near-optimal solutions found by GA-MC over a set of increasing N for OWLS-TC 03 215

6.1 A decision tree to guide practitioners for choosing algorithms240

List of Tables

2.1 Summary of web service composition approaches. 57

3.1 QoS calculation for a composite service expression C. 91

3.2 Mean fitness values for EDA-NHM and EDA-EHM in comparison to PSO, FL and PathSearch. (Note: the higher the fitness the better) 113

3.3 Summary of statistical significance tests for fitness, where each column shows win/draw/loss score of an approach against others. 113

3.4 Mean execution time (in s) for EDA-NHM and EDA-EHM in comparison to PSO, FL and PathSearch. (Note: the shorter the time the better) 115

3.5 Summary of statistical significance tests for execution time (in s), where each column shows win/draw/loss score of an approach against others. 115

3.6 Mean fitness values for our memetic EDA algorithms in comparison to EDA-NHM and MEFL. (Note: the higher the fitness the better) 129

3.7 Summary of statistical significance tests for fitness, where each column shows win/draw/loss score of an approach against others. 129

3.8 Mean execution time (in s) for our memetic EDA algorithms in comparison to EDA-NHM, MEFL. (Note: the shorter the time the better) . . . 131
3.9 Summary of statistical significance tests for execution time (in s), where each column shows win/draw/loss score of an approach against others. . . . 131

4.1 Mean IGD of MNSGA2-EDA with three groups of parameter settings over WSC08-3 (Note: the lower the IGD the better). . . . 155
4.2 Mean Hypervolume of MNSGA2-EDA with three groups of parameter settings over WSC08-03 (Note: the higher the hypervolume the better). . . . 155
4.3 Mean execution time (in seconds) for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the shorter the time the better). . . . 157
4.4 Summary of statistical significance tests for the execution time, where each column shows the win/draw/loss score of one method against a competing one for all tasks of WSC08 and WSC09. . . . 157
4.5 Mean IGD for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the lower the IGD the better). 158
4.6 Summary of statistical significance tests for IGD, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09. 159
4.7 Mean Hypervolume for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the higher the hypervolume the better). . . . 159
4.8 Summary of the statistical significance tests for hypervolume, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09. . . . 160

4.9 Mean fitness values for our approach in comparison to FL [47] (Note: the higher the fitness the better). 178
4.10 Mean execution time (in s) for our approaches in comparison to FL (Note: the shorter the time the better). 180
4.11 Mean fitness values of solutions per task for our approaches in comparison to PMFEA, EDA-NHM and FL [47] (Note: the higher the fitness the better). 183
4.12 Mean execution time (in s) over all the tasks for our approaches in comparison to PMFEA [185], EDA-NHM and FL [47] (Note: the shorter the time the better). 184

5.1 Mean fitness values tested based on the baseline solutions for our approach in comparison to FL. (Note: the higher the fitness the better) 215
5.2 Mean execution time (in seconds) observed for our approach in comparison to FL at the design stage. (Note: the shorter the time the better) 217
5.3 Mean execution time (in milliseconds) per scenario by local search based on the baseline solutions found by our approach in comparison to FL. (Note: the shorter the time the better) 217
5.4 Mean fitness values tested based on the baseline solutions for GA-2Stage in comparison to GA-RE, FL and GA-MC. (Note: the higher the fitness the better) 221
5.5 Summary of statistical significance tests for mean fitness values, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09. 222
5.6 Mean execution time (in s) observed for GA-2Stage in comparison to GA-RE, FL and GA-MC at the design stage. (Note: the shorter the time the better) 224

5.7 Mean execution time (in ms) per scenario by local search based on the baseline solutions found by GA-2Stage in comparison to GA-RE, FL and GA-MC. (Note: the shorter the time the better) 225
5.8 Summary of statistical significance tests for mean execution time of the design stage, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09. 226
5.9 Summary of statistical significance tests for mean execution time per scenario for local search at the execution stage, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09. 226
5.10 Results of three statistical correlation tests using Pearson, Kendall's tau, and Spearman's rho. 227

Glossary of Terms

We provide a brief definition on some recurrent terms in this thesis as follows:

Abstract service A slot that specifies inputs and outputs of concrete web services.

Abstract service workflow A pre-defined service execution workflow that chains abstract services.

Assortative mating A breeding method of multi-factorial evolutionary algorithm, and it employs two genetic operators — crossover and mutation to produce offspring for multiple tasks.

Constructs Basic structures of composite services that determine how services are associated with each other.

Crossover A genetic operator that can generate two child solutions by exchanging some parts of two selected parent solutions.

Decoding An interpretation of a candidate solutions from indirect representation (e.g., vectors or permutations) to execution workflows of services (e.g., directed acyclic graphs).

Estimation of Distribution Algorithm (EDA) An evolutionary computation technique that learns probabilistic models over a set of promising candidate solutions. These models are adjusted iteratively with the aim to sample fitter candidate solutions.

Evolutionary Multi-objective Optimization (EMO) A collection of evolutionary algorithms that simultaneously optimize more than one objective function in the field of multi-objective optimization paradigms.

Evaluation A process that uses functions to measure the fitness (i.e., goodness) of candidate solutions.

Evolutionary computation A family of optimization techniques that are inspired by Darwinian evolutionary principles. Based on a population of solutions that is subject to natural selection, new fitter solutions are produced in the next generation.

Fully automated service composition A strategy of performing service composition, it constructs workflows of composite services simultaneously with selections of atomic services.

Genetic algorithm (GA) An evolutionary computation technique that evolves fitter candidate solutions with the help of genetic operators.

Hypervolume A performance evaluation metric for multi-objective optimizations algorithms. Hypervolume measures the dominated volume covered by a reference point (e.g., a point (1, 1) utilized in a minimization problem) and the front evolved by a multi-objective algorithm. The higher the Hypervolume, the better the algorithm.

Individual A candidate solution in a population.

Inverted generational distance (IGD) A performance evaluation metrics for multi-objective optimization algorithms. IGD measures the distance from the nearest point of the non-dominated set produced by an multi-objective algorithm to a true Pareto front. The lower the IGD, the better the algorithm.

Local search A heuristic method for solving optimization problems. It aims to generate good neighbouring solutions by apply local changes on a given candidate solution until a stopping criterion is met.

Multi-factorial evolutionary algorithm (MFEA) An evolutionary optimization algorithm that can efficiently evolve candidate solutions for solving multiple tasks concurrently via implicit parallelism of population-based search.

Mutation A genetic operator that generates a child candidate solutions by apply a small change on a selected parent solution, maintaining the diversity of a population from generation to generation.

Neighbourhood A set of individuals that are generated based on a given individual via small modifications.

Non-dominated sorting genetic algorithm II (NSGA-II) A multiobjective form of GA that aims to produce a set of approximated Pareto solutions. The key idea of this algorithm is a fast non-dominated sorting strategy based on the concept of Pareto dominance and the crowding distance.

Ontology Semantic descriptions that are used to describe the inputs and the outputs of web services. By utilizing these semantic descriptions, semantic web service can be matched based on the inputs and outputs.

Pareto dominance A vector $\vec{x}$ dominates another vector $\vec{x}'$ if all the objectives values of $\vec{x}$ are better or equal to those of $\vec{x}'$, and at least one objective value of $\vec{x}$ is better than those of $\vec{x}'$. Otherwise, these two vectors are non-dominated to each other.

Pareto front A set of candidate solutions are non-dominated to each other based on multiple objectives, the number of which is more than

one.

Quality of semantic matchmaking (QoSM) A quality measurement on the matchmaking of any two web services via their functional properties, i.e., inputs and outputs.

Quality of service (QoS) A quality measurement on the non-functional properties of any web service. It often refers to four quality criteria, i.e., time, cost, availability and reliability.

Representation A key component of EC techniques that is used to represent an individual. A properly designed representation always has a proper mapping between the phenotype and genotype spaces, which is critical for EC techniques.

Reproduction A genetic operator that clones a parent candidate solution as an offspring for the next generation.

Service repository A set of services available over a network.

Service discovery A process of identifying services, satisfying the required inputs and outputs.

Skill factor The skill factor of an individual denotes the most effective task among multiple tasks in multi-factorial evolutionary algorithm.

Semi-automated service composition. A strategy of performing service composition, it selects a suitable atomic service for each abstract service slot of a pre-defined abstract service workflow.

Task A composition task (also called a service request) that consists of a set of provided task inputs and a set of required task outputs.

T-test A most commonly used statistical hypothesis test that consists of one-sample T-test and two-sample T-test. In one-sample T-test,

it is used to determines whether there is a significant difference between a given mean value and the mean of one group of sample. In two-sample T-test, it is used to determine whether there is a significant difference between the means of two group samples. Often, a normal distribution is assumed to be followed by these samples.

Vertical cultural transmission In the multi-factorial evolutionary algorithm, the offspring is allowed to imitate the skill factor of any one of their parents.

Wilcoxon rank-sum test Similar to T-test, but Wilcoxon rank-sum test does not assume that a normal distribution is followed by the samples.

Web service A software module that are self-describing, self-contained and available over the internet.

Web service composition A set of web services that are loosely coupled via their inputs and outputs, providing more complex functionalities for users.

WSC-CQ A fully automated semantic **W**eb **S**ervice **C**omposition problem that aims to optimize **C**omprehensive **Q**uality (i.e., a combination of QoSM and QoS) of composite services.

WSC-MO A fully automated semantic **W**eb **S**ervice **C**omposition problem for **M**ultiple conflicting **O**jectives. This problem aims to simultaneously optimize multiple conflicting quality criteria in comprehensive quality and produces a set of approximated Pareto-optimal composite services.

WSC-MQP A fully automated semantic **W**eb **S**ervice **C**omposition problem for **M**ultiple user segments with distinctive **Q**oSM **P**references. This problem aims to simultaneously optimize comprehensive quality with distinctive QoSM preferences for different user segments

and produces a set of near-optimal composite services, each of which serves one user segment.

RWSC-SF A novel ***R**obust **W**eb **S**ervice **C**omposition* problem for handling stochastic **S**ervice **F**ailures. This problem aims to construct baseline composite services by explicitly considering stochastic service failures at the design stage. These baseline composite services can cope with unexpected service interruptions in a robust manner to resume the high quality at the execution stage.

Chapter 1

Introduction

1.1 Problem Statement

Service-oriented computing (SOC) is a popular computing paradigm that employs services as fundamental elements to achieve the agile development of cost-efficient and integrable enterprise applications in heterogeneous environments [133, 134]. It aims to be platform-neutral and language-agnostic, enabling integrable and seamless communication among those existing or newly-built independent services. *Service-Oriented Architecture* (SOA) could abstractly implement a service-oriented paradigm of computing. During the ongoing development, SOA has been contributing to the reuse of software components, from functions to units, and from units to services [22, 129]. SOA can be implemented with *web services*, which are designated as "modular, self-describing, self-contained applications that are available on the Internet" [39]. Service providers, such as AWS (Amazon Web Services [212]), advertise web services using formal description standards. These standards play a significant role in registering, invoking, and grounding web services on the web. For example, UDDI [38] and WSDL [99] are commonly used standards.

Since users' requirements cannot always be satisfied by some atomic web service, *web service composition* aims to loosely couple a set of web

services to provide a value-added composite service that accommodates users' requirements. To perform service composition with valid functionality, *interoperability* of services becomes the first essential requirement for web services to be composed [61]. The *interoperability* of web services presents challenges to syntactic and semantic descriptions. The syntactic descriptions can often be addressed by XML-based standards [219], such as $WSDL$. Particularly, services can be composed together based on the syntactical matches of input-output parameters of web services. Compared to syntactic descriptions, semantic descriptions enable better collaboration via ontology-based semantics [128], in which many standards are established, such as OWL-S [119], WSMO [100], SAWSDL [94], and SWSO [141]. When input-output parameters of web services cannot be matched syntactically, they could be matched semantically. Web service composition that aims to find composed services with optimal semantic matches of inputs and outputs, gives birth to *semantic web services composition*. Therefore, the quality of semantic matchmaking (QoSM) becomes an essential functional quality concern for service users, raising researchers' interest in searching for composite services with high QoSM.

In addition to the functional aspect of QoSM, it is also important to take the non-functional aspect into account while performing web service composition. This non-functional aspect is often measured by Quality of Service (QoS) [61], which often refers to four commonly used quality criteria, i.e., cost, time, reliability, and availability. Different from *semantic web services composition* discussed above, *QoS-aware service composition* aims to find composite services with optimized QoS. Such solutions with optimized QoS are preferred by users because they can fulfill users' functional requirements with good QoS, such as low price and short response time. In practice, when users have clear preferences on each quality aspect in QoS, i.e., the importance of each quality can be weighted by users, a single composite service can often be returned for a service composition request. Such a solution is optimized with a QoS score measured by a weighted

aggregation of all quality criteria in QoS. Service composition that aims to construct such a solution is referred as *single-objective QoS-aware service composition*. In contrast, *multi-objective QoS-aware service composition* focuses on finding a set of optimized trade-off solutions (i.e., approximated Pareto solutions) because users could have no clear preferences on QoS before they see the trade-offs of the solutions. For example, users might be willing to trade one objective, cost, for another objective, response time [113], where cost and response time are conflicting QoS criteria that can be optimized independently. In a nutshell, both *single-objective QoS-aware service composition* and *multi-objective QoS-aware service composition* are important research areas that serves users' different needs.

Services available for composition can experience QoS changes over time. On the one hand, new services are published, and old ones are modified or removed due to the changes in users' demands or service providers [96]. In fact, newly published services might be more suitable because they could be faster, cheaper, and aggregate multiple functionalities of a composite service. On the other hand, services being composed can become unavailable at the time of execution, which may be due to service overload, software/hardware failures, and network issues [81]. Such changes may render existing composite services invalid or present new opportunities for building more preferable composite services.

Delivering composite services with reliable QoS is a critical and significant challenge. This dynamic problem is often referred to as *dynamic web service composition*. In practice, QoS changes can be related to many different quality criteria in QoS [48], such as response time, throughput, failure probability, availability, price, and popularity. Among these QoS criteria, the failure probability of web services is the most critical uncertainty [23]. This is because the composite services constructed at the design stage can become completely useless at the time of its execution if any component service fails. *Service failure probability* can be often approximated by dividing the number of failed invocations by the total number of invocations

conducted in the past [232].

Different service composition approaches [6, 30, 42, 45, 72, 80, 101, 108, 117, 145, 152, 192, 221] have been proposed to cope with the composition challenges discussed above. These works can be grouped into two main categories: *semi-automated web service composition* and *fully automated web service composition* with two different assumptions based on whether workflows of service composition are known in advance [147]. The first group of research works assumes that users know an abstract service composition workflow, and all the composite services produced by the composition system must strictly obey this workflow. Therefore, semi-automated QoS-aware web service composition turns to select concrete services for each abstract service in the given workflow to achieve the best possible QoS. Due to the tremendous growth in enterprise applications, the number of web services has increased dramatically at an unprecedented pace [5]. The process of designing abstract workflows manually is fraught with difficulties. The second group of research works on *fully automated web service composition* does not rely on any existing workflow. Instead, a composite service workflow will be constructed from scratch while selecting and connecting concrete atomic services from the service repository [147]. Therefore, this construction process not only selects services to achieve best QoS but also searches for the optimal service workflows. Apparently, compared to semi-automated web service composition, fully-automated web service composition is more difficult, but it also opens new opportunities to improve QoS and QoSM without being restricted to pre-defined workflows.

The different service composition approaches discussed above can also be classified based on the assumptions on QoS, i.e., QoS of a web service is either static or dynamic. The first group, namely *static web service composition*, assumes that the QoS of web services seldom changes or does not change at all. In this group, QoS often refers to the mean values of the historical QoS, which is accessible to service users [89]. In contrast, the second

group, namely *dynamic web service composition*, focuses on handling dynamic QoS. In this group, dynamic QoS values vary in bounded-interval values [9, 125, 190] or can be estimated based on the past QoS distributions [6, 30, 80, 108]. Apparently, these two groups of works handle QoS in different ways, but they still attract many ongoing research works in both areas.

The service composition problems discussed above are known to be NP-hard (i.e., **N**on-deterministic **P**olynomial-time **H**ard) problems [123], which means it might be difficult to find optimal solutions. To cope with such a complexity, a variety of techniques have been investigated to solve this problem. In the literature, these techniques can always fall into three categories — *exact optimization methods*, *heuristic methods* and *meta-heuristic methods*. Firstly, exact optimization methods, such as 0-1 Linear Programming [130], can produce optimal solutions when the searching space is small. However, they also have critical disadvantages, such as poor scalability and high consumption of computation resources. Compared to the exact optimization methods, heuristic methods can improve the scalability to some extent, but they can be easily trapped in a local optimum. For example, $A^{\star}$ search is used to efficiently find a near-optimal composite service [151]. Lastly, meta-heuristic methods can reach a good trade-off between quality of composite services and computational time of algorithms. The existing work mainly focuses on developing meta-heuristic methods to find near-optimal composite services efficiently. Particularly, *Evolutionary Computation (EC) techniques* are widely studied in combinatorial optimization for web service composition [42, 45, 47, 117, 140, 152, 221]. EC techniques are particularly useful in practice as they can efficiently find "good enough" (i.e., near-optimal) composite services. For example, Particle Swarm Optimization (PSO) is utilized to find an optimized queue of services (i.e., a permutation), which can be decoded into a corresponding composite service [45].

A variety of EC techniques have been demonstrated to be highly

promising in solving the *QoS-aware web service composition* problem. Particularly, direct and indirect representations have been carefully investigated since they could significantly affect the performance of EC-based approaches. Direct representations, such as tree- and graph-based representations [41, 152], represent composite services intuitively from the human perspective, displaying actual execution flows of composite services. By contrast, indirect approaches [47, 45] often represent composite services as permutations, which require a decoding process to build up actual execution workflows. Apart from EC-based approaches, hybridized approaches that combine *Artificial Intelligence (AI) planning-based approaches* and EC techniques are introduced [137, 191]. AI planning is mainly utilized to solve the fully-automated web service composition problem via a plan-making process. For example, given a service composition request (consisting of provided inputs and required outputs), a composite service can be a planning process, with the inputs as the initial state and the outputs as the desired goal state, and the component services as actions triggered by one state and resulted in another state. Combining EC with AI planning techniques can ensure the valid functionality of composite services in the context of fully automated service composition while optimizing the QoS of composite services [42, 117].

In this thesis, we will focus on developing EC-based techniques with AI planning algorithms to efficiently find near-optimized composite services with valid functionalities in the context of fully automated web service composition. To find near-optimal composite services, we will consider both QoSM and QoS so as to cope with both the functional requirements and non-functional preferences from users. We aim to address this fully automated web service composition problem in single-objective, multi-objective, and dynamic contexts. The motivations related to these contexts will be discussed in Section 1.2.

1.2 Motivations

The motivations of this proposed research lie in four key aspects that simultaneously account for: 1. *Simultaneously handling QoSM and QoS*. 2. *Multi-objective semantic service composition*. 3. *Dynamic semantic service composition*. 4. *Hybridized techniques for automated web service composition*. These key aspects will be discussed in more detail below.

Comprehensive Quality of Semantic Web service Composition

QoS is often utilized to measure and distinguish composite services based on users' non-functional preferences. Besides QoS, outputs of web services often do not perfectly match inputs of web services [103]. Thus, QoSM becomes a critical measure of the validity of composite services. In fact, many different composite services can meet a user request but differ in both QoS and QoSM. Let's consider an example of a service request for a weather forecasting service, which provides weather information (i.e., *weather info*) of a given city (i.e., *city*). Suppose that two services can be considered for this purpose. One service S_1 produces an output *weather info* by giving an input *location* at a *price* of 6.72 cents. The other service S_2 produces an output *weather info* by giving an input *city* at a *price* of 16.87 cents. According to the semantic descriptions regarding the inputs and the outputs, *city* better matches the service request than *location*, so S_2 clearly enjoys better QoSM than S_1. However, S_2 has a negative impact on the QoS of the composite service because its price is much higher. One can easily imagine that similar challenges frequently occur when looking for service compositions. Hence, a good balance between QoSM and QoS should be studied in service composition problems.

The majority of existing works on service composition address QoS and QoSM separately. For QoSM, many related works fall into the context of service discovery, which aims to find a concrete atomic service with the best semantically matched inputs and outputs. Other works [16, 25, 122]

do consider service composition but they focus on minimizing the number of component services in a composite service. However, QoS of a composite service is not only determined by the number of component services but also QoS of its component services. Therefore, these approaches cannot guarantee an optimized QoS. On the other hand, many efforts have been devoted to studying QoS-aware web service composition [41, 45, 72, 117, 145, 221] in the past few years. Among these works, some works consider the differences between semantic matchmaking types (e.g., Exact and Plugin matches [132], see details in Sect.2.1.2) when they compose services. However, they treat different matchmaking types equally and do not evaluate the QoSM of composite services during the search process. Few works address both QoSM and QoS for the web service composition problem. To the best of our knowledge, [58, 101, 143] report about the first attempts that consider both QoSM and QoS, but these works can only support semi-automated web service composition. To address the limitations discussed above, a comprehensive quality model that jointly considers QoSM and QoS should be proposed. Meanwhile, this model should support easy computation of QoS and QoSM for composite services with varied workflow structures in fully automated service composition.

Multi-Objective Web Service Composition

In this subsection, we discuss the motivations of two different categories of the multi-objective web service composition problem. The first category of multi-objective web service composition is different from single-objective web service composition. This is because the number of objective functions used in the multi-objective web service composition is more than one. In single-objective approaches, one composite service is often returned for a composition request, where the preferences of each quality criteria in QoS are provided by users. A weighted sum of different quality criteria is often used to form such a single objective. However, users do not always have clear preferences for different quality criteria before they

see trade-off solutions. Therefore, multi-objective optimization becomes a natural way to generate a set of trade-off solutions that cope with conflicting quality criteria. The majority of existing works on multi-objective service composition [113, 172, 213, 214] can only support semi-automated service composition while handling conflicting objective functions on QoS. That is, they purely focus on selecting atomic services into the abstract service slots to achieve the best possible overall QoS. However, to our best knowledge, [40, 46] report the first two attempts to solve fully automated multi-objective service composition. Particularly, two hybrid approaches [40] (called Hybrid and Hybrid-L) that combines the use of two multi-objective optimization algorithms, i.e., NSGA-II and MOEA/D, achieves outstanding performances in finding good Pareto solutions. Particularly, Hybrid-L further extends Hybrid by incorporating local search. Despite this recent success, the large number of decomposed subproblems is predefined (e.g., 500 subproblems in Hybrid and Hybrid-L in [40]), and a simple form of local search (i.e., so-called one-point "swap" in Hybrid-L) is ineffective and inefficient to make local improvements because it is randomly applied to subproblem representatives without focusing on the most suitable candidate solutions. Meanwhile, each one-point "swap" local search searches solutions in the space of candidate solutions based on only one solution (i.e., selected subproblem representative), ignoring any information of other promising candidate solutions that could be jointly used for guiding the local search. Therefore, the effectiveness and efficiency of the local search need to be further improved in our thesis.

The second category of multi-objective web service composition considers multiple objectives, which jointly optimize QoSM and QoS with segment users' distinctive preferences on QoSM. Such a multi-objective problem can be treated as a multitasking problem that simultaneously searches for solutions for multiple service composition requests of different user segments. Although multiple service requests can be tackled separately as multiple purely single-objective problems. However, simulta-

neously solving them can save a large amount of computation time. Furthermore, during the process of searching for solutions over multiple requests, good solutions for one request can be helpful to evolve solutions for the other requests. That is, some knowledge of good solutions can be shared for different requests. Existing service composition algorithms are designed primarily to solve each service composition request independently [152, 42, 182, 221], ignoring similarities between different requests that could be dealt with collectively. Therefore, there is a lack of research works in handling such a category of multi-objective web service composition problem.

Herein we briefly introduce the second category of multi-objective web service composition problem: Due to a significant increase in service composition requests, many requests have similar functional requirements (i.e., input and output) requirements. In a market-oriented environment, to distinguish different types of users, service developers often strategically group all the users (i.e., service requesters) into several segments, e.g., platinum, gold, silver, and bronze users, and provide different composite services for different segment users with distinct QoSM. Although different segment users have distinctive preferences on QoSM, their requests share the same functional requirements. Therefore, these requests can be dealt with collectively as a multitasking problem.

Dynamic Semantic Web Service Composition

Web services can experience QoS changes over time [197]. These changes are prevalent over the internet. Therefore, remedy actions must be taken if the QoS and the functional validity of the original composite services cannot be guaranteed any further [106].

A variety of strategies [8, 18, 93, 124, 157, 173, 215] have been developed to cope with dynamic web service composition, once negative QoS changes and/or service failures (i.e., any component service of composite services can not be executed) are detected. Many of them work on ser-

vice reconfiguration techniques. For example, some approaches [8, 18, 93] extend WS-BPEL (i.e., a standard executable language for specifying service composition) with Event Condition Action to guide the operations for the reconfiguration of executable composite services. However, such approaches are difficult to manage and error-prone because its success strongly relies on dynamic events that are manually enumerated and configured using WS-BPEL. EC techniques have shown promise in dealing with dynamic optimization problems via a re-optimization process [210]. Particularly, existing works [6, 9, 30, 80, 108, 125, 190] track the optimum of composite services with respect to QoS changes. For example, some works [9, 125, 190] continuously re-optimize QoS of composite services. However, the frequency of the re-optimization is assumed to happen periodically (e.g., every few generations [190] or every period of time [9, 125]). A few works [6, 30, 80, 108] assume that the changes of QoS follow some historical patterns and can be predicted for the future. In reality, services often fail sporadically in a highly unpredictable manner, and sufficient historical data is not always available for newly registered web services for building up a reliable a prediction model. Other works [124, 157, 173, 215] utilize decision tree learning, reinforcement learning and Rtree query techniques for re-selecting suitable component services. These re-selection strategies do not allow any changes to the workflow structure of any composite service. In other words, they can only cope with semi-automated service composition. Apart from the discussed limitations above, existing works only focus on handling dynamic changes at the execution stage, ignoring the potential benefits of handling such changes at the design stage. Therefore, it would be very interesting and motivating to deal with QoS changes at both the planning stage and the execution stage.

Hybridized methods for Web Service composition

Various techniques have been utilized to solve service composition, such as AI planning, local search, and EC techniques [60, 136, 145, 192]. AI

planning ensures the functional validity of composite services. However, optimizing QoS is not its focus. Local search often exhaustively searches neighboring solutions from a starting solution point, until a stopping criterion is met, such as an optimal solution found or a time-bound reached. This technique can make fine-grained local improvements, but can easily be trapped in local optima. EC techniques are good at solving global optimization problems and are less prone to premature convergence in complex search spaces [52]. They often utilize knowledge, which is defined as useful information acquired through experience (i.e., promising composite services) to evolve new solutions. Such knowledge can be implicit or explicit based on a practical or theoretical understanding of promising solutions. By iteratively updating and utilizing the knowledge, new candidate solutions are generated until the best solutions are found. Hybridized methods can outperform methods that utilize a single technique only in finding high-quality solutions because they are designed to combine the advantages of each single technique [70]. For example, they could benefit from escaping local optima more easily and improving the rate of convergence [149]. However, hybridized methods often consume more computation time for their execution.

Many researchers only use a single AI planning technique for service composition problems based on classical planning algorithms [118, 137] (e.g., GraphPlan [20] and Enhanced Planning Graph (EPG) [109]). To support optimization in AI planning, a combined Graphplan and Dijkstra's algorithm [60] is proposed to find a functionally valid solution with the aim to minimize the number of component services or optimize a single criteria in QoS. To cope with multiple quality criteria in QoS, some researchers combine AI planning with both EC techniques and local search for solving QoS-aware service composition [44, 136]. Despite some recent successes in hybridized methods, opportunities remain to further improve the performance of such hybridized methods, such as reducing the execution time of the hybridized methods while retaining or improving their

ability of finding high-quality composite services.

1.3 Research Goals

The overall goal of this thesis is to *develop novel, effective and efficient EC-based hybrid approaches for comprehensive quality-aware fully automated semantic web service composition*. More specifically, the research focus will be on: (1) developing single-objective EC-based approaches that jointly optimize QoS and QoSM in our proposed comprehensive quality, (2) developing multi-objective EC-based approaches for optimizing multiple quality criteria involved in our comprehensive quality, and (3) developing EC-based composition approaches to dynamic web service composition in consideration of stochastic service failures. Our research aims to develop EC-based approaches combined with local search and/or AI planning techniques for effectively and efficiently handling the service composition problems listed as (1), (2) and (3) above. The research goal described above will be achieved by completing the following set of objectives, which are also outlined in Fig. 1.1.

1. **To develop EC-based approaches to comprehensive quality-aware fully automated semantic web service composition that simultaneously optimizes both QoSM and QoS**. Particularly, we extend existing works on QoS-aware service composition by jointly optimizing QoSM and QoS, which will be formalized through a comprehensive quality model. In addition, representations of the composite services are a key aspect of EC-based approaches. Previous studies have shown that permutation-based representation of composite services contributes to excellent performance in searching high-quality solutions for QoS-aware web service composition [44, 45], and graph-based representation is capable of presenting the information of QoSM and QoS on a weighted directed acyclic graph (DAG)

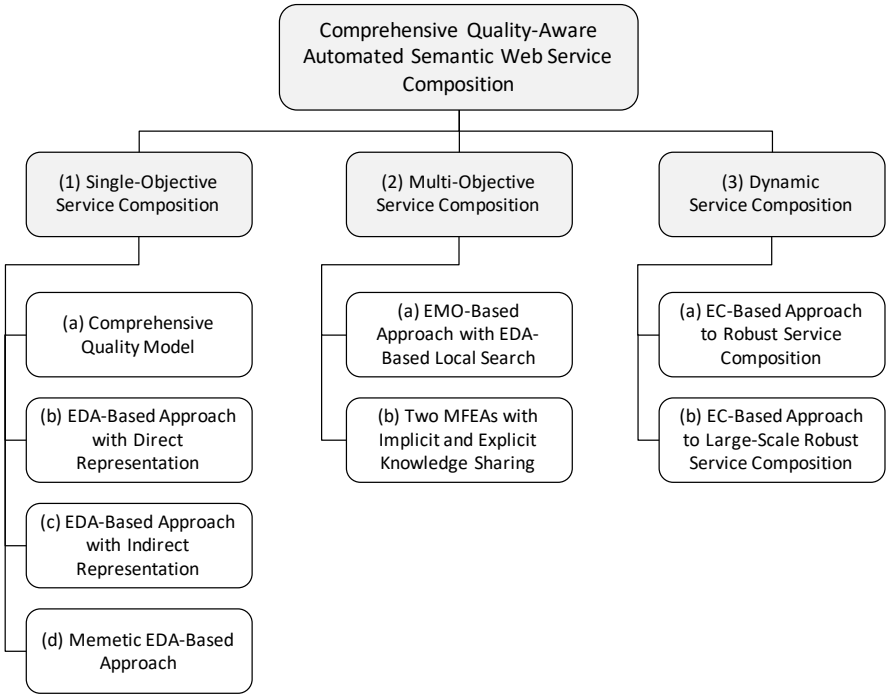


Fig. 1.1: Research objectives and sub-objectives.

[101] as the most intuitive way of presenting semantic composite services. We will investigate the effectiveness of both representations in solving single-objective comprehensive quality-aware fully automated semantic web service composition. Apart from representations, conventional EC techniques have been used to solve QoS-aware web service composition, such as GA, GP and PSO, which implicitly use the knowledge of good solutions from previous generations. To effectively use the knowledge explicitly from good solutions, Estimation of Distribution Algorithm (EDA) samples promising solutions from the knowledge encoded by a probabilistic model based on the distribution of a sub-population, which is formed from a set of parent individuals. It has been suggested for other combinatorial optimization problems, such as arc routing and assembly flow-shop scheduling [189, 195], information revealed by the explicit

knowledge, in particular, distributions and dependencies of variables in permutation-based solutions, can make the searching ability of EDA more effective and efficient [29]. Therefore, we investigate EDA-based service composition approaches by determining suitable probabilistic models and proposing sampling methods for effectively and efficiently solving fully automated service composition. We will investigate the following sub-objectives to handle this objective.

(a) *To propose a comprehensive quality model that addresses QoSM and QoS simultaneously to reach a desirable compromise.* This sub-objective is to establish a quality model for computing QoSM and QoS of composite services. Such quality model can be evaluated on service composition tasks using the most popular benchmark datasets, e.g., WSC-08 [15], WSC-09 [92] and OWLS-TC [97].

(b) *To propose an EDA-based approach with an indirect representation for fully automated comprehensive quality-aware web service composition.* This sub-objective is to propose an EDA-based approach with a novel permutation-based representation that captures the information of service positions in composite services. The permutation-based representation should allow reliable and accurate learning of a suitable distribution of composite services with varied workflows in fully automated service composition. Such a distribution can be used to sample new promising candidate solutions with near-optimal comprehensive quality.

(c) *To propose an EDA-based approach with a direct representation for fully automated comprehensive quality-aware web service composition.* This sub-objective is to propose an EDA-based approach that uses a graph-based representation to learn a suitable distribution of composite services that is different from the one talked

above. This distribution must naturally capture the essential ingredients for building graph-based composite services, i.e., service dependencies of promising graph-based solutions. This is because any graph-based solution can be disassembled into a set of service dependencies. In addition, to ensure the functional validity of sampled composite services, a novel sampling technique must be proposed to sample such solutions from the learned distribution.

(d) *Extend our EDA-based approach with a local search for fully automated comprehensive quality-aware web service composition.* This sub-objective is to extend one of the above EDA-based approaches (that presents the highest effectiveness) by introducing local search into its evolutionary process, called memetic EDA-based approaches. To achieve this goal, we will develop a novel strategy for effective interactions between our EDA-based approach and local search procedures. Moreover, we will propose domain-specific local search operators, which can properly form the neighborhood of composite services.

2. **To develop multi-objective approaches to fully automated comprehensive quality-aware semantic web service composition**. In this objective, we study two different categories of the multi-objective service composition problem. The first one refers to independently optimizing multiple conflicting quality criteria (i.e., QoS and QoSM) in our comprehensive quality model, producing a set of approximated Pareto composite services. Despite some recent successes, opportunities still remain to improve both the effectiveness and efficiency of the state-of-the-art fully automated and multi-objective service composition approaches, such as Hybrid and Hybrid-L discussed in Sect 1.2, by addressing their limitations. The second category of multi-objective service composition is related to multiple objectives that aim to jointly optimize QoSM and QoS with segment

users' distinctive preferences on QoSM, as discussed in Sect 1.2. Particularly, we will formulate these objectives as multiple optimization tasks, which can be solved via multi-factorial evolutionary algorithms (called MFEAs, see the introduction to MFEAs in Sect. 2.1.3).

(a) *To develop an EMO-based approach with a local search for multi-objective fully automated semantic web service composition*. This sub-objective corresponds to the first category of multi-objective semantic web service composition. Particularly, we employ a popular EMO technique in [40, 46], namely, NSGA-II, to multi-objective web service composition by incorporating a novel model-guided local search based on EDA. This local search is expected to address the limitations of a simple form local search (i.e., so-called one-point "swap") in Hybrid-L [40] discussed in Sect 1.2, and make effective local improvements on Pareto solutions found by NSGA-II via sampling. We expect that our new approach can achieve high performance in finding better approximated Pareto composite services efficiently.

(b) *To develop fully automated multi-objective web service composition approaches subject to multiple segment users' preferences*. This sub-objective corresponds to the second category of multi-objective semantic web service composition. We treat this sub-objective as a multitasking problem. Particularly, the multiple objectives that aim to jointly optimize QoSM and QoS with segment users' distinctive preferences on QoSM are formulated as multiple independent service requests. Each request (i.e., a composition task) is formulated as a request from one user segment with its corresponding QoSM preferences. Subsequently, to simultaneously solve the multiple requests, we will propose novel MFEAs with permutation-based representations. Meanwhile, to improve the effectiveness of MFEAs in finding high-quality

solutions with respect to the multiple requests, we will investigate both implicitly and explicitly learning and sharing the knowledge of good solutions for multiple service requests, inspired by assortative mating of MFEA (see the concept of assortative mating in Sect. 2.1.3).

3. **To develop EC-based approaches to effectively and efficiently solve dynamic semantic web service composition with respect to service failures**.

   The execution of composite services obtained from the design stage may fail due to unexpected QoS changes at the execution stage. In this objective, we will handle the most critical uncertainty in web service composition — stochastic service failures. Although current works take potential service failures into account, these works handles service failures only at the execution stage, ignoring the potential efforts that could be made at the design stage. In fact, for a service composition system, more computation time can be easily allocated to the design stage for handling this problem. Furthermore, the assumption of periodical changes or sufficient historical QoS data for predicting QoS changes poses noticeable feasibility challenges in these works. This is because services often fail sporadically in a highly unpredictable manner in the real world. To cope with these limitations, we propose an novel robust service composition problem with the goal of building robust composite services at the design stage via EC techniques. These composite services are expected to effectively and efficiently handle service failures in a robust manner.

   (a) *To propose an EC-based approach to robust service composition with stochastic service failures.* In this sub-objective, we firstly aim to propose a novel robust service composition problem that specifically handles stochastic service failures, targeting both the design stage and execution stage. GA is a popular EC technique

that has successfully solved several challenging service composition problems [44, 47]. Therefore, we will propose a GA-based approach with a novel robustness evaluation method to generate robust baseline solutions (i.e., composite services). Particularly, the robustness of composite services in terms of expected comprehensive quality (i.e., a combination of QoSM and QoS) is estimated via an application of Monte Carlo sampling as a fitness function. In the event of stochastic service failures, the composite services with high robustness are expected to continue to work reliably or be easily repaired with a negligible impact on the quality at the execution stage.

(b) *To propose an EC-based approach to large-scale robust service composition with stochastic service failures.* In this sub-objective, we propose a novel, effective and efficient robustness estimation method, which is expected to work reliably for large-scale robust service composition. The large-scale refers to a significant increase in the size of the service repository. The method is employed as a fitness function that allows composite services evolved by GA to be accurately ranked during the evolutionary process. To further reduce the overall computation time of our GA-based approach while maintaining its effectiveness in finding composite service with high robustness, an evolutionary control strategy is introduced into the evolutionary process of the GA-based approach.

1.4 Major Contributions

This thesis proposes three major contributions to the state-of-the-art fully automated web service composition:

1. This thesis proposes two EDA-based and memetic EDA-based approaches for fully automated web service composition. The two

EDA-based approaches, namely EDA-NHM and EDA-EHM, use permutation- and graph-based representations, respectively, with different distribution models and sampling techniques. EDA-NHM exploits a novel permutation-based representation, supporting reliable and accurate learning of the Node Histogram Matrix in the domain of fully automated service composition. EDA-EHM learns a suitable distribution in the form of the Edge Histogram Matrix by considering service dependencies based on the graph-based representations. We also propose a novel guided edge histogram-based backward graph sampling algorithm to effectively sample functionally valid candidate composite services with high comprehensive quality. The findings were that EDA-NHM outperforms EDA-EHM and the state-of-the-art approaches, such as a PSO-based approach [175], FL [47] and PathSearch [32], in finding near-optimal solutions. In addition, EDA-EHM achieves the highest efficiency among the competing EC-based approaches, delivering moderate effectiveness in finding high-quality composite services. Because of the apparent advantage of EDA-NHM over EDA-EHM in terms of the effectiveness, we further propose more effective memetic EDA-based approaches based on EDA-NHM. Particularly, one memetic EDA-based approach with the constrained layer-based one-point swap (called MEEDA-LOP) achieves significantly better effectiveness and efficiency, compared to a state-of-the-art memetic approach, such as MEFL [47], our proposed memetic EDA-based approaches that excludes MEEDA-LOP, and the baseline EDA-NHM.

This contribution has been published in:

(a) WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Comprehensive quality-aware automated semantic web service composition. In *AI 2017: Advances in Artificial Intelligence* (2017), Springer, pp. 195–207

(b) WANG, C., MA, H., CHEN, G., AND HARTMANN, S. GP-based approach to comprehensive quality-aware automated semantic web service composition. In *Simulated Evolution and Learning* (2017), Springer, pp. 170–183

(c) WANG, C., MA, H., AND CHEN, G. EDA-based approach to comprehensive quality-aware automated semantic web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2018), GECCO '18, ACM, pp. 147–148

(d) WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Knowledge-driven automated web service composition — an EDA-based approach. In *Web Information Systems Engineering – WISE 2018* (2018), Springer, pp. 135–150

(e) WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Towards fully automated semantic web service composition based on estimation of distribution algorithm. In *AI 2018: Advances in Artificial Intelligence* (2018), Springer, pp. 458–471

(f) WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Memetic EDA-based approaches to comprehensive quality-aware automated semantic web service composition. *IEEE Transactions on Services Computing* (Submitted)

2. This thesis studies two categories of multi-objective web service composition problem. The first one aims to find a set of approximated Pareto composite services in consideration of QoSM and QoS. The second one has the goal of simultaneously finding multiple solutions with optimized QoSM and QoS, each of which serves the request of one user segments with distinctive user preferences. For the first problem, we proposed a novel memetic NSGA-II with an EDA-based local search (called MNSGA2-EDA) to fully automated multi-objective semantic web service composition for explor-

ing trade-offs between QoSM and QoS. The novelty of this memetic method is: (1) to employ a clustering technique to select suitable candidate solutions from solutions evolved by NSGA-II for local search, and (2) to propose a EDA-based local search that constructs distribution models from the selected solutions and other good candidate solutions evolved by NSGA-II for sampling new solutions. Empirical comparisons between MNSGA2-EDA, NSGA-II and Hybrid and Hybrid-L show that MNSGA2-EDA achieves the highest effectiveness and efficiency in finding Pareto solutions. For the second problem, we formulate multiple service requests for diverse user segments with distinct QoSM preferences as a multitasking problem. We proposed two novel Permutation-based Multi-factorial Evolutionary Algorithms (called PMFEA and PMFEA-EDA) to solve this problem. The main novelty of PMFEA and PMFEA-EDA lies in that they either implicitly or explicitly learn and share the knowledge of good solutions evolved so far for different tasks. PMFEA, PMFEA-EDA, and two single-tasking approaches (i.e., EDA-NHM and FL [47]) are compared to each other. Our experiment showed that PMFEA-EDA is found to be more effective than PMFEA and other single-tasking methods, confirming that learning and sharing knowledge explicitly is superior to learning and sharing knowledge implicitly, and that PMFEA-EDA can achieve better effectiveness than single-tasking service composition approaches. We also found that both PMFEA-EDA and PMFEA are performed at the cost of only a fraction of time compared to the single-tasking approaches, which solve one task at a time.

This contribution has been published in:

(a) WANG, C., MA, H., CHEN, G., AND HARTMANN, S. A memetic NSGA-II with EDA-based local search for fully automated multiobjective web service composition. In *Proceedings of the Ge-*

netic and Evolutionary Computation Conference Companion (2019), GECCO '19, ACM, pp. 421–422

(b) WANG, C., MA, H., AND CHEN, G. Using EDA-based local search to improve the performance of NSGA-II for multiobjective semantic web service composition. In *Database and Expert Systems Applications* (2019), Springer, pp. 434–451

(c) WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Evolutionary multitasking for semantic web service composition. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), pp. 2490–2497

(d) WANG, C., MA, H., CHEN, A., HARTMANN, S., AND ONG, Y.-S. Using an Estimation of Distribution Algorithm to achieve multitasking semantic web service composition. *ACM Transactions on Evolutionary Learning and Optimization* (Major revision)

3. This thesis proposes a new dynamic service composition problem for specifically handling stochastic service failures. This problem aims to build robust composite services that serve as baseline composite services, coping with stochastic service failures. In particular, in the event of service failures, these baselines can continue to work reliably or be easily repaired with a negligible impact on the quality through fast local search. Afterwards, we propose a novel GA-based approach (called GA-MC) to solve this robust web service composition problem. Particularly, two essential techniques jointly form an effective method for searching robust composite services in GA. The first technique is to adopt the Monte Carlo sampling technique [153] to effectively approximate the robustness of any composite services. The second technique is to use an efficient re-optimization method (i.e., local search) to repair composite services in response to arbitrary service failures. The finding is that GA-MC can produce baseline composite services with significantly higher robust-

ness compared to FL [47], which is reported to achieve outstanding performance in finding near-optimal solutions without considering the robustness. Furthermore, we study a large-scale robust web service composition problem with a large service repository. We propose a new robustness approximation method based on a lower bound of the expected quality of QoS and QoSM over service failure scenarios. This is achieved by carefully selecting the scenarios based on service repositories. Furthermore, a two-stage GA-based approach (called GA-2Stage) is proposed with an adaptive evolution control to support two sequential evolutionary stages by using two different fitness evaluation methods. One GA-based approach that employs the lower bound robustness estimation throughout the generations (called GA-RE), GA-2Stage and GA-MC are compared to each other to determine the most suitable method for large-scale robust service composition. Our experiment comparisons reveal that GA-RE can outperform GA-MC in finding composite services with high robustness regardless of the size of the service repository. This indicates that the lower bound estimation is more effective and accurate than the Monte Carlo sampling for the robustness measures. Furthermore, compared to GA-RE, GA-2Stage achieves much better efficiency with a negligible impact on the effectiveness.

This contribution has been published in:

(a) WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Towards robust web service composition with stochastic service failures based on a genetic algorithm. In *AI 2019: Advances in Artificial Intelligence* (2019), Springer, pp. 445–459

(b) WANG, C., MA, H., CHEN, A., HARTMANN, S., AND BRANKE, J. Robustness estimation and optimisation for semantic web service composition with stochastic service failures. *IEEE Transactions on Emerging Topics in Computational Intelligence* (Major re-

vision)

1.5 Organization of Thesis

The remainder of the thesis is organized as follows:

Chapter 2 introduces some fundamental concepts related to web service and web service composition. Afterwards, we review some related works on automated service composition in single-objective, multi-objective, and dynamic contexts.

Chapter 3 proposes two EDA-based approaches and one memetic EDA-based approach with four different local search operators to solve the single-objective web service composition problem. The first approach is an EDA-based approach with permutation-based representation that explicitly learns the distributions of composite services in NHM. The second approach is an EDA-based approach with graph-based representation that explicitly learns the distributions of composite services in EHM. The last approach is a memetic EDA-based approach that extends one of the EDA-based approaches with different stochastic local search operators.

Chapter 4 proposes three approaches to solve two categories of multi-objective service composition problem. For one problem that refers to simultaneously optimizing multiple conflicting quality criteria, an NSGA-II with EDA-based local search is proposed with a novel way of making local improvements on Pareto solutions. For the other problem that refers to concurrently solving multiple service requests with distinctive preferences from user segments, two approaches are proposed based on two different ways of learning and sharing the knowledge of composite services for multiple service requests.

Chapter 5 proposes a novel dynamic web service composition problem that specifically handles stochastic service failures in a robust manner. Particularly, two important robustness estimation methods are proposed and employed in two novel GA-based approaches, which are proposed

for small-scale and large-scale dynamic web service composition, respectively.

Chapter 6 discusses the objectives achieved in this thesis, the main conclusions reached by our contribution chapters and insights that guide the future work.

Chapter 2

Background and Literature Review

This chapter presents some basic concepts of web service composition and related works in this field. We start by introducing some background knowledge in Sect. 2.1, including the functional and non-functional properties of both web service and web service composition, and an overview of both EC and AI planning techniques. Subsequently, in Sect. 2.2, we review some related works in web service composition, addressing the areas of interests in single-objective, multi-objective and dynamic contexts. Finally, Sect. 2.3 summarizes the reviews in these works and some limitations.

2.1 Background

2.1.1 Web Service

Web services are self-describing, self-contained software modules available over the internet [56]. As described in [56], web services are loosely coupled software modules, where service interfaces enable applications to work cooperatively regardless of the platforms, operation systems and

programming languages. Besides that, Web services can communicate with each other via internet protocols, e.g., HTTP, and are described by standard description languages, e.g., WSDL (Web Services Description Language) [99]. These description languages are mainly used to describe functional properties of Web services in terms of service inputs and service outputs, and provide mechanisms to users for searching desired services.

Web services are classified into two groups based on their functionalities: *information-providing services* and *world-altering services* [120]. The first type of services expects some data to be returned by giving inputs or nothing. For example, an air velocity transducer web service reads the wind speed and returns the velocity. This service does not require any inputs. In contrast, a city weather web service requires a city name and returns weather information of that city. Information-providing services do not produce any side effect on the world. The second type of services not only provide data information but also alters the status of the world by producing side effects. For example, a PayPal service may cause a reduction in the balance of users' bank account. Both types of Web services requires inputs and produces outputs, sometimes causing sides affects. As the majority of existing works does in the literature, *the functionalities of Web services are described as inputs and outputs*. Side effects are not considered because it does not affect how we composes services at the design stage. Therefore, we do not consider side effects in our thesis.

On the one hand, the functional attributes determine what the service does. On the other hand, the non-functional attributes often refer to some quality criteria, which are utilized to rank services [3]. For example, when several services provide the same functionality, users would prefer a service at a lower cost. Herein, we will demonstrate both the functional and non-functional properties in the following subsections.

Functional Properties of Web Services

The operational characteristics of web services are related to the functional properties, which demonstrate the behaviours of web services, i.e., what information is needed to invoke a service successfully and what information will be returned after their execution. In other words, a set of inputs I are required by a service and a set of outputs O are returned by a service. For *semantic web services*, one or more Ontologies are employed to describe the functional properties of web services semantically, enabling better interoperability (i.e., QoSM between inputs and outputs) between semantic web services, compared to those web services that are described syntactically. The functional properties of web services are demonstrated in Fig. 2.1.

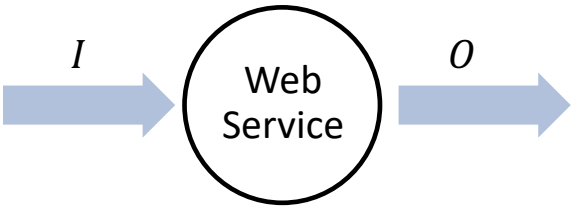


Fig. 2.1: Functional properties of a web service.

Non-functional Properties of Web Services

Apart from the functional properties of Web services discussed above, the non-functional properties of Web services refer to QoS, playing an important part in composing services. For example, customers always prefer a web service with the lowest execution cost and the highest response time and reliability. According to [226], the four most often considered QoS criteria are listed as follows:

- *Response time* (t) comprises of the execution time and waiting time [121], and it measures the expected delay in seconds between the

moment when a request is sent and the moment when the results are received.

- *Cost* (ct) is the amount of money that a service requester has to pay for executing a web service.
- *Reliability* (r) is the probability that a request is correctly responded within the maximum expected time, which often refers to the execution time.
- *Availability* (a) is the probability that a web service is operational and accessible when it is required for use.

2.1.2 Web Service Composition

Since an atomic web service may not satisfy users' complex functional requirements, web service composition composes existing web services collectively to produce a complex functionality to meet users' requirements. Composing service composition manually is very time-consuming and less productive. Therefore, many approaches have been developed to achieve semi-automated or fully automated service composition. The **semi-automated service composition** is inspired by business processes that require prior knowledge to build up abstract workflows [123]. These workflows consist of abstract services, each of which is defined by a pair of inputs and outputs. Therefore, semi-automated service composition focuses on service selections, which select a concrete atomic web service for each abstract service in the workflows. Fig. 2.2 shows five typical stages in the process of semi-automated service composition. The details of the service composition process are discussed as follows:

1. *Service request:* The first step is to collect users' requirements for a composition goal that comprises of the functional and non-functional requirements. This step is achieved by building up an abstract workflow, including a series of service discovery tasks for each abstract

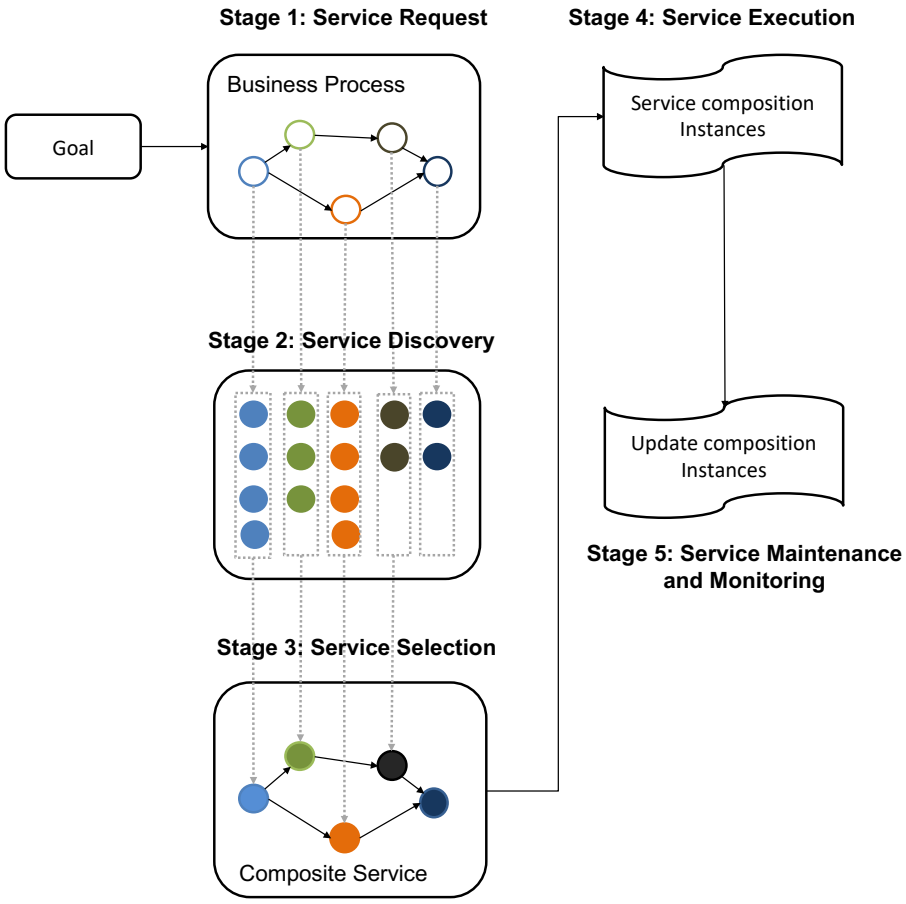


Fig. 2.2: Semi-automated web service composition process [123].

service with required functionalities. These tasks could be completed by selecting proper concrete services to reach the desired QoS. This *service request* is different from what we discussed in the fully automated service composition, which does not include an abstract workflow.

2. *Service discovery:* Once the goal is clearly specified, concrete Web services will be selected for each task regarding its functional requirement. Often, more than one concrete web service is likely to be found to match each service discovery task. Those matched Web services

are always different in QoS [89].

3. *Service selection:* At this stage, many techniques have been studied to select Web services to best match each service discovery task for the satisfaction of functional requirement and overall QoS of the business process (i.e., composite service). Therefore, a near-optimal composite service is created ahead of its execution.

4. *Service execution:* The composite service created at the service selection stage is executed as instances at the run time.

5. *Service maintenance and monitoring:* The composite service is monitored for any changes or services failures during its execution. In this stage, some actions, e.g., service reconfiguration [173], are taken for adapting these changes.

There is a distinction between semi-automated and fully automated approaches. In the context of semi-automated service composition, a complete workflow must be designed before the service selection. In addition, it could be impractical to design an optimal workflow manually. In **fully automated service composition**, a given workflow is not required in advance. Instead, AI planning algorithms (e.g., Graphplan algorithm [20]) can be utilized to achieve service composition, where service workflow is gradually built up along with the service discovery and service selection.

Fig. 2.3 shows a popular example of a composite service for a travelling domain. In this example, travelling agencies provide customers with a serial of services for booking flights, accommodations and buses, and generating tourist maps for the conference city. In Fig. 2.3, $TaskInput$ (i.e., $TravelDepartureDate$, $TravelReturnDate$, $HomeCity$ and $Conference\ City$) are gathered from customers, and $TaskOutput$ (i.e., $BusTicket$, $Flight\ Ticket$, $HotelReservation$ and $StreepMap$) are expected to be returned. The component services are gradually discovered and selected to construct a workflow from the $TaskInput$ node to the $TaskOutput$ node

if the inputs of the selected services are fulfilled with $TaskInput$ and any outputs of the previously selected services (i.e., predecessors). Therefore, the inputs of all the component services are satisfied. In this example, we begin by executing the FlightBooking service and GenerateMap service whose inputs can be immediately fulfilled by the task input, the Flight-Booking service books the flights and determines an $arrivalDate$. Then, using the $arrivalDate$ together with other provided information in the task input, we can book the hotel and bus, and generate a Map for the conference city. Consequently, these four services are properly chained together, producing $TaskOutput$ for customers.

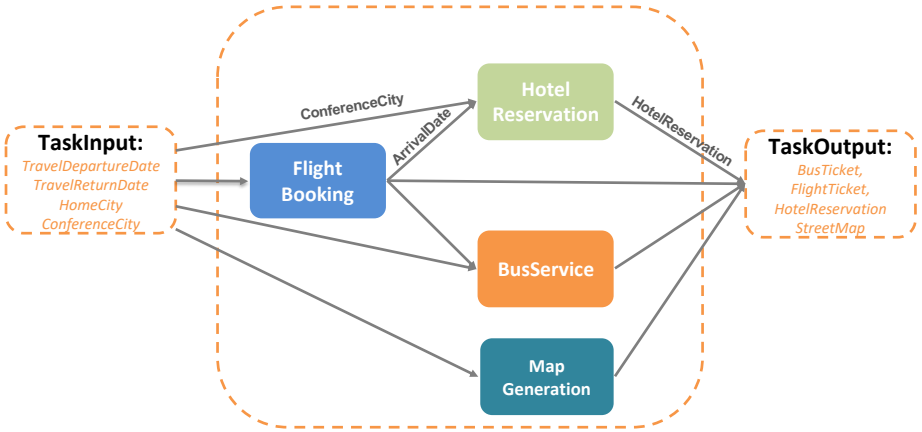


Fig. 2.3: An example of service composition for a travel agency.

In this thesis, we will concentrate on fully automated semantic web service composition, where service discovery and service selection are considered as interrelated tasks. The importance of pursuing this research direction is due to two reasons: (1) an abstract service composition workflow may be not known in advance, and (2) designing an optimal abstract workflow manually is fraught with difficulties when the number of services is large.

Functional Properties of Semantic Web Service Composition

We have demonstrated an example of a service composition in Fig. 2.3. In this example, two crucial characteristics are addressed for the functional validity of web service composition: every component service must be fulfilled for its inputs, and task output must be a subset of the outputs of the involved component Web services.

Fig. 2.4: An example of a component service for demonstrating QoSM.

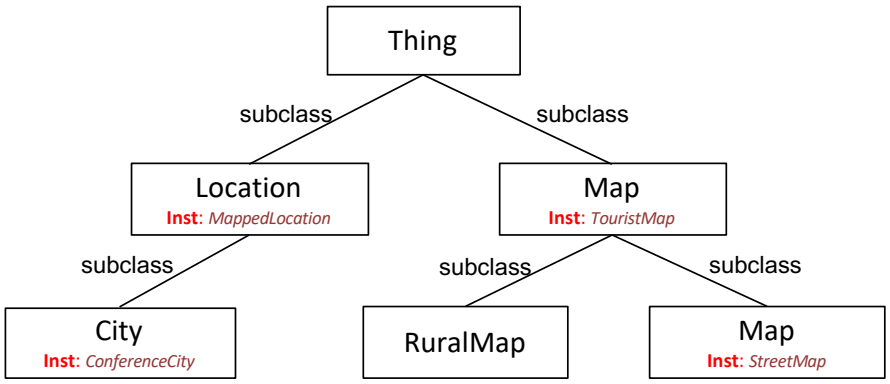


Fig. 2.5: Input and output-related concepts and instances described for MapGeneration service in Fig. 2.4.

Often, the inputs of a component service cannot be exactly matched by the provided inputs (that combines the inputs of composition task and the outputs produced by its predecessors). In Fig. 2.5, we consider a component service (i.e., MapGeneration service) from a composite service in Fig. 2.3. This component service requires inputs $MappedLocation$ and produces output $StreetMap$. These inputs and outputs are semantically described using an Ontology shown in Fig. 2.5. Particularly,

$ConferenceCity$ is an instance of $City$, $MappedLocation$ is an instance of $Location$, and $City \sqsubseteq Location$. Based on these semantic descriptions, $ConferenceCity$ does not exactly match $MappedLocation$. Therefore, QoSM between the given information $ConferenceCity$ and the required input $MappedLocation$ is relatively low.

Many works [16, 101, 102, 104, 148, 193] utilize description logic (DL) reasoning between input and output parameters of Web services for measuring QoSM, where four *matchmaking types* have been considered for measuring the QoSM. Given two concepts a, b in ontology $\mathcal{O}$, the four commonly utilized matchmaking types are used to describe the level of semantic matches [132]:

- *exact*: if a and b are equivalent (denoted as $a \equiv b$),
- *plugin*: if a is a sub-concept of b (denoted as $a \sqsubseteq b$),
- *subsume*: if a is a super-concept of b (denoted as $a \sqsupseteq b$),
- *fail*: if none of previous matchmaking types is returned.

Apart from the matchmaking types, other quality criteria have also been studied in the literature [102, 143, 160] for measuring QoSM. We will discuss these methods as follows:

For the first method, QoSM is measured by two quality criteria, i.e., matchmaking types and common description rate [102]. In addition to the four commonly used matchmaking types, matchmaking type $interaction$, denoted as $a \sqcap b$, is considered. In this work, QoSM is represented as a causal link $sl_{i,j} \doteq \langle S_i, Sim_T(Out\_s_i, In\_s_j), S_j \rangle$ that is created between the inputs of service S_i and outputs of S_j. In particular, both *exact* match and *plugin* match are presented as *robust causal links*, while both *subsume* match and $intersection$ match are presented as *valid casual links*. However, valid casual links are not specific enough to be utilized as the input of Web services. Thus, Extra Description, denoted as $In\_s_x \setminus Out\_s_y$, is required to

enable proper service composition using Eq. 2.1. By using Extra Description, *subsume* and $intersection$ are transferred to be *exact* and *plugin* respectively to formulate a robust link. Consequently, common description rate, $q_{cd}(sl_{i,j})$is calculated based on Extra Description and Least common subsume, denoted as $lcs(In\_s_x, Out\_s_y)$, using Eq. (2.2).

$$In\_s_x \setminus Out\_s_y \doteq \underset{\preceq d}{min}\{B | B \sqcap Out\_s_y \equiv In\_s_x\}, since\ Out\_s_y \sqsupseteq In\_s_x \quad (2.1)$$

$$q_{cd}(sl_{i,j}) = \frac{lcs(In\_s_x, Out\_s_y)}{In\_s_x \setminus Out\_s_y + lcs(In\_s_x, Out\_s_y)} \quad (2.2)$$

The second method for measuring QoSM utilizes a similarity measurement based on information retrieval. In [143], the similarity is calculated by an average value of similarity between two matched output $S_i.out_k$ and input $S_j.in_k$, denoted as $F\_Measure(S_i.out_k, S_j.in_k)$. It is calculated based on precision and recall between these two matched inputs and outputs.

The third method for measuring QoSM utilizes a semantic similarity measure proposed in [160]. The semantic similarity of two concepts a, b in $\mathcal{O}$, denoted as $sim(a, b)$, is calculated based on an edge counting method over an Ontology tree using Eq. (2.3). This method has the advantages of a simple calculation with good accuracy for measuring QoSM. In Eq. (2.3), N_a, N_b and N_c measure the distances from concept a, concept b, and their closest common ancestor c to the top concept of the ontology $\mathcal{O}$, respectively. L is the shortest distance between the two concepts, a and b, while D is the depth of the ontology tree. Also, λ is set to 1 for the neighbourhood concepts or 0 for the concepts from the same hierarchy.

$$sim(a, b) = \frac{2N_c \cdot e^{-\lambda L/D}}{N_a + N_b} \quad (2.3)$$

In this thesis, we are only interested in service composition, where only robust causal links (i.e., exact and plugin matches) are considered as most of existing works do [45, 72, 117, 41, 145, 221]. We suggest to consider the semantic similarity of concepts when comparing different plugin matches. As argued in [101],

plugin matches are less preferable than *exact* matches due to the overheads associated with data processing

Nonfunctional Properties of Web Service Composition

The non-functional properties of web service composition is determined by QoS of all the component Web services in the composed composite services. The aggregation value of QoS for Web services composition varies with respect to different constructs, which determines how services are associated with each other in a composite service [226].

- *Sequence construct*: Service composition executes each atomic service associated with a sequence construct in a sequential order. The aggregated time (T) and execution cost (CT) is computed as a sum of time and cost of Web services involved, respectively. The aggregated availability and reliability in a sequence construct are calculated by multiplying the availability and reliability of each component web service. This construct is shown in Fig. 2.6.

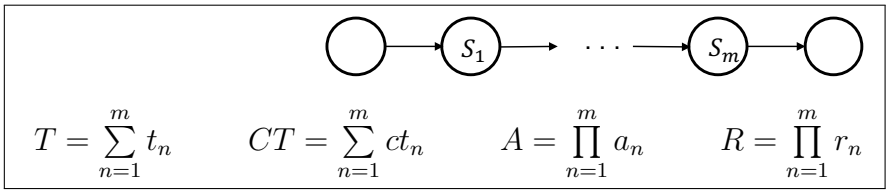


Fig. 2.6: Sequence construct and calculation of its QoS [221].

- *Parallel construct*: Web services in a parallel construct are executed concurrently. The aggregated execution cost, availability and reliability are calculated in the same way as those for the sequence construct, while the aggregated time (T) is determined by the most time-consuming path in the composite flow. This construct is presented in Fig. 2.7.

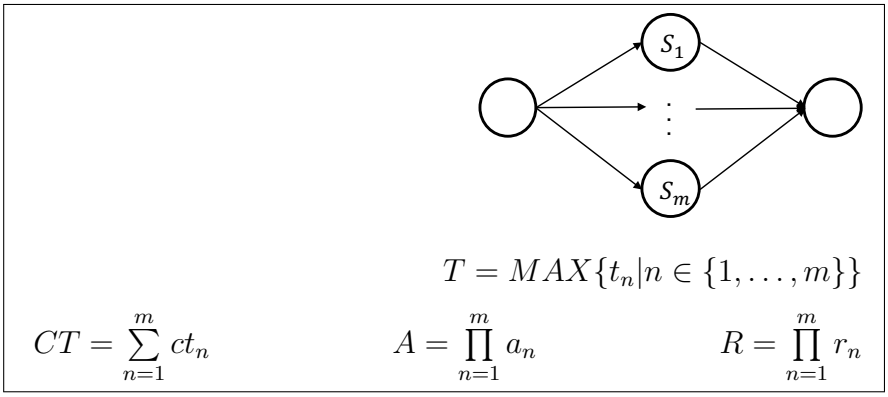


Fig. 2.7: Parallel construct and calculation of its QoS [221].

- *Choice construct*: Only one service path is executed in a choice construct depending on the satisfaction of the conditions on each path. In Fig. 2.8, assuming the choice construct has n branches, $p_1, \ldots, p_n$ with $\sum_{k=1}^{d} p_k = 1$ denote the probabilities of the different branches being selected. For example, the aggregated total cost CT is obtained by summing the multiplication of the total cost of each branching and the corresponding branch possibility, p, over all branches.

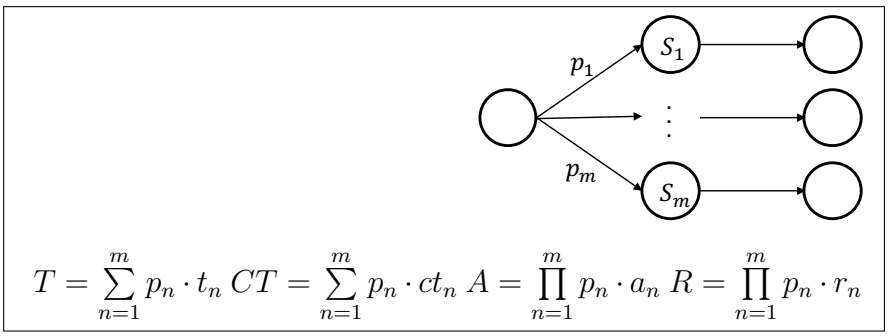


Fig. 2.8: Choice construct and calculation of its QoS [221].

- *Loop construct*: Web services composed with a loop construct are executed repeatedly until a certain condition is satisfied. In Fig. 2.9, assuming the average number of iterations is ℓ, and t, ct, r, and a are

corresponding aggregated value of a composite service. Therefore, for a loop construct, aggregated response time (T) and execution cost (CT) are $\ell \cdot t$ and $\ell \cdot ct$ respectively, while aggregated availability A and reliability R are the ℓ^{th} power of the value of one iteration, i.e., $A = a^{\ell}$ and $R = r^{\ell}$.

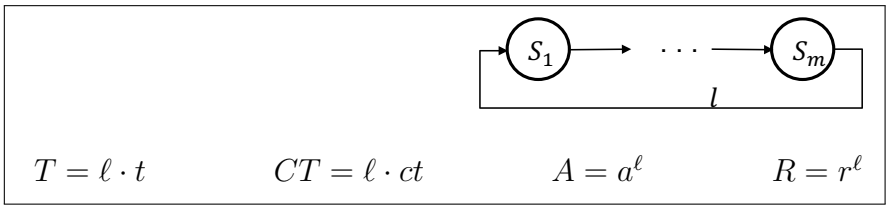


Fig. 2.9: Loop construct and calculation of its QoS [221].

In this thesis, we mainly focus on two constructors, sequence and parallel constructs, similar as in most automated service composition works [31, 32, 33, 41, 42, 45, 75, 85, 209, 117], where composite services can be represented as DAGs.

Web Service Discovery

To generate a composite service, mechanisms must be provided for service discovery as a fundamental technique in all service composition approaches. Agarwal et al. [2] summarize three mechanisms for semantic web service discovery: classification-based, functionality-based and hybrid mechanisms. Those service discovery mechanisms are further explained as below.

The classification-based mechanism makes use of classes provided by service semantic annotation using WSMO-Lite language [171]. Therefore, service requesters can use class names to express the desired web service, which is a straightforward method. However, classes without a clear semantic definition could lead to incomprehensibility issues. For example, several classes may be declared in either different terms for the same meaning (e.g., content and joy can be used for the same meaning of hap-

piness) or the same term is used for different meanings (e.g., content can be used for either information or happiness).

The functionality-based mechanism does not take classes into account but considers the functional properties of a web service. In particular, the desired functionality is defined by both inputs and outputs of web services. A discovery algorithm must be developed to handle matches of inputs and outputs that are described semantically under a given Ontology. The key idea of this matchmaking is to check whether services accept all the inputs provided by users and whether the desired outputs are delivered by services. One advantage of this technique is that it potentially meets the demands of all the comprehensible discovery, but lacks efficiency and scalability.

The hybrid mechanism combines classification and functionality-based mechanisms. Particularly, a classification hierarchy is proposed to achieve automatic semantic reasoning regarding the classes, inputs and outputs provided by different services. For example, a class can be associated with super-classes and sub-classes, where more general functionality and more specific functionality are presented, respectively. Meanwhile, to ensure consistency of the classification hierarchy, the inputs and outputs of a class must subsume the inputs and outputs of its sub-classes. This approach can benefit from both classification-based and functionality-based mechanisms. However, the classification hierarchy demands heavy maintenance work to remain consistent when a new web service is published or updated.

As discussed above, the classification-based and the hybrid mechanisms are considered to be less effective or require researchers to build up Ontologies for supporting the class hierarchy along with functional properties. Therefore. *We propose to use the functionality-based technique. That is, Ontologies are utilized to facilitate semantic matchmaking on the functional aspect of service approaches in our thesis.*

2.1.3 An Overview of EC Techniques

Based on the principles of Darwin natural selection, the natural evolution and selection of individuals in a population are virtualized and simulated in the EC techniques. In particular, a population of individuals are initialized for direct or indirect representation of the solutions. Those individuals are evolved and evaluated using a fitness function to evaluate the degree of how good (or bad) each individual is. Therefore, it is possible to find a solution with a near-optimal fitness value. EC techniques have been showing their promises in solving combinatorial optimization problems [13], achieving good flexibility for encoding many different problems, and good performances for efficiently finding good enough solutions.

EC techniques can be divided into two categories — conventional EC techniques and probabilistic model-building EC techniques [159]. Conventional EC techniques generate new candidate solutions using implicit knowledge via one or more variation operators on selected parent solutions. For example, GA produces new candidate solutions by using crossover operated on two selected parent individuals. Probabilistic model-building EC techniques use explicit distribution models to sample new promising solutions. For example, EDA uses explicit knowledge encoded by the distribution of a set of parent individuals, which often refers to a superior sub-population.

EC techniques have become increasingly popular in solving service composition problems in the contexts of single-objective, multi-objective, and dynamic contexts [158, 9, 125, 190]. For example, GA and PSO have shown to be particularly useful in the single-objective context [47, 175]. NSGA-II (**N**on-dominated **S**orting **G**enetic **A**lgorithm **II**) is very suitable for handling the first category of multi-objective service composition [46, 40], and MFEA(**M**ulti-**F**actorial **E**volutionary **A**lgorithm) can be very useful to solve the second category of multi-objective service composition. These EC techniques are discussed as follows:

Genetic Algorithm

GA evolves a population of solutions that are encoded as vectors via crossover, mutation and reproduction. One basic structure of GA is shown in ALGORITHM 1. It starts with initializing a population of candidate solutions, and the fitness of the candidate solutions are then evaluated. Afterwards, an iteration procedure begins until a stopping criterion is met. In each iteration, new solutions are produced by crossover, mutation and reproduction, forming a new population. These newly produced solutions are evaluated, and the best solution with the best fitness values is recorded and updated. Lastly, we return the solution with the best fitness value at the end of the iteration process.

ALGORITHM 1. Basic Structure of GA.

```
1: Population Initialization;
2: Evaluate each solution in the population;
3: while stopping criteria are not met do
4:     Populate new population through the use of genetic operators;
5:     Evaluate each solution in the newly formed population;
6: Return the solution with the best fitness;
```

One vector-based representation utilized in GA is a fixed-length binary vector. Such a vector can be initialized randomly. For example, the well-known 0-1 knapsack problem can be solved by GA with this vector-based representation, where each position of the vector is associated with a type of item that is restricted as zero or one (see an example in Fig 2.10). The evaluation of such a vector is often performed by checking the restrictions on positions of the vectors from left to right.

Crossover, mutations and reproduction can modify the vectors. As shown in Fig. 2.11a, two children are produced by exchanging two parts of two selected parents. In mutation, a new child can be produced by flipping a bit in one position of a selected parent, shown in Fig. 2.11b. The

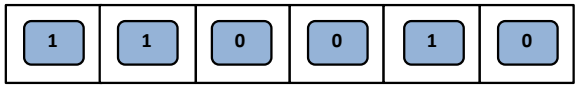
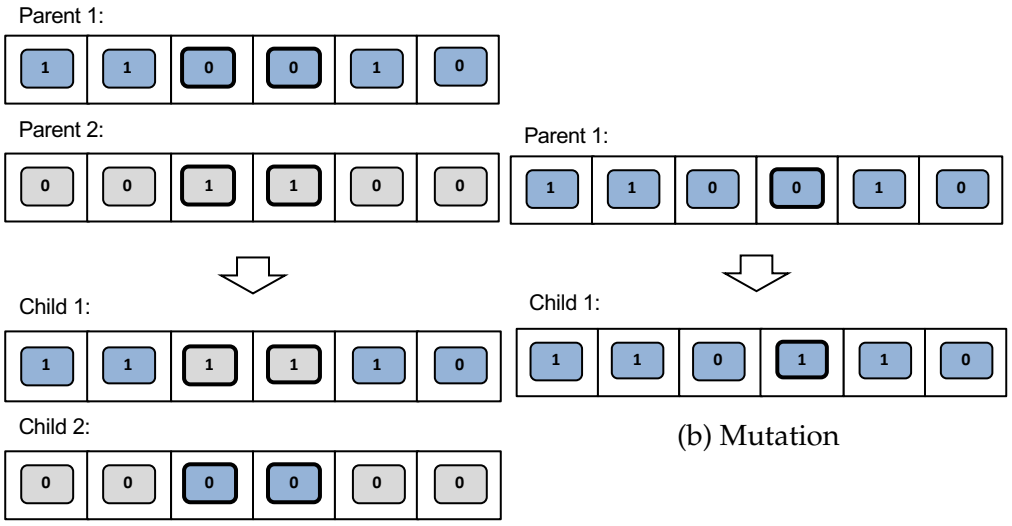


Fig. 2.10: An example of a vector-based representation in GA.



(a) Crossover

(b) Mutation

Fig. 2.11: Examples of crossover and mutation in GA.

reproduction operator simply copies a selected solution and put it into the next generation.

Memetic Algorithm

Memetic Algorithms (MAs), coined by Moscato [91] for the name of memetic, are a class of augmented global search heuristics that combine evolutionary algorithms with local search techniques for improving the quality of evolved solutions. These algorithms have shown its promise in solving a variety of problems, such as combinatorial optimization and multi-objective optimization problems.

Local search is a heuristic method that uses neighbourhood relations to iteratively examines a set of solutions, starting from one current solution,

and replace the current solution with a better neighbor if one exists. ALGORITHM 2 shows a basic structure of local search. It starts with a starting solution $best$. Afterwards, the iteration procedure will be carried out until a stopping criterion is met. This criterion is often defined as the number of steps that explores the neighbourhood via some move operator (i.e., local search operator). During the iteration procedure, a next neighbouring solution nb is produced by the application of the move operator. If the fitness of nb is better than the fitness of $best$, we set $best$ with nb. Consequently, $best$ is returned at the end of the iteration process.

ALGORITHM 2. Basic Structure of Local search.

```
1: Set a starting solution as best;
2: while stopping criteria are not met do
3:     Generate the next neighboring solution nb;
4:     if nb is better than best then
5:         set best with nb;
6: Return best;
```

In the design of memetic GA, practitioners typically choose to only apply local search once to the offspring. ALGORITHM 3 shows a basic structure of Memetic GA. Steps 1 to 5 follow its baseline GA in ALGORITHM 1 while Steps 4 to 8 are a local search procedure. From steps 4 to 8, a set of individuals, noted as Set, are selected from the newly formed population. Often two types of selection schemes (i.e., random selection scheme and statistical scheme) have been utilized for selecting suitable individuals for local search [34]. Afterwards, each of the selected individual undergoes local search with Lamarckian or Baldwinian scheme [73]. Lamarckian scheme uses the improved neighbouring individual to replace the original one in the population while Baldwinian scheme uses the fitness function of the neighbouring solution to evaluate the original solution. As discussed in [19], it is hard to decide which one is better for solving a par-

ticular problem, but it brings many possibilities to fine tune the memetic evolutionary algorithm.

ALGORITHM 3. Basic Structure of Memetic GA.

```
1: Population Initialization;
2: Evaluate each solution in the population;
3: while stopping criteria are not met do
4:     Populate new population through the use of genetic operators;
5:     Evaluate each solution in the newly formed population;
6:     Select a set of individuals, Set, from the new population;
7:     foreach individual in Set do
8:         Perform local search with Lamarckian or Baldwinian
           scheme.
9: Return the solution with the best fitness;
```

Estimation of Distribution Algorithm

EDA relies on an explicit probabilistic model that represents distributions learned from a set of promising candidate solutions. During the evolutionary process, the probabilistic model is adjusted in every generation with the aim to sample better candidate solutions. The basic structure of EDA is shown in ALGORITHM 4. It starts with a population of random candidate solutions. Afterwards, an iterative procedure begins and the procedure will not terminate until a stopping criterion is met. In each iteration, we evaluate each solution in the population and select a set of promising solutions among them. These selected solutions will be used to build a probability model, which is used to sample new candidate solutions. Consequently, a solution with the best fitness value is returned at the end of the iteration process. The key components of EDA are a class of probability models to capture the distribution of candidate solutions, a

procedure for learning this model and a procedure for sampling solutions from this model.

ALGORITHM 4. Basic Structure of EDA.

```
1: Initialize a population;
2: while stopping criteria are not met do
3:     Evaluate the population;
4:     Select a set of promising solutions from the population;
5:     Estimate the probability model;
6:     Sampling new candidate solutions;
7: Return the best solution;
```

The probabilistic models in EDA can be classified based on how they decompose the problems, for which graphical models can represent unconditional or conditional dependencies. Particularly, no dependencies, pairwise dependencies, multivariate dependencies and full dependence are four types of graphical models [139]. No dependencies models (e.g., a univariate model in Fig. 2.12a) assume that each problem variable is independent of each other. Different from no dependencies models, chain and forest models are examples of pairwise dependencies models (see Fig. 2.12b and Fig. 2.12c). For example, a forest model is a collection of trees, and each tree can be modelled based on the conditions on their parent variables except for the root. DAGs or undirect graphs are multivariate models (e.g., Bayesian network and Markov network in Fig. 2.12e and Fig. 2.12f). In addition, the marginal product model is a specific multivariate model, where problem variables fall into disjoint clusters (see Fig. 2.12d). *In our thesis, we suggest to use a univariate model because learning a reliable univariate model does not require a large number of instances, compared to other models*.

In EDA, candidate solutions can be represented by fixed-length strings, which can be used to learn some models discussed previously [139]. For

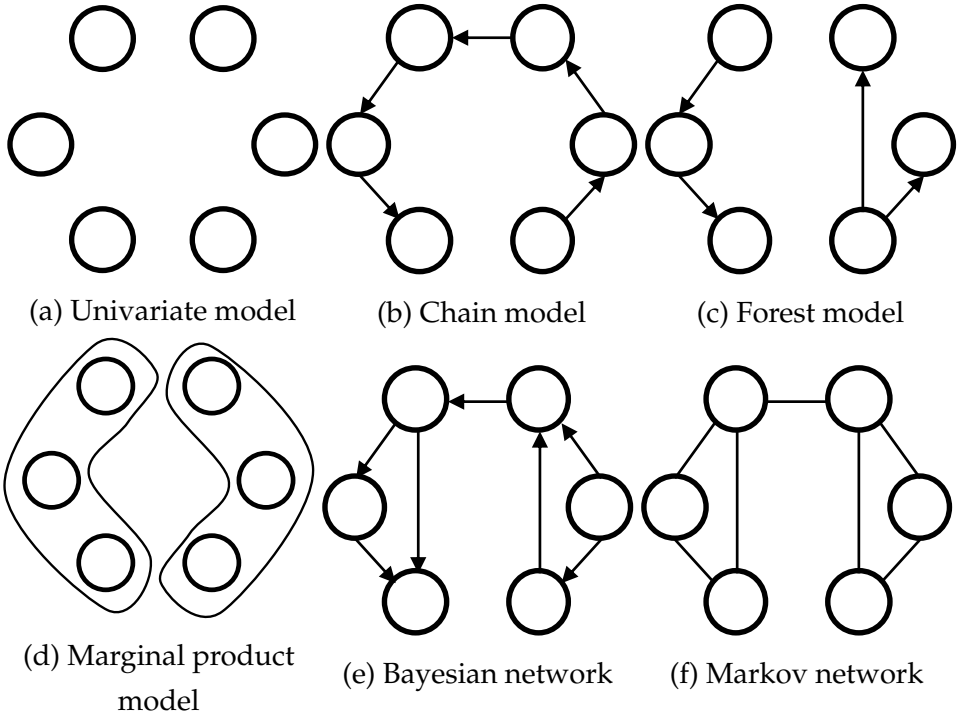


Fig. 2.12: Examples of probabilistic models in EDA [139].

example, a univariate model can be supported by a binary string [14]. Such a string can be generated by a probability vector $(p_1, p_2, \ldots, p_k, \ldots, p_n)$, where n is the size of bits in the string, and p_k denotes the probability that a 1 appears in the k^{th} position. A chain model can also be supported by the binary string [50]. For example, a chain distribution can be learnt by: (1) position orders in the string, (2) a probability that a 1 appears in the first position and (3) conditional probabilities of other positions based on their predecessors.

Other representations that extend the fixed-length strings are real-valued vectors and permutations. Particularly, permutations can be naturally used to present the candidate solution in many problems, such as scheduling or resource allocation problems. EDA often utilizes permutations to capture two types of information: the first one is the absolute po-

sition of variables and the second one is the relative ordering of variables. In the quadratic assignment problem, these two types of information are equally important to learn. Unlike permutations, real-valued vectors are not suitable for dealing with the discrete problem directly, where a mapping is required to interpret real-valued representations to a discrete one. [138].

Non-dominated Sorting Genetic Algorithm II

NSGA-II was introduced in [54], and it is one of the most popular algorithm for solving Multi-objective optimization problem (MOP), which can be defined as:

$$\begin{aligned} \text{Minimize } \vec{f}(\vec{x}) &= [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \\ \text{subject to } C &\in \mathcal{Z} \end{aligned} \tag{2.4}$$

$\vec{f}(\vec{x})$ is a vector function and $\mathcal{Z}$ is a feasible solution space.

The key idea of NSGA-II is the fast non-dominated sorting strategy for sorting solutions with respect to multiple objectives. This strategy includes two important steps: The first step is to rank the candidate solutions from the mating pool (that combines current population and the promising candidates from the last generation) based on the concept of *Pareto dominance*. *Pareto dominance* can be defined as: a vector $\vec{x}$ dominates $\vec{x}'$ (denoted by $\vec{x} \prec \vec{x}'$) if $f_i(\vec{x}) \leq f_i(\vec{x}')$ for all the i functions involved in $\vec{f}(\vec{x})$ and there is at least one i such that $f_i(\vec{x}) < f_i(\vec{x}')$. Therefore, a vector $\vec{x}^*$ is Pareto optimal if not exists a vector $\vec{x}' \in \Omega$ such that $\vec{x}' \prec \vec{x}^*$. This Pareto dominance strategy in NSGA-II allows us to find a set of non-dominated vectors, forming the *Pareto front* (also called the first front), which can be defined as: a Pareto optimal set $PF^* = \{\vec{x}^* \in \Omega\}$, where $\vec{x}^*$ is Pareto optimal. Following this idea, the second front can be formed based on the remaining individuals, and its rank is lower than the first

front. Motivated by this strategy, we can rank all the candidates according to the ranks associated with the fronts.

The second step is to use crowding distance [54] for sorting the candidates if they have the same rank. The crowding distance is an indicator for measuring the diversity of candidate solutions in the front. It can be calculated as the average side length of the cuboid, shown in Fig. 2.13. In this example, a cuboid is formed based on the nearest neighbours of the i^{th} solution on the Pareto front. By using crowding distance, a candidate solution with a larger crowding distance is more likely to be kept.

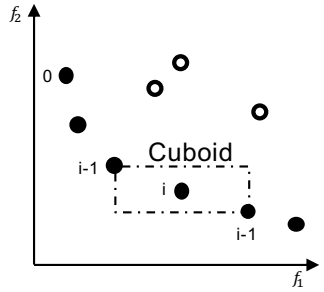


Fig. 2.13: Crowding distance calculation based on points marked with filled circles [54].

Fig. 2.14 demonstrates a process of the non-dominated sorting strategy in NSGA-II. Particularly, n promising solutions are selected from a mating pool of $2n$ to form population P^{t+1} using non-dominated sorting and crowding distance sorting.

ALGORITHM 5 shows the basic structure of NSGA-II, starting with initializing a population of n candidate solutions. These solutions are evaluated regarding every objective. Afterwards, the iteration part will be repeated until a stopping criterion is met. During the iteration, n promising solutions are selected from the current population based on the fast non-dominated sorting strategy discussed previously. We apply the genetic operator to produce new offspring based on the n promising solutions. These newly produced solutions are combined with the n promising so-

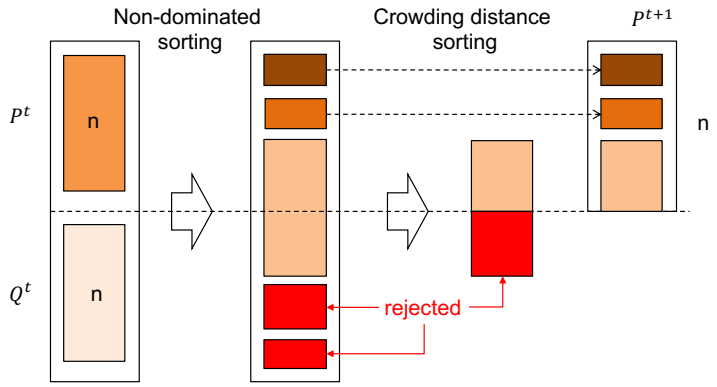


Fig. 2.14: Fast non-dominated sorting strategy in NSGA-II [54].

ALGORITHM 5. Basic Structure of NSGA-II.

1: Initialize a population of n candidate solutions;
2: Evaluate each solution with respect to each objective;
3: **while** *stopping criteria are not met* **do**
4: Choose n solutions over the population using the non-dominated sorting strategy;
5: Populate new offspring through the use of genetic operators ;
6: Form a mating pool by combining new offspring and the n solutions;
7: Evaluate each solution in the mating pool;
8: Return solution set with highest rank in the last generation;

lutions to form an updated mating pool of size $2n$. After the iteration, a set of solutions with the highest rank is returned from the last generation. The representation and genetic operators in NSGA-II is similar to those of GA. Therefore, we do not discuss them again.

Multi-Factorial Evolutionary Algorithm

Different from single-tasking EC techniques, MFEA is a new evolutionary paradigm, optimizing K optimization tasks concurrently, where each task

contributes a factor that affects the evolution of a single population. In other words, searching for solutions for one task may contribute to problem solving on related tasks. In MFEA, a unified representation allows a unified search space made of the search spaces of the K tasks. This unified representation can be decoded into solutions for each individual task. The following definitions are also defined in [71], which capture the key attributes associated with each individual Π in MFEA. For simplicity, we assume all the tasks are maximization problems.

Definition 1: The *factorial cost* f_j^{Π} of individual Π measures the fitness value with respect to the K tasks, where $j \in \{1, 2, \ldots, K\}$.

Definition 2: The *factorial rank* r_j^{Π} of individual Π on task T_j, where $j \in \{1, 2, \ldots, K\}$, is the index of Π in the population sorted in descending order according to their factorial cost with respect to task T_j.

Definition 3: The *scalar fitness* φ^{Π} of individual Π is calculated based on its best factorial rank over the K tasks, which is given by $\varphi^{\Pi} = {}^{1}/_{min_{j\in\{1,2,\ldots,K\}} r_j^{\Pi}}$.

Definition 4: The *skill factor* of individual Π denotes the most effective task among the K tasks, and is given by $\tau^{\Pi} = argmin_j\{r_j^{\Pi}\}$, where $j \in \{1, 2, \ldots, K\}$.

Based on the scalar fitness, evolved solutions in a population can be compared across the K tasks. In particular, an individual associated with a higher scalar fitness is considered to be better. Therefore, *multifactorial optimality* is defined as below:

Definition 5: An individual $\Pi^{\star}$ associated with factorial cost $\{f_1^{\star}, f_2^{\star}, \ldots, f_K^{\star}\}$ is optimal iff $\exists j \in \{1, 2, \ldots, K\}$ such that $f_j^{\star} \geq f_j^{\Pi}$, where Π denotes any solution on task T_j.

The basic structure of MFEA is outlined in ALGORITHM 6. It starts with generating a population of candidate solutions, and each of them is evaluated for its factorial cost regarding K tasks. After computing the factorial cost, we can compute factorial rank, scalar fitness, and skill factor of candidate solutions. Subsequently, the iteration process will be carried out until

ALGORITHM 6. Basic Structure of MFEA.

```
Initialize a population of candidate solutions;
Evaluate each candidate solutions for its factorial cost with respect
  to K tasks;
Compute factorial rank, scalar fitness, and skill factor of each
  candidate solution;
while stopping criteria are not met do
    Populate offspring using assortative mating, see
      ALGORITHM 7;
    Evaluated each solution for its factorial cost based on the
      selected task, see ALGORITHM 8;
    Combine the offspring and current population;
    Update factorial rank, scalar fitness, and skill factor of the
      combined candidates;
    Form next population by selecting fittest candidates from the
      combined candidates;
Return the best solutions for K tasks;
```

stopping criteria are met. During the iteration, new offspring are produced by assortative mating for breeding offspring for K tasks. In particular, crossover and mutation operators in assortative mating will be employed, see details in ALGORITHM 7. Once a new offspring is generated, each candidate solution in the new offspring is only evaluated for its factorial cost based on the selected task. The selected task is determined by vertical cultural transmission using ALGORITHM 8. We then combine the current population and assortative mating offspring population. These combined candidate solutions are evaluated for factorial rank, scalar fitness, and skill factor. Consequently, we form the next population by selecting the fittest candidates from the combined candidates. When the stopping criteria are met, we return the best candidate solution for each task. The key compo-

nents of PMFEA are assortative mating and vertical cultural transmission, which are further discussed as follows:

The procedure of assortative mating for breeding offspring for K tasks is outlined in ALGORITHM 7. In particular, two randomly selected parent candidates undergo crossover if they have the same skill factors. Otherwise, a randomly generated probability, $rand$, is used to balance exploitation and exploration across tasks. That is, crossover is performed over the parent candidates with different skill factors, or mutation is performed on each parent.

ALGORITHM 7. Assortative Mating [71].

Randomly select two parents Π_a^g and Π_b^g from $\mathcal{P}^g$;
$rand \leftarrow Rand(0, 1)$;
if $\tau^{\Pi_a^g} = \tau^{\Pi_b^g}$ *or* $rand < rmp$ **then**
    Perform crossover on Π_a^g and Π_b^g to generate two children Π_c^g and Π_d^g;
else
    Perform mutation on Π_a^g to generate one child Π_e^g;
    Perform mutation on Π_b^g to generate one child Π_f^g;

Vertical cultural transmission is proposed to allow the offspring to imitate the skill factor of any one of their parents. As illustrated in ALGORITHM 8, any child produced by assortative mating is only evaluated on one selected task based on the skill factors of its parents.

2.1.4 An Overview of AI Planning

AI planning is a branch of AI that concerns action sequences to be executed for reaching the desired outcome. In general, three key components are often described in a planning problem [137]: a description of the initial state for the environment, a description of the possible actions that can be

ALGORITHM 8. Vertical Cultural Transmission [71].

if Π_k^g *is produced by two parents* Π_a^g *and* Π_b^g **then**
  Generate a random $rand$ between 0 and 1;
  if $rand < 0.5$ **then**
    Π_k^g imitates the skill factor $\tau^{\Pi_a^g}$ of Π_a^g;
    Π_k^g is only evaluated on task $T_{\tau^{\Pi_a^g}}$;
  else
    Π_k^g imitates the skill factor $\tau^{\Pi_b^g}$ of Π_b^g;
    Π_k^g is only evaluated on task $T_{\tau^{\Pi_b^g}}$;
else
  Let Π_e^g be the only one parent of Π_k^g;
  Π_k^g imitates the skill factor $\tau^{\Pi_e^g}$ of Π_e^g;
  Π_k^g is only evaluated on task $T_{\tau^{\Pi_e^g}}$;

executed in the environment, and a description of the desired goal that specifies the desired state of the environment.

Existing works on AI planning can be classified into four groups: *state-space based planning, graph-based planning, partial order refinement planning*, and *satisfiability and logic programming based planning*[137]. The *state-space based planning* aims to solve a planning problem by searching a set of possible states that achieves the desired goal. This search can be conducted in both a forward or backward way. Particularly, the forward state search starts with the initial state, towards the state that satisfies the desired goal, while the backward state search begins with a state that satisfies the desired goal, towards the initial state. The *graph-based planning* is to construct a directed levelled graph, where three typical levels, i.e., proposition level, action level and another proposition level, are associated with the graph. Particularly, the first proposition level is used to decide the satisfaction of the precondition of the actions in the second level, and the third level updates the proposition by considering the effects caused by the actions

in the second level. The *Partial order refinement planning* is different from the previous two planning techniques. The major differences lie in that a partially ordered plan is produced, resulting in different linearizations that achieve the desired goal. The *satisfiability and logic programming based planning* transfer planning problem as a reasoning problem.

In the domain of web service composition, a graph building technique based on state-space based planning, e.g., Graphplan [20, 155], have shown to be very useful for constructing composite services automatically [40, 47, 174, 192, 200] . The basic structure of Graphplan is shown in ALGORITHM 9.

ALGORITHM 9. Basic structure for Graphplan [20, 155].

Input : an initial state, a set of actions, a goal state
Output: a sequence of actions
1: Initial a planning graph with one level that contains an initial state;
2: **while** *goal state is not reached* **do**
3: Identify actions that are satisfied by states in current level;
4: Append the graph with a new level that contains states reached by the actions;
5: **return** a sequence of actions extracted from the initial state to the goal state;

The key idea of Graphplan is to build up a graph with levels, whose nodes represent states and edges represent actions. From the graph, a sequence of actions, i.e., a composite service, is expected to be identified by extracting actions from the graph. Graphplan takes three inputs: an initial state, a set of actions, and a goal state. It starts with constructing a graph with one level, where only the start state is contained. As long as the goal state is not reached, we expand the structure of the graph by adding a new level, which consists of new states that are caused by pos-

sible actions taken from the previous state. Consequently, one solution is extracted from the graph once the goal state is reached. Such a solution contains a set of actions represented by a set of edges, starting from the initial state to the goal state. *In our thesis, we will keep utilizing this Graphplan technique because it has shown its promise in building composite services in the literature*.

2.2 Literature Review

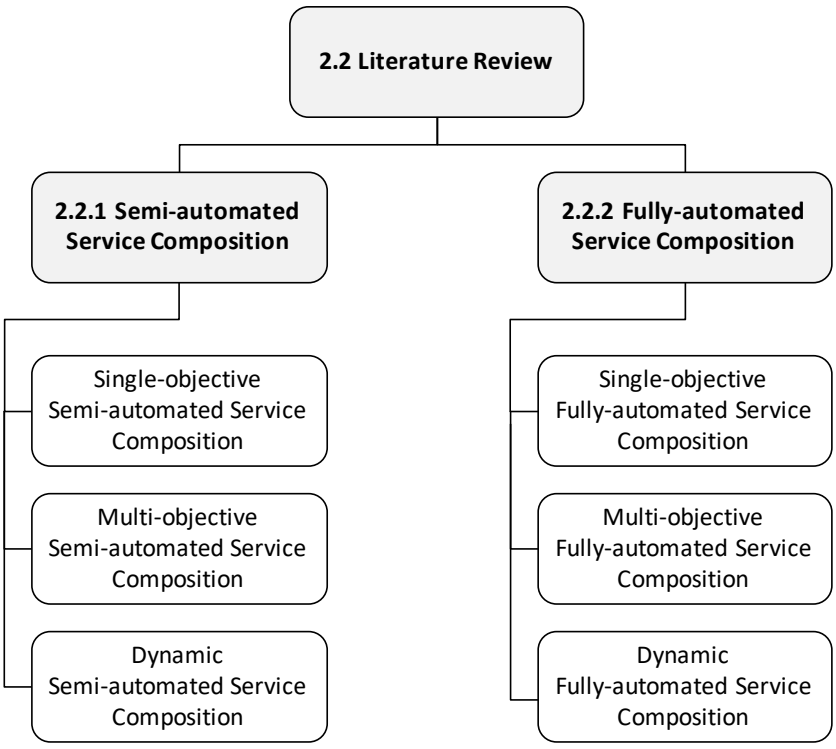


Fig. 2.15: An overview of the literature review.

As discussed in Sect. 2.1.2, web service composition is performed using two strategies: this first one assumes a pre-defined service composition workflow is known, and it consists of abstract service slots that specify the required functionalities for atomic web services; the second one does not

strictly follow any workflow, and it constructs workflows simultaneously with atomic service selections. These two strategies result in two groups of works: semi-automated web service composition and fully web service composition, respectively. In this section, we review some recent works in these two groups in the context of single-objective, multi-objective and dynamic optimizations. Fig. 2.15 shows a diagram to guide our discussion on the related works for the literature review. These works are also summarized in Table 2.1 based on 91 recent papers.

Category	Semi automated	Fully automated
Single-objective	**Genetic algorithm** [26, 27, 28, 58, 64, 101, 167, 188, 191, 198] **Particle swarm optimisation** [7, 76, 82, 112, 127, 196, 194, 207, 199, 231] **Clonal selection algorithm** [143, 208] **Ant colony optimisation** [35, 37, 218] **Articial bee Colony** [11, 206] **Differential evolution algorithm** [144] **Cukoo search** [24, 36, 66] **Integer linear programming** [62, 63, 67, 216] **Dynamic programming** [77, 204]	**Genetic Programming** [42, 117, 126, 152, 182, 202, 221, 222] **Graph-based Genetic Programming** [41, 43] **Genetic algorithm** [44, 47, 156] **Particle swarm optimisation** [47, 45, 175] **Graph search** [31, 32, 33, 57, 75, 85, 209]
Multi-objective	**The First Category:** **Multi-objective GA** [113, 172] **Multi-objective PSO** [110, 150, 214, 228] **Multi-objective ACO** [187, 220, 230] **Many-objective NSGA-II** [51] **The Second Category:** **Multi-factorial Evolutionary Algorithm** [17]	**The First Category:** **NSGA-II** [46] **Hybrid** [40] **The Second Category:** No related works
Dynamic	**Bounded-interval** [9, 125, 190] **Predicted approaches** [6, 30, 80, 108] **Service reconfiguration approaches** [23, 111, 124] **Robust approaches** [65, 173]	No related works
No optimization	**AI planning** [77, 146, 174, 192] **ER model-based approach** [205]	

Table 2.1: Summary of web service composition approaches.

2.2.1 Semi-Automated Web Service Composition

Single-Objective Semi-Automated Approaches

The majority of the existing works treat service composition problems as single-objective optimization problems. The single objective is handled based on users' concerns in different ways: the first one is to optimize the number of component services in a composite service [12, 126, 152, 174]; the second one is to optimize QoS based a combined score [24, 117, 202, 221, 222], e.g., a weighted QoS score based on a simple additive weighting (SAW) technique [79]; the third one is to jointly optimize QoS and QoSM [36, 58, 101]. These different objectives are optimized using many different algorithms, which will be discussed separately as follows:

GA is an effective searching technique for solving combinatorial optimization problems [163], and it has been popularly applied to single-objective semi-automated web service composition with vector-based representations [26, 27, 28, 58, 64, 101, 167, 188, 191, 198]. In general, vectors utilized in GA represent composite services, and each element of vectors corresponds to one abstract service slot of a given composition workflow. Genetic operator, such as crossover and mutation, are performed on vectors to generate new vector-based candidate solutions. For example, [26] proposes a GA-based approach with an objective to optimize a combined QoS score. They experimentally find out that when the number of service candidates increases, their GA-based approach is preferable to a Integer Linear Programming based approach for finding high-quality composite services. To enhance the performance of GA, the **hybridization of GA and other techniques**, such as local search, are also explored. For example, [167] proposes a hybrid approach utilizing GA and local search. In particular, a local optimizer is performed at the beginning of GA, and it aims to improve the quality of candidate solutions that are randomly generated in the initial population. This GA method with local optimizer contributes to better performance compared to GA without the local optimizer. The

success of GA is due to the use of the vector-based representation, which is a straightforward way of presenting composite services that strictly follow a fixed composition workflow. However, such representation can not directly cope with composite services over varied workflows in the fully automated service composition.

PSO has been widely used as a service selection technique for solving single-objective semi-automated web service composition problem [7, 76, 82, 112, 196, 194, 207, 199, 231]. Compared to GA, a different vector-based presentation is utilized in PSO as particles' positions, and each dimension of the particle corresponds to one abstract service slot of a given composition workflow. To improve the efficiency of PSO, a recent work [207] proposes a chaotic PSO algorithm based on the predatory search strategy for semi-automated service composition, where a cotangent sequence with chaotic characteristics is introduced into PSO, instead of a random searching strategy. The **hybridization of PSO and other techniques**, such as GA, are also explored. For example, Liu et al. [112] propose a hybrid Genetic PSO method for handling semi-automated service composition. Particularly, PSO is hybridized with GA for the purpose of local exploitation via crossover operators. Besides that, the number of generations in both PSO and GA is properly balanced to reach a good exploration and exploitation. Another recent work [127] investigates the use of an agent-based framework to compose web services by identifying the QoS parameters, and employs PSO for service selection over abstract services of composition workflows. Although PSO is originally designed for effectively tackling continuous optimization problems [114], proper modifications are made in these PSO-based approaches for solving the discrete optimization problem on service composition.

Clonal Selection Algorithm (CSA) is an Artificial Immune System (AIS) technique for solving optimization problems. The principle of CSA lies in the features of immune memory, affinity maturation. In particular, the antigen is considered as a fitness function. A proportion of antibody,

rather than the best affinity antibody, are chosen to proliferate. Furthermore, the speed and accuracy of the immune responses grow higher and higher after each infection, even confronting cross-reactive response. Apart from that, hypermutation and receptor editing contribute to avoiding local optimization and selecting optimized solution respectively. CSA has also been proposed for solving semi-automated service composition [143, 208]. For example, Yan et al. [208] encoded composite service using a binary string as an antibody for evaluating the affinity value of the antigen (i.e., fitness function), and the antibody with low concentration will be selected for crossover and mutation. The **hybridization of CSA and other techniques**, such as AI planning, are also explored. For example, Pop et al. [143] combined an enhancing planning graph (EPG) and CSA to handle single-objective semi-automated web service composition. The EPG model is constructed with actions and layers involved in multiple stages, where each action represents clustered Web services, and each layer represents inputs/outputs. During the clonal selection process, the antigen is represented as a fitness function, and the antibody is represented as a binary alphabet to encode EPG. Despite some success in CSA, the speed of convergence often suffers due to the high diversity of the population preserved by the cloning of antibodies [158].

Ant Colony Optimisation (ACO) is utilized to tackle semi-automated service composition [35, 37, 218]. The idea of ACO-based approaches is to perform a path construction from a planning graph, e.g., two-layered graph. This planning graph is modelled based on an abstract composition workflow and a set of candidate solutions for each abstract service slot. The goal of these approaches is to find an optimized path by traversing on the planning graph using a group of ants. In every iteration of ACO, every ant constructs a path by selecting edges of the graph according to the strength of the pheromone left by other ants. This pheromone is associated with every edge of the planning graph. Once an ant constructs a solution, the pheromone is updated along with the evaporation

strategy on the edges. For example, [37] proposes an ant-inspired selection method based on an EPG, which extends the classical planning graph [154] with clusters of services and service inputs/outputs. This method is tested over a few instances of EPGs with different complexities. ACO is a very intuitive way of demonstrating the optimization process of a service composition problem. However, ACO can consume a large memory when a complex planning graph is constructed.

Other EC methods have also been investigated to tackle web service composition problems. For example, **Artificial Bee Colony (ABC)** is an optimization technique that simulates the foraging behaviour of honey bees, and has been used for service selections in single-objective semi-automated web service composition [11, 206]. For example, a recent work [11] proposes an Integrated Probability Multi-search and Solution Acceptance Rule-based Artificial Bee Colony Optimization Scheme, named IPM-SAR-ABCOS, to optimize transaction and QoS characteristics. The rule in IPM-SAR-ABCOS can enhance the rate of diversification and intensification in the employee bee phase and the onlooker bee respectively, preventing from being easily trapped into the local optimal. Unlike GA, **Differential Evolution algorithm (DE)** is designed for solving optimization problems with variables in continuous domains. It has been utilized to solve a discrete service composition problem with the help of a genome encoding [144]. **Cuckoo search** is introduced in [211], which is inspired based on cuckoo species which lay their eggs in the nests of other host birds. For example, cuckoo search has been adopted for solving QoS-aware semantic service composition problem over a searching space represented by an EPG [36]. To further consider the distributed network environment, a cuckoo search based approach [66] is proposed to find composite services with optimized QoS and network QoS over geo-distributed cloud environments.

Other approaches (without utilizing EC techniques) do not rely on bio-inspired techniques. They target optimal composite services by some

other methods, such as Integer Linear Programming (ILP), dynamic programming, and local search. **Integer Linear Programming** is used to achieve single-objective semi-automated web service composition. Generally, an ILP model is created with three inputs: a set of decision variables, an objective function and a set of constraints. The outputs of ILP are decision variables and values of maximized/minimized objective function. ILP is flexible for handling QoS constraints and optimizing problems for semi-automated QoS-aware service composition [62, 63, 67, 216]. For example, Gao et al. [63] define a zero-one ILP model for web service composition based on an abstract service workflow, where services with the same functionality but different QoS are classified into the same class and are selected for each abstract service slot. Yoo et al. [216] formulate the web service composition problem based on a zero-one ILP model introduced in [63]. A very recent work [67] proposes a LP-based approach, named "LP-WSC", to QoS-aware web service composition in a geographically distributed cloud environment. These works take both QoS and constraints on QoS into account. However, due to an increase in the number of decision variables, ILP may lead to exponentially increased complexity and cost in computation [107]. Besides that, QoS of composite services in ILP-based approaches is calculated by summing up the individual QoS score of every component services. Such a QoS calculation is not always appropriate [158]. For example, the availability of composite services should be calculated by multiplying the availability of every component service.

Dynamic Programming Approach is an effective method for solving optimization problems that can be broken into subproblems, avoiding solving them repeatedly. This technique is also utilized for handling single-objective semi-automated service composition [77, 204]. For example, in [77], an efficient pruning approach is developed combing three techniques — a forward filtering algorithm for searching task-related services, a modified dynamic programming approach for handling a sub-

problem (i.e., a problem on the satisfaction of each concept pool of every graph layer), and a backward-search method for searching optimal composite services. Xu et al. [204] worked on large-scale service composition problem with a guarantee in QoS, where a dynamic programming algorithm is developed to optimize every subproblem (i.e., a service path). To derive an execution plan, a depth-first search is utilized to trace back paths with the aim to optimize QoS.

Multi-Objective Semi-Automated Approaches

To solve **the first category of multi-objective service composition** discussed in Sect. 1.2, many evolutionary multi-objective techniques [35, 110, 150, 113, 172, 187, 201, 214, 217, 220, 230] have been developed based on the idea of ranking a set of optimized solutions based on the dominance relationship. The aim of the first category multi-objective techniques is to find a set of non-dominant solutions, i.e., *Pareto Front* (also called *Pareto solutions*).

Multi-objective GA has been widely used for the semi-automated web service composition [113, 172]. For example, Liu et al. [113] propose a service composition model, i.e., multi-constraint and multi-objective optimal path, where only the sequence composition construct is supported. In their work, different paths, i.e., composite services, are searched by GA. Wada et al. [172] investigate a semi-automated approach with a vector-based representation. Each vector presents three composite services for three user groups. Two multi-objective GAs (called E^3-MOGA and X-E^3) are proposed in this work. Particularly, E^3-MOGA is designed to search for equally distributed Pareto-optimal solutions in the multi-objective space, while X-E^3 is designed to search for Pareto-optimal solutions that can reveal the maximum range of trade-offs, covering extreme solutions in the search space.

Multi-objective PSO is another interesting technique that handles semi-automated service composition [110, 150, 214]. For example, Yin et

al. [214] combines genetic operators and a PSO optimization algorithm together to tackle multi-objective semi-automated web service composition. In particular, the updates of each particle's velocity and position are achieved by the introduced crossover operator. Furthermore, a mutation strategy is introduced to increase the diversity of the particles, and it is performed on the *gbest* particle when the diversity of the proposed swarm indicator is below a certain value. A recent work [228] studies the service level agreement aware service composition problem, where resource sharing property is considered to decrease the cost for service deployment. To find trade-off composite services that considers the cost and other QoS attributes in this problem, a multi-objective PSO is employed with a novel particle position form. This form is proposed to cope with the multi-level demands from the resource sharing.

Multi-objective ACO is used to solve multi-objective semi-automated service composition [187, 220, 230]. For example, Zhang et al. [230] employ a "divide and conquer" strategy, which decomposes a given abstract workflow into multiple abstract execution paths without overlapped abstract services. This decomposing strategy results in a much smaller length of the execution paths, compared to the decomposition method proposed in [220]. This is due to that the decomposing strategy in [220] allows overlapped abstract services appear in the decomposed paths. Both two works employ an ACO algorithm, where the phenomenon is presented as a k-tuple for k objectives, rather than a single value. Wang et al. [187] includes a trust degree into non-functional attributes of Web services and proposes an adaptive ant colony optimization algorithm with a dynamically adjusted coefficient of pheromone strength.

Few works, such as [51], report on **many-objective and semi-automated service composition**. For example, De et al. [51] employ NSGA-II to optimize five different quality criteria (i.e., runtime, price, reputation, availability and reliability) simultaneously.

To solve **the second category of multi-objective service composition**

discussed in Sect. 1.2, MFEA can naturally cope with it through solving multiple combinatorial optimization tasks concurrently, producing multiple solutions, with one for each task. MFEA searches a unified search space based on a unified random-key representation over multiple tasks and sharing the implicit knowledge of promising solutions through the use of assortative mating across multiple tasks. It has shown its efficiency and effectiveness in several problem domains [17, 59, 225, 233]. For example, Yuan et al. [225] employ MFEA to concurrently handle four optimization problems with different local search operators. The success of this work is due to the use of permutation-based representations and the local search. Compared to the default random-key representation in MFEA, permutation-based representations are often effective for dealing with permutation-based problems, such as TSP, QAP, LOP, and JSP [225]. In the context of service composition, to meet the efficiency and cost requirements, [17] reported the first attempts that employ MFEA to solve two service composition tasks together using MFEA. This method achieves competitive results compared to single-objective EC techniques. However, due to the design of their representation, this work can only cope with semi-automated service composition. Furthermore, the number of tasks to be optimized concurrently is relatively small.

Dynamic Semi-Automated Approaches

Dynamic service composition do not assume that QoS remains static. In fact, it focuses on handling dynamic QoS. In the literature, such QoS is handled via different QoS models, such as a QoS model that assumes QoS values vary in bounded-intervals or can be estimated based on the past QoS distributions.

Bounded interval-based approaches rely on the bounded-interval QoS values to simulate periodically changed QoS and re-optimize composite services periodically [9, 125, 190]. For example, one work [190] employs ACO to handle QoS changes. Particularly, a newly proposed

pheromone updating strategy is used in the event of QoS changes. The idea of this updating strategy ensures that new solutions can be constructed by resetting pheromone information on the edges of their planning graph. Mostafa et al. [125] combine multi-objective optimization and Reinforcement Learning (RL) technique to solve multi-objective service composition problem in an uncertain and dynamic environment. RL is a popular machine learning technique for solving sequential decision-making problems with the aim to maximize some long-term rewards. It is utilized to deal with how actions are taken in an uncertain environment. In the dynamic service composition context, this uncertain environment is related to the dynamic QoS. In this work, web service composition is modelled based on the Partially Observable Markov Decision Process, and solutions to services composition are considered to be a set of decision policies. Each decision policy is obtained through a procedure of service selection towards constructing a composite service. They propose a method to learn an optimal selection policy to build near-optimal solutions. Several concurrent works, e.g., [83], propose a fuzzy-based QoS model based on the bounded-interval QoS values to measure the uncertainties in QoS. This fuzzy-based measure is used to rank preferences of candidate solutions evolved by GA. In a nutshell, QoS changes in the works discussed above are assumed to happen every fixed time. This assumption is the victim of idealization.

Predicted approaches are alternative approaches that assume QoS changes follow some historical patterns and can be predicted in the future [6, 30, 80, 108]. Some works [30, 80] assume that QoS follows a known probability distribution, and can be estimated based on the past QoS values. For example, [80] consider QoS as a discrete random model, and search for near-optimal composite services using simulated annealing. Different from [80], [30] employs stochastic models, i.e., quantile function [68] and Tchebysheff's inequality [135], for the estimation of QoS over the cases that the probabilistic distribution is known or unknown,

respectively. However, these approaches do not consider the impact of time on QoS. To address this limitation, [165] consider time-varying QoS by proposing a time-series prediction model while optimizing the predictive QoS of composite services. This problem is recently studied again as predictive-trend-aware service composition by the same authors [166]. They further conduct extensive case studies with diverse randomly-generated composition workflows. Although these works are capable of handling QoS in a predictive manner, they often require sufficient historical data. However, sufficient historical data are not always available for newly registered web services. In such cases, building an accurate and reliable prediction model may not be feasible.

Service reconfiguration approaches [23, 111, 124, 173] have been investigated by searching for a replacement of the component services that cannot be executed or have negative impacts on QoS. However, those methods are only effective for the semi-automated service composition, where a service composition workflow is known. Some of these works also aim to identify services for the replacement without violating the end-to-end QoS constraints. However, optimizing QoS in the event of service failures is not their focus. For example, [173] proposes a method to cluster services based on their functional properties. This method determines a set of backup services for service failures. Meanwhile, a multi-objective technique with GA is employed to select a suitable service from the backups. To minimize the number of backups for the selection, decision tree learning is used for the prediction of the performance based on QoS [124]. This technique outperforms other classification techniques investigated in this work, such as backpropagation neural network, support vector machine probabilistic neural network, and regression tree. In a nutshell, service reconfiguration approaches assume many backup services exist in advance, and they focus on developing effective and efficient methods for selecting proper services for the replacement of component services in the semi-automated context.

Robust approaches explore an interesting idea of handling service failures through distributed service deployment [65, 173]. In these works, a sufficient number of duplicated services can be jointly deployed to prevent service failures. They assume that sufficient services do exist in advance, and can be used for replacing the component services. These works are related but are clearly targeted to address a different robust problem than our thesis. Our robust problem focuses on constructing composite services with high robustness at the design stage. Such composite services are expected to be easily repaired in the event of service failures.

As summarized here, semi-automated dynamic service composition takes potential service failures, or QoS changes into account, focusing on the execution stage. These techniques assume that QoS changes periodical or can be predicted based on the historical data. In reality, services often fail sporadically in a highly unpredictable manner, and the lack of historical QoS data poses noticeable feasibility challenges. Other works focus on reselecting component services of composite services, assuming a sufficient number of services with compatible functionalities exist in advance. In reality, such an assumption is not always satisfied. Technically, all these approaches do not allow the changes of composition structures.

2.2.2 Fully Automated Web Service Composition

Single-Objective Fully Automated Approaches

Genetic Programming has been widely used for supporting fully automated service composition [42, 117, 126, 152, 182, 202, 221, 222] with direct representations. Particularly, tree-based representations could be ideal for practical use, since they can present all composition constructs (discussed in Sect. 2.1.2) as inner nodes of trees. GP is utilized for searching optimal solutions represented as trees. [152] proposes a context-free grammar for randomly initializing tree-based composite services with correct structures of composite services. In contrast, [221] randomly initializes

tree-based service composite services completely, but they develop adaptive crossover and mutation rates according to the diversity of the population for accelerating the speed of convergence. The two approaches [152, 221] discussed above utilize a penalization method for filtering incorrect solutions in their fitness functions. The **hybridization of GP and other techniques**, such as local search, are also explored. For example, to overcome the premature and proneness of GP, Tabu search is combined with GP to solve QoS-aware data-intensive web service composition [223]. Other works [42, 117] utilize a greedy search to create functionally valid DAG-based composition services, which are mapped to tree-based ones. During the evolutionary process, the correctness of the solutions is ensured by specific crossover and mutation operators. However, the tree-based composite services suffer a scalability issue since many replicas of subtrees are produced from the mapping process. To overcome this issue, [182] proposes a tree-like representation, on which the replicas of subtrees are handled by removing them, and inserting edges from the root of the replicas to the roots of the copies.

Graph-based Genetic Programming also attracts researchers' interests for solving fully automated service composition with direct representations [41, 43], such as DAGs. For example, GraphEvol is introduced in [41] with graph-based genetic operators, which are utilized to evolve individuals represented by DAGs. GraphEvol is compared to the previously discussed GP-based approaches [42, 117], and their experiment shows that graph-based approaches can produce composite services with much higher QoS. DAG-based representation can hardly present some composition constructs, such as loop and choice. To support choice construct, one recent work [43] investigates DAG-based representation supporting branches (i.e., choice) using GraphEvol [41]. However, this method has a big limitation, i.e., any nested choice composition construct is not supported.

GA has been widely used to solve fully automated service composi-

tion with indirect representations, e.g., permutation-based representations [44, 47, 156]. Such permutations need to be interpreted into service composition workflows using a decoding strategy. For example, [47] investigates a fixed-length or a variable-length permutation-based representation, and propose corresponding genetic operators to search the solutions evolved by GA. They find out that the fixed-length GA method outperforms the variable-length GA and GraphEvol [41]. Furthermore, the fixed-length GA enhanced by an exhaustive local search can achieve the best performance among all the competing methods, including two baseline methods (the variable-length GA and the fixed-length GA), GraphEvol [41], and a memetic variable-length GA. However, the neighbours produced by the exhaustive local search operators are likely to be decoded into the same composite solution. Therefore, both the effectiveness and efficiency of this local search operator demands further improvement.

PSO has also been introduced to tackle fully automated service composition, where composite services are represented as vectors, which are then decoded into solutions represented as DAGs [47, 45, 175]. For example, [45] proposes a PSO-based fully automated approach to generate a composite service from an optimized particle location. The idea is to translate the particle location into a service queue that can be decoded into a composite service. Therefore, finding an optimal composite solution is to discover an optimal location of the particle in the search space. Our recent work extends [45] to jointly optimize QoSM and QoS, where a weighted DAG is decoded, where edge weights correspond to QoSM between services. This work is not covered in this thesis because it does not achieve outstanding results in terms of effectiveness. However, it will be compared with our EDA-based methods in Sect. 3.7.

Graph search [31, 32, 33, 57, 75, 85, 209] is an alternative approach to fully automated service composition, compared to all the previously discussed EC-based approaches. Graph search works on searching composite services, which are constructed by subgraphs or paths from a service de-

pendency graph based on service dependencies. However, constructing such a service dependency graph could suffer a scalability issue when dealing with a large service repository with complexity service dependencies. This issue can get even worse when QoS optimization is considered [90]. For example, $A^\star$ search [164] is utilized to search composite services presented as paths, which are constructed from a sub-graph of a service dependency graph [151]. This sub-graph is extracted based on service requests. A recent work [57] transforms each search step on the service dependency graph as a dynamic knapsack problem, and proposes a knapsack-variant algorithm to effectively and efficiently generate composite services with minimal number of component services. However, these two works [57, 164] only focus on minimizing the number of component services in composite services without considering QoS or QoSM. Besides that, the scalability of this method suffers when the service repository grows. To address this critical issue, [31] proposes an QoS-aware service composition via a scalable way of pruning dependency graphs, and a novel path-based construction and selection method. This method can efficiently construct near-optimal composite services. However, it only considers a single quality criterion in QoS. To consider multiple quality criteria in QoS, a recent work [32] proposes an improved path-based search method based on [31]. Particularly, a node (i.e., an atomic service) associated with a higher rank is preferred in a path construction, and nodes are ranked based on the concept of dominance over multiple QoS quality criteria.

In summary, with the help of conventional EC techniques, indirect approaches, such as permutation-based representations with decoding strategies, can often reach better quality, compared to direct approaches with DAG and tree-based representations. Furthermore, an indirect memetic approach, e.g., a fixed-length memetic GA, has reported its outstanding effectiveness in finding high-quality composite solutions. Despite recent success in indirect approaches, new composite services

are produced using conventional EC techniques, relying on the implicit knowledge of promising solutions. As discussed previously, EDA is different from conventional EC techniques, it can explicitly learn the knowledge of promising solutions. Therefore, new permutation-based representations combined with EDA can be further studied to handle single-objective fully automated service composition.

Multi-Objective Fully Automated Approaches

Very limited works have ever proposed EC-based approaches to **the first category of multi-objective fully automated service composition approaches**, but many works on multi-objective semi-automated service composition have been reported [35, 110, 150, 113, 172, 187, 201, 214, 217, 220, 230]. To the best of our knowledge, [40, 46] are the two recent attempts on fully automated service composition with the aim of handling two quality criteria in QoS. [46] develop a multi-objective method using **NSGA-II** and a fragmented tree-based representation. However, this fragmented tree-based representation does not show its effectiveness for finding better Pareto solutions in their experiment, comparing to an indirect representation. To improve the performance of NSGA-II, the **hybridization of NSGA-II and other techniques** are also explored. For example, the same authors in [46] later proposed two hybridized methods [40], called Hybrid and Hybrid-L, with the indirect representation. Both methods are proposed based on a hybridization of NSGA-II and MOEA/D. In addition, based on Hybrid, Hybrid-L introduces local search to the decomposed subproblems. The limitations of this work have already been addressed in Section 4.1. For example, a large number of decomposed subproblems is pre-defined, and a simple form of local search is not effective to be performed on each subproblem. Despite achieving some promising results, opportunities still exist to address these limitations. One work [224] is reported as the first attempt on **many-objective and fully automated service composition** problem. Particularly, a many-objective optimization

algorithm, NSGA-III [53] is employed with a tree-based representation.

To the best knowledge of us, no works have been conducted for solving **the second category of multi-objective fully automated service composition** problem, but one previously discussed worked [17] is reported to cope with a similar problem but in a semi-automated way.

Dynamic Fully Automated Approaches

Although many techniques for semi-automated dynamic service composition have been explored, these techniques do not support fully automated service composition. In addition, they ignore the importance of considering dynamic changes when constructing composite services at the design stage. In line with this issue, it is important to consider the robustness of solutions at the design stage. This robustness is defined as the expected quality of a composite service across all possible scenarios for the dynamic changes. However, it is difficult to enumerate the scenarios. Therefore, fitness approximation becomes an important technique for measuring the robustness [87] under the frame of EC. Besides that, evolutionary control is a technique that manages fitness approximation on the evolved solutions by EC. Herein we will review these two important techniques as follows:

Fitness approximation has been widely used to solve computationally expensive single-objective and multi-objective problems [87, 86]. Existing fitness approximation techniques can be classified into three types: problem approximation, data-driven functional approximation, and fitness inheritance [87, 86]. Problem approximation often uses an approximate, easier-to-solve problem to replace the original problem. For example, Monte Carlo sampling, instead of complete sampling, is used to measure the robustness of solutions. Data-driven functional approximation trains explicit models (also called meta-models or surrogates) based on historical solutions using various meta-modelling techniques, such as polynomials or Gaussian processes. Fitness inheritance estimates the fitness of one individual by the fitness of other similar individuals. In the

literature, fitness approximation has been utilized to find solutions with optimized robustness in many problems, such as aggregate production planning [105] and flexible job-shop problems [203]. Such problems are called robust optimization problems. In fact, decision-makers are not only concerned the performance of the solution but also the sensitivity of performance with respect to small changes.

Evolutionary control is a technique to manage fitness approximation for the evaluations on individuals. It aims to achieve a good trade-off between the accuracy of an evaluation and computational cost. Techniques in evolutionary control can be grouped into the individual-based and the generation-based [87, 86]. In the individual-based, some of the individuals are evaluated using the fitness approximation while the others are evaluated using the real fitness function. For example, [69] proposed an evolutionary control that ensures individuals with good estimated fitness values will be re-evaluated on the real fitness function. These good individuals are selected from each individual cluster of each population. In contrast, in the generation-based evolutionary control, individuals in one generation are either evaluated using the fitness approximation or the real fitness function. Recently, much attention has been paid to adaptive adjustment on the frequency of fitness approximation in both individual-based and generation-based evolutionary control. This is due to that a fixed evolutionary control frequency can be ineffective as the fidelity of the approximate model may vary significantly during the optimization [87]. For example, an adaptive generation-based evolutionary control is proposed based on the error of the fitness approximation [88].

Like many real-world robust optimization problems, "real fitness function" actually does exist for measuring the robustness of composite services. However, it is not feasible to compute the robustness over all possible events of service failures. Proposing an effective and accurate robustness measure for composite services has not been studied in the literature. In addition, evolutionary control techniques that manage the robustness

estimation also need to be studied to achieve a good trade-off between the accuracy and computation cost.

Service Composition Approaches without Handling Optimization

A few techniques do not focus on optimization. Instead, they focus on constructing functionally valid composite services. **AI planning techniques** have been widely employed in service composition for such a purpose [137]. Various AI planning approaches [77, 146, 174, 192] have been presented to solve semantic web service composition problems using Graphplan algorithm [20]. For example, [192] employs Graphplan to secure the functionally valid composite services, where atomic web services are concretely selected and accurately matched for the desired functionality. The pitfalls of this approach are procuring linear sequences of actions only, without dealing with optimization. Some other approaches [102, 162] rely on frameworks supported by particular agent programming languages, such as Golog [162] and SHOP2 [161]. For example, in [162], a composition framework supported by Golog is used to describe the properties of services and users' preferences. Therefore, Golog can effectively perform a search for constructing a preferable composite service. As summarized here, on the one hand, AI planning techniques may suffer scalability issues when large service repositories are given [137], not capable of handling optimization. To overcome these limitations, the hybridization of AI planning techniques and other techniques, such as EC, have been proposed in effectively and efficiently handling optimization in web service composition [36, 37, 47, 117, 143].

Most small and medium business rely on relational database systems to manage information and data. By utilizing the database systems, one work introduces an **Entity-Relationship (ER) model based approach** for web service composition [205]. This work relies an ER model to transform existing knowledge into ontology in OWL for the service annotation, which allows discovery, reasoning and composition of semantic web ser-

vices. However, loop and switch constructs cannot be effectively handled in their approach, and they do not deal with optimization at all.

2.3 Summary

This chapter presents a background knowledge in web service composition and an overview of related techniques in the areas of EC and AI planning. It also reviews some recent works conducted in the context of single-objective optimization, multi-objective optimization, and dynamic optimization for both semi-automated and fully automated service composition. The first related area is **semi-automated web service composition**, where an abstract workflow of service execution is given in advance and is strictly followed by all the candidate composite services. Therefore, semi-automated web service composition focuses on selecting proper services to fit each abstract slot of the composition workflow [123]. Various techniques have been investigated and treat service requests as single-objective optimization problem [7, 24, 26, 27, 28, 35, 36, 37, 58, 62, 63, 64, 76, 77, 82, 101, 112, 143, 144, 167, 188, 191, 194, 198, 196, 199, 204, 216, 207, 208, 206, 218, 231], multi-objective optimization problem [51, 110, 113, 150, 172, 187, 214, 220, 230], and dynamic optimization problem [9, 6, 30, 23, 65, 80, 108, 125, 190, 111, 124, 173]. Particularly, the single-objective optimization problem aims to find composite services with a minimized number of component services [12, 126, 152, 174], or with an optimized score based on QoS [24, 117, 202, 221, 222], or with an optimized score based on both QoS and QoSM [36, 58, 101]. The multi-objective optimization problem aims to find a set of approximated Pareto solutions that present different trade-offs. The dynamic optimization problem aims to efficiently find high-quality composite services that can cope with frequent QoS changes or unexpected service failures. The most critical limitation of the semi-automated works is that they do not allow the changes of composition structure, overlooking other workflow structures that lead to better qual-

ity. In addition, very limited number of works, i.e., [36, 58, 101], is reported to jointly consider QoSM and QoS of composite services.

The second related area is **fully automated web service composition**, which gradually build service composition workflows while selecting web services without strictly obeying any composition workflow. In the context of **single-objective fully automated web service composition**, various techniques have been explored for this purpose. Particularly, GP with tree and graph-based representations has been studied for fully automated web service composition, where trees and DAGs are evolved with corresponding genetic operators for the searching candidate solution with optimized QoS [41, 42, 43, 117, 126, 152, 182, 202, 221, 222]. On the other hand, GA with indirect representation, such as permutations, has shown its advantage in finding higher quality composite services [44, 47, 156], compared to GP-based approaches. However, it often consumes more computation time as decoding strategies are required to decode permutations into functionally valid DAGs. Apart from GP and GA, PSO is also employed to solve this service composition problem with a similar decoding strategy utilized in GA [47, 45, 175]. Furthermore, alternative composition approaches based on Graph search can also cope with fully automated service composition, searching composite services represented as paths or sub-graphs based on a service dependency graph [31, 32, 33, 75, 85, 209]. Despite some recent successes, existing fully automated EC-based approaches use implicit knowledge of promising solutions to search for near-optimal composite services. The performance of using explicit knowledge of promising solutions remains to be studied. EDA is different from most conventional EC techniques and uses explicit knowledge encoded by a probabilistic model based on the distribution of a set of parent individuals. This technique will be studied in Chapter 3 with the aim to jointly optimize both QoSM and QoS, and it is compared to some state-of-the-art EC-based techniques and graph search techniques.

For **the first category of multi-objective fully automated web service**

composition, two recent works [40, 46] report the first two attempts to cope with this problem. One of the reported works that proposes two hybridized approaches, namely, Hybrid and Hybrid-L [40], outperforms the other recent work [46] and NSGA-II in finding much better Pareto solutions. Both Hybrid and Hybrid-L combine the use of two optimization algorithms, i.e., NSGA-II and MOEA/D. Particularly, Hybrid-L allows the local search to be performed on numerous decomposed single-objective scalar optimization subproblems. Despite this success, two drawbacks of this method can be addressed: (1) a large number of decomposed subproblems needs to be pre-defined, and (2) a simple form local search (i.e., so-called one-point "swap") is less effective and efficient to make local improvements. This is because their local search is randomly applied to every subproblem without focusing on the most suitable candidate solutions in each generation. These drawbacks are addressed by proposing a novel memetic method in the first part of Chapter 4 in the thesis. For **the second category multi-objective fully automated web service composition**, one recent work reports the first attempt to employ MFEA to solve the second category of multi-objective web service composition but in a semi-automated way. Besides that, the number of composition tasks to be optimized is very small, i.e., two. To address these limitations, the second part of Chapter 4 will introduce two novel MFEAs.

The last related area is **dynamic fully automated service composition**. To the best of our knowledge, no works have been reported in this area. Instead, many works have been reported on semi-automated dynamic service composition, handling QoS changes via different QoS models, such as bounded-intervals [9, 125, 190] or prediction models [6, 30, 80, 108]. One key limitation of these bounded-interval based approaches is that they assume QoS changes happen periodically. Meanwhile, prediction-based approaches often require sufficient historical data, which is not always available for newly registered web services. Other approaches, such as service reconfiguration approaches [23, 111, 124, 173] and robust approaches

[65, 173], do not point out how and when QoS changes exactly happen. They assume that their methods can be applied once QoS changes are detected. They also assume that a large number of services exist in advance for every abstract service slot of composition workflows. In reality, such assumptions are not always satisfied. Apart from each specific limitation associated with different approaches, all these approaches ignore the importance of handling QoS changes at the design stage. To address all these limitations, Chapter 5 will introduce a new dynamic service composition problem and propose novel EC-based methods to effectively and efficiently solve this problem.

Chapter 3

Single-Objective Fully Automated Web Service Composition

3.1 Introduction

The two most notable challenges for semantic web service composition, discussed in Chapter 1, are to ensure interoperability of services and achieve Quality of Service (QoS) optimisation. These two challenges give birth to *semantic web services composition* that enables a good interoperability for chaining semantic web services via functional attributes [170, 227], and *QoS-aware service composition* that aims to find composite services with optimised QoS [158], respectively. Existing works on service composition focus mainly on addressing only one of the two challenges above, ignoring the importance of addressing these two challenges jointly. In fact, composite services could be ranked not only by the well-known non-functional attributes (i.e., QoS), but also by the functional attributes (i.e., QoSM) as an indicator of functional fit [101]. In this chapter, we will *address these two challenges jointly by introducing a comprehensive quality model*. We will focus on fully automated semantic **W**eb **S**ervice **C**omposition with the aim to

optimize **C**omprehensive **Q**uality (i.e., a combination of QoSM and QoS) of composite services (henceforth referred to as **WSC-CQ**).

In the literature, huge efforts have been devoted to QoS-aware web service compositions in the domain of *semi-automated web service composition* [26, 27, 28, 64, 101, 167, 188, 191, 198]. These works produce composite services that strictly follow a pre-defined service composition workflow. In the past few years, *fully automated service composition* has been a promising research field, and it constructs service workflows automatically with service selections, without strictly obeying any specific workflows [147]. Fully automated service composition problem is an NP-hard problem [123]. To efficiently find "good enough" composite services, many approaches [45, 72, 117, 41, 145, 221] employ EC techniques to automatically generate composite services with high QoS.

The successes of the EC-based service composition approaches rely on the use of knowledge, which is defined as useful information acquired through experience (i.e., promising composite services) [131]. Based on a practical or theoretical understanding of promising solutions, the knowledge can be implicit (e.g., recombining genetic information of two selected parent solutions) or can be explicit (e.g., learning distribution models from a set of promising solutions). By iteratively updating and utilizing the knowledge, new candidate solutions are generated until the most desired solution is found [74].

Conventional EC techniques use implicit knowledge of promising solutions to successfully achieve QoS-aware web service composition [117, 152, 175, 221], where new candidate solutions are generated by one or more evolution operators on parent individuals. For example, GA produces new candidate solutions by crossover, operated on two selected parent individuals. Whereas, EDA is different from most conventional EC techniques. As we discussed in Sect. 2.1.3, EDA uses explicit knowledge encoded by a probabilistic model based on the distribution of a set of parent individuals, which often refers to a superior subpopulation. Par-

ticularly, permutations are often utilized in EDA, and they can naturally present solutions for many problems.

Despite recent successes on applications of EDA [189, 195], such as arc routing and assembly flow-shop scheduling, how can EDA to be used to successfully solve the WSC-CQ problem remains an important research question. To the best of our knowledge, two approaches [140, 142] utilize EDA for solving semi-automated service composition problems. [142] employs a binary representation to encode composite services, and it learns a set of independent probabilities where 0 or 1 appears at each bit of the binary representation. Different from [142], a different distribution model based on Restricted Boltzmann Machine is learned based on a similar binary representation [140]. These two works can only cope with semi-automated service composition because their binary representations are designed to encode solutions with a fixed structure of composition workflows. Learning distributions over a pre-defined fixed structure is relatively less challenging. To support learning distributions over varied structures of composite services, opportunities still exist to further investigate new representations with suitable distribution models that can capture useful knowledge for achieving fully automated service composition. In addition, effective sampling algorithms may need to be investigated for sampling solutions from these distribution models. Therefore, we will *propose EDA-based approaches for fully automated service composition, where the comprehensive quality of composite services (see details in Sect. 3.3) is to be optimized*. These approaches will be compared to some state-of-the-art works [47, 32, 175] that were proposed to solve the same or similar problem.

EDA stresses more on global exploration, rather than local exploitation [195]. This is because the distribution model in EDA focuses on exploring more promising regions in the entire solution space, without attempting to improve the quality of any specific solutions evolved previously. However, the optimization performance can often be improved directly through local modifications on promising solutions. By restricting the tar-

get region for local search and avoiding most of the randomness involved in sampling directly from the distribution model, it can potentially expedite the search of optimal solutions. Therefore, memetic EDA can improve the competency of EDA in finding more effective solutions by introducing local search. It has been successfully applied to many optimization problems, such as arc routing and assembly flow-shop scheduling [189, 195].

Despite some recent successes in memetic EDA, those memetic approaches [189, 195] cannot be simply applied to our web service composition problems. This is because their local search operators are designed to exploit the neighbourhood of solutions, which are encoded by problem-specific representations. Besides that, local search operators are often proposed based on domain-specific knowledge for making effective local improvements. To develop a memetic EDA with local search that works effectively and efficiently for the WSC-CQ problem, several challenges remain to be addressed:

Firstly, a composite service is commonly represented as a DAG [41, 45]. Exploring the neighbourhood of a DAG, especially large DAGs, is computationally infeasible [1]. Therefore, researchers [47, 98] often indirectly define the neighbourhood of a composite service represented in the form of a permutation, which can be converted to a DAG through a separate decoding process. For example, so-called 'swap' operators proposed in [44] can produce neighbours by swapping two random elements in a permutation. Consequently, a neighbourhood is defined by a collection of permutations obtainable through a number of swaps performed on any given permutation. However, such neighbourhood often contains a large proportion of neighbouring permutations with inferior quality. For effective local search, the neighbourhood must be refined to exclude most of the clearly unwise swapping choices by exploiting domain-specific knowledge.

Secondly, as we know, it is very challenging to determine which candidate solutions are to be selected for local search. Particularly, some ques-

tions always need to be answered while developing memetic algorithms: Should an equal chance be given to all the candidate solutions or should only elite solutions be considered for local search? What are elite solutions, and how many of them should be modified locally? It is difficult to answer these challenging questions, because answers often vary due to many factors, such as EC algorithms.

Third, a traditional strategy that exclusively explores the whole neighbouring space of composite services can incur high computation cost. For example, if a swap operator is utilized for exploring the neighbourhood of a permutation of length n, the size of neighbourhood over this permutation is $\frac{n(n-1)}{2}$, based on all possible combinations for pair swaps [98]. In [44], this neighbourhood size is reduced to $n-1$ by restricting all the pair swaps to always include a selected position. In the context of service composition, the length of such a permutation is usually equivalent to the size of the service repository. When the size of the service repository is very large, e.g., a maximum of 15211 web services are contained in WSC-09 benchmark [92], exploiting the neighbourhood of size $n-1$ becomes very expensive for practical use.

Fourth, one proportion of promising solutions are likely to be repetitively sampled from an adjusted probability distribution model throughout generations. These repeatedly sampled solutions also require computation time for repetitive decoding and evaluations, which are very costly. However, these solutions are often favourable to users since they are candidate solutions with high quality, and help the distribution model to get converged along with the generations. Therefore, reducing computation time caused by the repetitive sampling while helping the distribution model to converge is very challenging.

To address these challenges above, *we will propose memetic EDA-based approaches with the aim to optimize the comprehensive quality of the composite services*. For the purpose of investigating the effectiveness of several neighbourhood structures of composite services, we propose several domain-

specific local search operators. Experimental study on the effectiveness of employing these different local search operators in EDA is conducted. Particularly, one proposed memetic EDA achieves the highest effectiveness and efficiency among all the competing approaches. The following objectives are sought in this chapter:

1. To propose a comprehensive quality model that addresses QoS and QoSM to reach a desirable compromise.

2. To learn a suitable distribution that can capture the positions of component services over varied service composition structures, and to produce composite services with optimized comprehensive quality, we propose an **EDA**-based approach based on **N**ode **H**istogram **M**atrix (henceforth referred to as **EDA-NHM**). Particularly, we first transform the service composition problem into a permutation-based problem with a novel fixed-length, permutation-based representation. This representation enables reliable and accurate learning of the underlying probability distribution model. Subsequently, we employ histogram-based sampling [168] to sample promising permutations, which also demand to be decoded into DAG-based composite services. Furthermore, we utilize an archiving technique to reserve promising composite services. By including these solutions in the next generation for learning a NHM, we not only can save computation time in sampling new solutions, but also better trace promising searching areas.

3. To learn a suitable distribution that can naturally capture the information of service dependencies for constructing effective composite services, and to produce composite services with optimized comprehensive quality, we propose an **EDA**-based approach based on the **E**dge **H**istogram **M**atrix (henceforth referred to as **EDA-EHM**). Particularly, to initialize correct entries of EHM, we develop an ontology-based querying technique for efficiently querying all the

possible service dependencies among all the services in the service repository. Subsequently, we propose a way of using EHM to learn service dependencies from a set of promising composite services. Consequently, to effectively and efficiently sample functionally valid high-quality composite services from EHM directly, we propose a **G**uided **E**dge **H**istogram-Based **B**ackward **G**raph **S**ampling **A**lgorithm (henceforth referred to as **GEHBGSA**).

4. To demonstrate the effectiveness of EDA-EHM and EDA-NHM, we first create a more challenging, augmented version of the service composition benchmarks based on WSC-08 [15] and WSC-09 [92]. In particular, the new benchmarks extend WSC-08 and WSC-09 with QoS attributes in QWS dataset [4]. Moreover, the number of web services in the service repository is doubled as new benchmarks (with much bigger searching space) to demonstrate that our EDA-based approaches can maintain high performance on large problems. These two benchmarks have been made freely available online [1]. We experimentally compare EDA-EHM and EDA-NHM with some state-of-the-art methods that have been recently proposed to solve the same or a similar service composition problem using the new benchmark. These state-of-the-art methods include a PSO-based approach [175] (henceforth referred to as PSO), a fixed length GA-based approach [47] (henceforth referred to as FL), and a non-EC deterministic approach based on a path construction and selection method from the service dependency graph [32] (henceforth referred to as PathSearch). Our experimental results illustrate that EDA-NHM can outperform all the competing methods in finding composite services with high comprehensive quality, while EDA-EHM can achieve the highest efficiency among all the competing EC-based methods with a reasonable trade-off in the comprehensive

[1]Two augmented benchmarks for automated web service composition is available from https://github.com/chenwangnida/Dataset

quality.

5. To further improve the performance of EDA-NHM, we propose memetic EDA-based approaches by introducing an effective local search. Particularly, we first propose several neighbourhood structures for permutations. These neighbourhoods are created by developing several novel domain-specific local search operators, based on constructing and swapping building blocks of composite services. Secondly, to significantly reduce the computation time of our proposed memetic EDA-based approach, an efficient local search strategy is introduced by combining a fitness uniform distribution scheme for selecting suitable solutions and stochastic local search operators for effectively and efficiently exploiting neighbours.

6. To evaluate the performances of our proposed memetic EDA-based approaches, we experimentally compare them with one state-of-the-art memetic GA method [47] (henceforth referred to as MEFL) and EDA-NHM as a baseline. MEFL was recently proposed to solve similar service composition problems, achieving the highest effectiveness in finding high-quality composite services with the help of local search. Meanwhile, EDA-NHM is chosen as a baseline because it achieves the best effectiveness without local search in our previous experiment. Our experimental results illustrate that one of our proposed memetic EDA-based approaches can achieve cutting-edge performance.

3.2 Chapter Organization

The remainder of this chapter is organized as follows: Sect. 3.3 formulates the WSC-CQ problem with a comprehensive quality model. Sect. 3.4 presents an initial step, i.e., re-processing service repository, for all service composition algorithms discussed in our thesis. Sect. 3.5 and Sect. 3.6

present our EDA-NHM and EDA-EHM algorithms to the WSC-CQ problem. Sect. 3.7 outlines the experimental design and results for evaluating the performance of EDA-NHM and EDA-EHM. Sect. 3.8 present our memetic EDA algorithms. Sect. 3.9 outlines the experimental design and results for evaluating the performance of memetic EDA algorithms. Sect. 3.10 summarises this chapter.

3.3 The WSC-CQ Problem

A *semantic web service* (*service*, for short) is considered as a tuple $S = (I_S, O_S, QoS_S)$ where I_S is a set of service inputs that are consumed by S, O_S is a set of service outputs that are produced by S, and $QoS_S = \{t_S, ct_S, r_S, a_S\}$ is a set of non-functional attributes of S. The inputs in I_S and outputs in O_S are parameters modeled through concepts in a domain-specific ontology $\mathcal{O}$. The attributes t_S, ct_S, r_S, a_S refer to the response time, cost, reliability, and availability of service S, respectively, which are four commonly used QoS attributes [226].

A *service repository* $\mathcal{SR}$ is a finite collection of services supported by a common ontology $\mathcal{O}$.

A *composition task* (also called *service request*) over a given $\mathcal{SR}$ is a tuple $T = (I_T, O_T)$ where I_T is a set of task inputs, and O_T is a set of task outputs. The inputs in I_T and outputs in O_T are parameters that are semantically described by concepts in the ontology $\mathcal{O}$.

Two special atomic services $Start = (\emptyset, I_T, \emptyset)$ and $End = (O_T, \emptyset, \emptyset)$ are always included in $\mathcal{SR}$ to account for the input and output of a given composition task T.

We use *matchmaking types* to describe the level of a match between outputs and inputs [132]. For concepts $\mathfrak{a}, \mathfrak{b}$ in $\mathcal{O}$ the *matchmaking* returns $exact$ if $\mathfrak{a}$ and $\mathfrak{b}$ are equivalent ($\mathfrak{a} \equiv \mathfrak{b}$), $plugin$ if $\mathfrak{a}$ is a sub-concept of $\mathfrak{b}$ ($\mathfrak{a} \sqsubseteq \mathfrak{b}$), $subsume$ if $\mathfrak{a}$ is a super-concept of $\mathfrak{b}$ ($\mathfrak{a} \sqsupseteq \mathfrak{b}$), and $fail$ if none of the previous matchmaking types is returned. As discussed in Sect. 2.1.2, we are

only interested in $exact$ and $plugin$ matches for robust compositions. As argued in [101], $plugin$ matches are less preferable than $exact$ matches due to the overheads associated with data processing. For $plugin$ matches, the semantic similarity of concepts is suggested to be considered when comparing different $plugin$ matches.

A *robust causal link* [103] is a link between two matched services S and S', denoted as $S \rightarrow S'$, if an output $\mathfrak{a}$ ($\mathfrak{a} \in O_S$) of S serves as the input $\mathfrak{b}$ ($\mathfrak{b} \in O_{S'}$) of S' satisfying either $\mathfrak{a} \equiv \mathfrak{b}$ or $\mathfrak{a} \sqsubseteq \mathfrak{b}$. For concepts $\mathfrak{a}, \mathfrak{b}$ in $\mathcal{O}$, the *semantic similarity* $sim(\mathfrak{a}, \mathfrak{b})$ is calculated based on the edge counting method in a taxonomy like WorldNet [160]. Advantages of this method are simple calculation and accurate measure [160]. Therefore, the *matchmaking type* and *semantic similarity* of a robust causal link is defined as follows:

$$type_{link} = \begin{cases} 1 & \text{if } \mathfrak{a} \equiv \mathfrak{b} \ (exact \text{ match}) \\ p & \text{if } \mathfrak{a} \sqsubseteq \mathfrak{b} \ (plugin \text{ match}) \end{cases} \tag{3.1}$$

$$sim_{link} = sim(\mathfrak{a}, \mathfrak{b}) = \frac{2N_{\mathfrak{c}}}{N_{\mathfrak{a}} + N_{\mathfrak{b}}} \tag{3.2}$$

with a suitable parameter p, $0 < p < 1$, and with $N_{\mathfrak{a}}$, $N_{\mathfrak{b}}$ and $N_{\mathfrak{c}}$, which measure the distances from concept $\mathfrak{a}$, concept $\mathfrak{b}$, and the closest common ancestor $\mathfrak{c}$ of $\mathfrak{a}$ and $\mathfrak{b}$ to the top concept of the ontology $\mathcal{O}$, respectively. However, if more than one pair of matched output and input exist from service S to service S', $type_e$ and sim_e will take on their average values.

The *QoSM* of a composite service is obtained by aggregating over all the robust causal links as follows:

$$MT = \prod_{j=1}^{m} type_{link_j} \tag{3.3}$$

$$SIM = \frac{1}{m} \sum_{j=1}^{m} sim_{link_j} \tag{3.4}$$

Formal expressions as in [116] are used to represent service compositions. The constructors $\bullet$, $\|$, $+$ and $*$ are used to denote sequential com-

position, parallel composition, choice, and iteration, respectively. The set of *composite service expressions* is the smallest collection $\mathcal{SC}$ that contains all atomic services and that is closed under sequential composition, parallel composition, choice, and iteration. That is, whenever $C_0, C_1, \ldots, C_d$ are in $\mathcal{SC}$ then $\bullet(C_1, \ldots, C_d)$, $\| (C_1, \ldots, C_d)$, $+(C_1, \ldots, C_d)$, and $*C_0$ are in $\mathcal{SC}$, too. Let C be a composite service expression. If C denotes an atomic service S then its QoS is given by QoS_S. Otherwise the QoS of C can be obtained inductively as summarized in Table 3.1. Herein, $p_1, \ldots, p_d$ with $\sum_{k=1}^{d} p_k = 1$ denote the probabilities of the different options of the choice $+$, while ℓ denotes the average number of iterations. Therefore, QoS of a composite service, i.e., availability (A), reliability (R), execution time (T), and cost (CT) can be obtained by aggregating a_C, r_C, t_C and ct_C as in Table 3.1.

Table 3.1: QoS calculation for a composite service expression C.

$C =$	$r_C =$	$a_C =$	$ct_C =$	$t_C =$
$\bullet(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} ct_{C_k}$	$\sum_{k=1}^{d} t_{C_k}$
$\| (C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} ct_{C_k}$	$MAX\{t_{C_k} \mid k \in \{1, ..., d\}\}$
$+(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} p_k \cdot r_{C_k}$	$\prod_{k=1}^{d} p_k \cdot a_{C_k}$	$\sum_{k=1}^{d} p_k \cdot ct_{C_k}$	$\sum_{k=1}^{d} p_k \cdot t_{C_k}$
$*C_0$	${r_{C_0}}^{\ell}$	${a_{C_0}}^{\ell}$	$\ell \cdot ct_{C_0}$	$\ell \cdot t_{C_0}$

As discussed in Sect. 2.1.2, we mainly focus on two constructors, sequence $\bullet$ and parallel $\|$, similar as most automated service composition works [117, 42, 41, 45] do, where composite services are represented as DAGs. We can easily calculate QoS of a composite service that is represented as a DAG according to Table 3.1.

When multiple quality criteria are involved in decision making, the fitness of a solution is defined as a weighted sum of all individual criteria in Eq. (3.5), assuming the preference of each quality criterion based on its relative importance is provided by the user [79]:

$$Fitness(C) = w_1\hat{MT} + w_2\hat{SIM} + w_3\hat{A} + w_4\hat{R} + w_5(1-\hat{T}) + w_6(1-\hat{CT}) \quad (3.5)$$

with $\sum_{k=1}^{6} w_k = 1$ ($w_k \geqslant 0$). This objective function is defined as a *comprehensive quality model* for service composition. We can adjust the weights according to the user's preferences. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$, $\hat{R}$, $\hat{T}$, and $\hat{CT}$ are normalized values calculated within the range from 0 to 1 using Eq. (3.6). To simplify the presentation, we also use the notation $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)$ $= (MT, SIM, A, R, T, CT)$. Q_1 and Q_2 have a minimum value of 0 and a maximum value of 1. The minimum and maximum value of Q_3, Q_4, Q_5, and Q_6 are calculated across all the relevant services, which are discovered using a greedy search technique, see the following Sect. 3.4

$$\hat{Q}_k = \begin{cases} \frac{Q_k - Q_{k,min}}{Q_{k,max} - Q_{k,min}} & \text{if } k = 1, \ldots, 4 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ \frac{Q_{k,max} - Q_k}{Q_{k,max} - Q_{k,min}} & \text{if } k = 5, 6 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \quad (3.6)$$

The goal of comprehensive quality-aware service composition is to find a composite service expression $C^\star$ that maximizes the objective function in Eq. (3.5). $C^\star$ is hence considered as the best possible solution for a given composition task T.

3.4 Pre-processing of A Service Repository

Pre-processing of a service repository is an initial step for building composite services. We achieve two goals at this pre-processing stage: (1) reduce the size of the service repository $\mathcal{SR}$ by keep only those that are relevant to the service composition task T, and (2) identify service layers of these relevant services. The service layers will be utilized in our proposed sampling technique in EDA-EHM in Sect. 3.6.4 and a proposed stochastic local search operator in Sect. 3.8.3.

ALGORITHM 10 outlines the pre-processing of a service repository using a greedy search that repeatedly searches services in $\mathcal{SR}$. Particularly, it starts with searching services in $\mathcal{SR}$ that can be fully satisfied by $OutputSet$, which is initialized with I_T. Subsequently, services fulfilled by $OutputSet$ are included into the first layer, denoted as $\mathcal{L}_1$, and are removed from SR. $OutputSet$ is then updated with the outputs of the fulfilled services found in the first iteration. This searching process will stop when no additional services can be fulfilled by the updated $OutputSet$. Finally, suppose n services layers are found, service layers (denoted as $\mathcal{L} = \bigcup_{l \in \{1,\ldots,n\}} \mathcal{L}_l$) are returned. Besides that, an updated $\mathcal{SR}$ is returned by combining all the services in each layer.

ALGORITHM 10. Pre-processing of a Service Repository.

Input : T, $\mathcal{SR}$, $l \leftarrow 1$

Output: an updated $\mathcal{SR}$ and service layers $\mathcal{L}$

1: Initial layer $\mathcal{L}_l$ with an empty web service set;
2: Initial $OuputSet$ with I_T;
3: **while** *at least one* S *satisfied by* $OutputSet$ **do**
4: Put satisfied services into layer $\mathcal{L}_l$ and remove them from $\mathcal{SR}$;
5: Add all outputs of the satisfied services to $OutputSet$;
6: $l \leftarrow l + 1$;
7: $\mathcal{L} = \bigcup_{l \in \{1,\ldots,n\}} \mathcal{L}_l$;
8: **return** service layers $\mathcal{L}$ and an updated $\mathcal{SR}$;

Example 1. *We consider a* composition task $T = (\{a, b\}, \{i\})$ *and a* $\mathcal{SR}$ *consisting of seven atomic services.* $S_0 = (\{b\}, \{i\}, QoS_{S_0})$, $S_1 = (\{a\}, \{f, g\}, QoS_{S_1})$, $S_2 = (\{a, b\}, \{h\}, QoS_{S_2})$, $S_3 = (\{f, h\}, \{i\}, QoS_{S_3})$, $S_4 = (\{a\}, \{f, g, h\}, QoS_{S_4})$, $S_5 = (\{a, c\}, \{f, g, h\}, QoS_{S_4})$ *and* $S_6 = (\{c, d, e\}, \{f, g, h\}, QoS_{S_4})$. *The two special services* $Start = (\emptyset, \{a, b, e\}, \emptyset)$ *and* $End = (\{i\}, \emptyset, \emptyset)$ *are defined by the given composition task* T. *Fig. 3.1 shows an example of discovering relevant*

services and service layers over a service request T, where five related services (i.e., S_0, S_1, S_2, S_3, and S_4) and two layers (i.e., $\mathcal{L}_1$ and $\mathcal{L}_2$) are found. In $\mathcal{L}_1$, S_0, S_1, S_2, and S_4 can be immediately satisfied by the task inputs I_T, i.e., $\{a, b\}$, of task T, and they have the same distance to $Start$ (Note that the distance is measured by the number of predecessors). Different from the services in $\mathcal{L}_1$, S_3 in $\mathcal{L}_2$ requires additional inputs provided by services in $\mathcal{L}_1$, with a longer distance to $Start$.

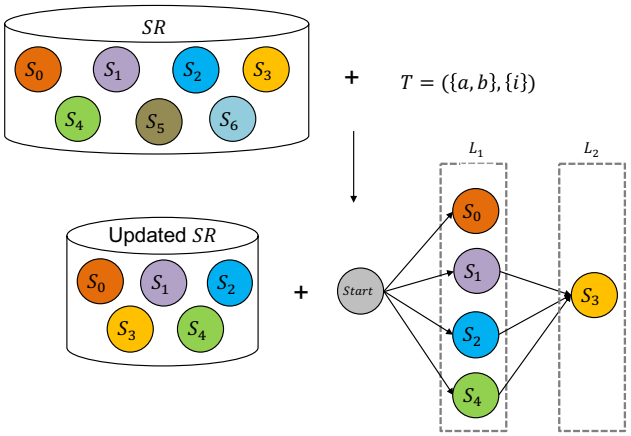


Fig. 3.1: An example of pre-processing of service repository for a service request T.

3.5 The EDA-NHM Algorithm

In this section, we present our EDA-NHM algorithm for the WSC-CQ problem. We will start with an outline of EDA-NHM in Sect. 3.5.1. Subsequently, we discuss two proposed ideas behind this approach: the first idea is to introduce a novel permutation-based representation for presenting composite services, allowing reliable and accurate learning of NHM from promising solutions (see details in Sect. 3.5.2); the second idea is to apply NHM for learning the knowledge of promising solutions (see details in Sect. 3.6.3).

The EDA strategy has been applied with some success to optimization problems where candidate solutions can be represented as permutations [29]. The success, however, strongly depends on the ability to define a suitable probability distribution model for the problem domain under investigation. One idea would be representing a service composition as a queue of services, i.e., a permutation of atomic services indexes from the service repository $\mathcal{SR}$. Such a permutation can be interpreted. For example, Graphplan [20, 155] has been widely used to decode permutations into DAGs automatically [40, 47, 174, 192, 200]. However, permutations could lead to conflicts in learning the knowledge of service index positions for composite services because different permutations could often be decoded into an identical DAG-based composite service. To reduce the chances of conflicts, we aim to efficiently produce a unique and more reliable permutation for the identical DAG-based composite service. In other words, a bi-directional map is ensured between permutations and DAGs, see details in Sect. 3.5.2.

To better trace promising searching areas, we need to consider the past good experience for constructing a new NHM in future generations. Therefore, an archive technique is introduced to reserve half the population-size of elite individuals to the next generation. Meanwhile, reserved individuals can also significantly reduce the overall computation time for their evaluation in the future.

3.5.1 Outline of EDA-NHM

EDA-NHM is outlined in ALGORITHM 11. To begin with, we initialize the initial population $\mathcal{P}^0$ by randomly generating m permutations. Afterwards, each permutation will be evaluated based on its decoded DAG-based solution, $\mathcal{G}_k^g$. Note that each permutation can be interpreted as a DAG through the use of a forward decoding technique (i.e., Graphplan [20, 155]). Next, we encode each individual in $\mathcal{P}^0$ with a different permuta-

ALGORITHM 11. EDA-NHM for the WSC-CQ problem.

Input : composition task T, service repository $\mathcal{SR}$
Output: a composite service

1: Initialize $\mathcal{P}^0$ with m random permutations as a Π_k^g (where $k = 1, \ldots, m$);
2: Evaluate each Π_k^g by decoding it into a DAG-based solution, $\mathcal{G}_k^g$;
3: Replace each solution in $\mathcal{P}^0$ with a corresponding encoded $\Pi^{\star g}_k$;
4: Create an archive with the top $\frac{m}{2}$ best solutions in $\mathcal{P}^0$;
5: Generate $\mathcal{NHM}^0$ from the top $\frac{m}{2}$ best solutions in $\mathcal{P}^0$;
6: Set generation counter $g \leftarrow 0$;
7: **while** $g <$ *maximum number of generations* **do**
8: Populate $\mathcal{P}^{g+1}$ with $\frac{m}{2}$ solutions Π_k^{g+1} sampled from $\mathcal{NHM}^g$;
9: Form $\mathcal{P}^{g+1}$ with additional $\frac{m}{2}$ solutions from the archive;
10: Empty the archive;
11: Evaluate each Π_k^{g+1} by decoding it into a DAG-based solution, $\mathcal{G}_k^{g+1}$;
12: Replace each solution in $\mathcal{P}^{g+1}$ with a corresponding encoded $\Pi^{\star g+1}_k$;
13: Update the archive with the top $\frac{m}{2}$ best solutions in $\mathcal{P}^{g+1}$;
14: Generate $\mathcal{NHM}^{g+1}$ from the top $\frac{m}{2}$ best solutions in $\mathcal{P}^{g+1}$;
15: Set $g \leftarrow g + 1$;
16: Let sol^{opt} be the best solution over all the generations;

tion $\Pi^{\star g}_k$, which is the new presentation proposed in EDA-NHM algorithm (see details in Sect. 3.5.2). In Step 4 and 5, based on the fitness value, only the top $\frac{m}{2}$ best solutions are used to generate $\mathcal{NHM}^g$ (following the basic structure of EDA in [168]) and reserved into an archive. The iterative part (Step 7 to 15) will be repeated until the maximum number of generations is reached. During each iteration, NHBSA [168] is employed to sample $\frac{m}{2}$ solutions from $\mathcal{NHM}^g$ for the next population $\mathcal{P}^{g+1}$, see details in Sect. 3.5.3.

Afterwards, we form the $\mathcal{P}^{g+1}$ with additional solutions reserved from the archive. This archive will be cleared up for the use of the next generation. Step 11 to 14 are similar to Step 2 to 5. Lastly, we return the best solution sampled over all the generations. In a nutshell, our proposed method introduces a novel permutation-based representation $\Pi^{\star g}_k$ that requires an decoding and encoding process, and an archive technique that reserves $\frac{m}{2}$ elite solutions in each generation.

3.5.2 A novel permutation-based representation

Composite services are commonly represented as DAGs [41, 45]. Let $\mathcal{G} = (V, E)$ be a DAG-based composite service from $Start$ to End, where nodes correspond to the services and edges correspond to the QoSM between two connected services (Often, V does not contain all services in $\mathcal{SR}$). As discussed previously, we present composite services as permutations, and we ensure a bi-directional map between permutations and DAGs. Let $\Pi^{\star} = (\pi_0, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_{n-1})$ be a permutation, elements of which are $\{0, \ldots, t, t+1, \ldots, n-1\}$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. Particularly, $\{0, \ldots, t\}$ are service indexes (i.e., id number) of the component services in the corresponding $\mathcal{G}$, and is sorted based on the longest distance from $Start$ to every component services of $\mathcal{G}$. While $\{t+1, \ldots, n-1\}$ be indexes of remaining services in $\mathcal{SR}$ are not utilized by $\mathcal{G}$. Note that the longest distance between the services and $Start$ can ensure that all the inputs of these services can be fulfilled. Meanwhile, multiple services could be associated with the same distance to $Start$. These services are sorted randomly because no information can be extracted from the DAG to determine the order of these services. We use $\Pi^{\star g}_k$ to present the k^{th} (out of m, m is population size) composite service, and $\mathcal{P}^g = [\Pi^{\star g}_0, \ldots, \Pi^{\star g}_k, \ldots, \Pi^{\star g}_{m-1}]$ to represent a population of composite services at generation g. To summarize a process of producing the new permutations in EDA-NHM, we outline this process in Fig. 3.2.

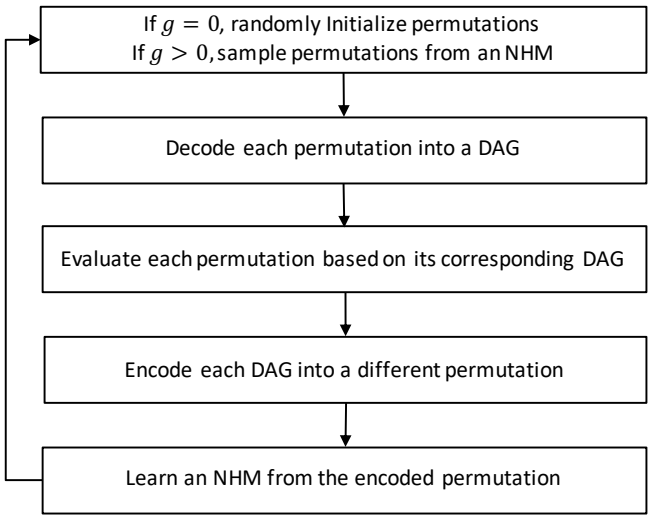


Fig. 3.2: A process of generating composite services as permutations.

Example 2. *Let us consider a composition task* $T = (\{a, b\}, \{e, f\})$ *and a service repository* $\mathcal{SR}$ *consisting of six atomic services.* $S_0 = (\{e, f\}, \{g\}, QoS_{S_0})$, $S_1 = (\{b\}, \{c, d\}, QoS_{S_1})$, $S_2 = (\{c\}, \{e\}, QoS_{S_2})$, $S_3 = (\{d\}, \{f\}, QoS_{S_3})$, *and* $S_4 = (\{a\}, \{h\}, QoS_{S_4})$. *The two special services* $Start = (\emptyset, \{a, b\}, \emptyset)$ *and* $End = (\{e, f\}, \emptyset, \emptyset)$ *are defined by a given composition task* T. *Fig. 3.2 illustrates an example of producing permutation* $[1, 2, 3, 0, 4]$ *from a DAG, which is decoded from a given permutation* $[4, 1, 0, 2, 3]$.

As an example in Fig. 3.3, take a permutation as $[4, 1, 0, 2, 3]$. This service index queue is decoded into a $\mathcal{G}$, representing a composite service that satisfies the composition task T. Note that, such a decoding process can ensure the functional validity of a solution. Afterwards, the $\mathcal{G}$ is mapped to a permutation $\Pi^{\star} = [1, 2, 3 \mid 0, 4]$. Herein, each position on the left side of $\mid$ corresponds to a service discovered by a BFS on $\mathcal{G}$ from $Start$. While the right side corresponds to the remaining atomic services in $\mathcal{SR}$, but not in $\mathcal{G}$. Note that $\mid$ is just displayed for the courtesy of the reader, rather than being part of the permutation-based representation. Furthermore, we also permit the encoding $[1, 2, 3 \mid 4, 0]$, as no information can be extracted from $\mathcal{G}$ to determine the order of 0 and 4.

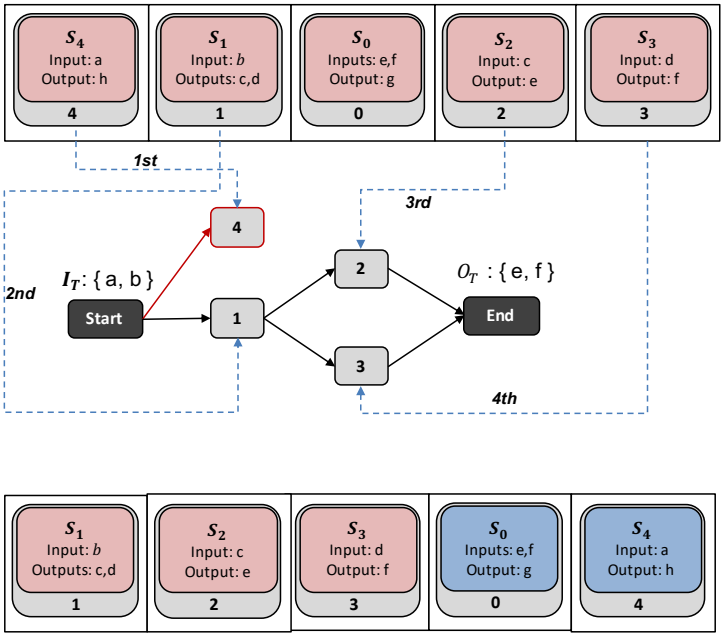


Fig. 3.3: A different permutation produced by a decoding and encoding process.

A permutation-based population $\mathcal{P}^g$ can be created with m permutation-based solutions. Consider $m = 6$, $\mathcal{P}^g$ could be represented as follows:

$$\mathcal{P}^g = \begin{bmatrix} \Pi^{\star g}_0 \\ \Pi^{\star g}_1 \\ \Pi^{\star g}_2 \\ \Pi^{\star g}_3 \\ \Pi^{\star g}_4 \\ \Pi^{\star g}_5 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & | & 0 & 4 \\ 0 & | & 1 & 2 & 3 & 4 \\ 0 & | & 1 & 2 & 3 & 4 \\ 4 & 3 & | & 0 & 1 & 2 \\ 4 & 3 & | & 0 & 1 & 2 \\ 2 & 1 & 3 & | & 0 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 0 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 0 & 1 & 2 \\ 4 & 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 0 & 4 \end{bmatrix}$$

3.5.3 Application of NHM Construction and NHBSA

This section presents a method for constructing a NHM and an application of node histogram-based sampling [168] for sampling solutions from the NHM. Using the novel permutation-based representation for candi-

date composition services, we are now able to apply this method to our problem.

The *node histogram matrix* (NHM) at generation g, denoted by $\mathcal{NHM}^g$, is an $n \times n$-matrix with entries $e^g_{i,j}$ as follows:

$$e^g_{i,j} = \sum_{k=0}^{m-1} \delta_{i,j}(\Pi^{\star g}_k) + \varepsilon \tag{3.7}$$

$$\delta_{i,j}(\Pi^{\star g}_k) = \begin{cases} 1 & \text{if } \pi_i = \text{j} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where $i, j = 0, 1, \ldots, n-1$, and $\varepsilon = \frac{m}{n-1} b_{ratio}$ is a predetermined bias. Roughly speaking, entry $e^g_{i,j}$ counts the number of times that service index π_i appears in position j of the permutation over all permutations in population $\mathcal{P}^g$.

Example 3. *Consider $\mathcal{P}^g$ in* **Example 2**, *the size of population m equals 6, the dimension size of each individual (i.e., permutation) n equals 5, and b_{ratio} = 0.2, we calculate $\mathcal{NHM}^g$ as follows:*

$$\mathcal{NHM}^g = \begin{bmatrix} 2.3 & 1.3 & 1.3 & 0.3 & 2.3 \\ 0.3 & 3.3 & 1.3 & 2.3 & 0.3 \\ 2.3 & 0.3 & 2.3 & 2.3 & 0.3 \\ 2.3 & 2.3 & 0.3 & 2.3 & 0.3 \\ 0.3 & 0.3 & 2.3 & 0.3 & 4.3 \end{bmatrix}$$

We pick up an element in the $\mathcal{NHM}^g$ as an example to demonstrate the meaning of each element in the NHM. For example, $e^g_{0,0}$(that equals 2.3) is made of integer and decimal parts: 2 and 0.3. The integer number 2 means that service index π_0 appears at the first position 2 times, while the decimal number 0.3 is a ε bias, calculated by $\frac{6}{5-1}0.2$.

Once we have computed $\mathcal{NHM}^g$, following ALGORITHM 12, we can sample a new candidate solution Π^{g+1}_k (with $k = 0, \ldots, m-1$) from a NHM^g for generation $g+1$. Particularly, NHBSA starts with sampling

ALGORITHM 12. NHBSA for sampling a permutation-based composite service [168]

Input : NHM^g
Output: Π_k^{g+1}

Generate a random position index permutation r[] of [0,1, ..., n-1];
Generate a candidate list C = $[0, 1, ..., n-1]$;
Set the position counter $p \leftarrow 0$;
while $p < n-1$ **do**
    Sample node x with probability $\frac{e^g_{r[p],x}}{\sum_{j \in C} e^g_{r[p],j}}$;
    Set $c[r[p]] \leftarrow x$ and remove node x from C;
    $p \leftarrow p + 1$;
$\Pi_k^{g+1} \leftarrow$ c[];
return Π_k^{g+1};

an element for a random position of a permutation with a probability calculated based on the elements of NHM^g. Then, it recursively continues sampling other elements for other positions in the permutation. Once a new permutation Π_k^{g+1} is returned, the same decoding part discussed in Sect. 3.5.2 will be performed on Π_k^{g+1} to produce a functionally valid composite service in the DAG form.

3.6 The EDA-EHM Algorithm

In this section, we introduce our EDA-EHM method for the WSC-CQ problem. We first outline our EDA-NHM method in Sect. 3.6.1. Subsequently, we discuss three ideas behind this approach: (1) a proposed ontology-based querying technique for querying service dependency in Sect. 3.6.2, (2) an application of EHM for learning service dependency in Sect. 3.6.3, and (3) a proposed sampling technique for building composite services in Sect. 3.6.4.

As presented in Sect. 3.5, EDA-NHM learns the distribution of each service index in $\mathcal{SR}$ at each absolute position of a permutation. However, many service indexes at the end of sampled permutations do not contribute to constructing DAGs. Therefore, we should focus on useful services (i.e., component services of DAGs), which can be captured by a set of service dependencies that forms DAGs. Meanwhile, distribution on the set of service dependencies can be easily presented in EHM. Consequently, we suggest learning EHM from promising composite services. In particular, we start with proposing an ontology-based querying technique for querying dependencies of services in $\mathcal{SR}$ (see details in Sect. 3.6.2). This technique can be utilized to initial valid entries of EHMs. Afterwards, we construct an EHM from a set of promising DAG-based composite services represented by a set of services dependencies (see details in Sect. 3.6.3).

Furthermore, to efficiently sample functionally valid DAG-based composite services from EHM, we propose a GEHBGSA by effectively using some information, such as services dependencies and service layers, to guide the sampling process (see details in Sect. 3.6.4).

3.6.1 Outline of EDA-EHM

We outline our proposed EDA-EHM in ALGORITHM 13. We start with labeling an ontology, $\mathcal{O}$, with task-related web services using ALGORITHM 14 in Sect. 3.6.2, Next, we initialize a population $\mathcal{P}^0$ with m DAG-based candidate solutions $\mathcal{G}_k^g$ by decoding m random permutations Π_k^g (where $k = 1, \ldots, m$) into DAGs. Those candidate solutions are evaluated using Eq. 3.5. Then, the top half best solutions are stored in an archive, and are also used to generate a $\mathcal{EHM}^g$ (where $g = 0$), see details in Sect. 3.6.3. Step 6 to 13 will be repeated until the maximum number of generations is reached: we sample $\frac{m}{2}$ new DAG-based solutions from $\mathcal{EHM}^g$ using our proposed GEHBGSA (see details in Sect. 3.6.4). These newly sampled solutions are combined with the archive to form the next

ALGORITHM 13. EDA-EHM for the WSC-CQ problem.

Input : composition task T, service repository $\mathcal{SR}$ and $g = 0$
Output: a near-optimal composite service $\mathcal{G}^{opt}$

1: Label $\mathcal{O}$ with task-related web services using ALGORITHM 14;
2: Initialize $\mathcal{P}^g$ with m DAG-based solutions, each solution represented as a $\mathcal{G}_k^g$ (where $k = 1, \ldots, m$);
3: Evaluate each solution in $\mathcal{P}^g$ using Eq. 3.5;
4: Create an archive with the top $\frac{m}{2}$ of best solutions in $\mathcal{P}^0$;
5: Generate $\mathcal{EHM}^g$ from the top $\frac{m}{2}$ of best solutions in $\mathcal{P}^0$;
6: **while** $g <$ *maximum number of generations* **do**
7: Sample $\frac{m}{2}$ $\mathcal{G}_k^{g+1}$ from $\mathcal{EHM}^g$ using ALGORITHM 15;
8: Populate $\mathcal{P}^{g+1}$ with the sampled solutions and additional $\frac{m}{2}$ solutions from the archive;
9: Empty the archive;
10: Evaluate each solution in $\mathcal{P}^{g+1}$ using Eq. 3.5;
11: Update the archive with the top $\frac{m}{2}$ of best solutions in $\mathcal{P}^{g+1}$;
12: Generate $\mathcal{EHM}^{g+1}$ from the top $\frac{m}{2}$ of the best solutions in $\mathcal{P}^{g+1}$;
13: Set $g \leftarrow g + 1$;
14: Let $\mathcal{G}^{opt}$ be the best solution over all the generations;

population $\mathcal{P}^{g+1}$, which will be evaluated and selected to learn $\mathcal{EHM}^{g+1}$. The archive is cleared up for the next generation. At last, the best solution, $\mathcal{G}^{opt}$ over all the generations is returned. In summary, we propose a way of learning EHM from high-quality solutions evolved by EDA and a novel sampling technique, GEHBGSA, for sampling functionally valid DAG-based composite services.

3.6.2 Discovery of Service Dependency

Service dependency represents a relationship between two services (e.g., service S_j and service S_i) that are determined by the existence of robust

causal links between these two services. In other words, one service, S_j, can be either partially or fully satisfied by its predecessor S_i, denoted as $S_i \rightarrow S_j$.

To identify service dependencies regarding each service, we propose an ontology-based querying technique to efficiently find all the predecessor of any service in $\mathcal{SR}$. We first create labels for concept nodes of a taxonomy tree in $\mathcal{O}$ with task-related services using ALGORITHM 14. In this algorithm, we mark each tree node with two sets of services, i.e., O_C and I_C, where service dependencies can be established from services in O_C to services in I_C. We can query the predecessors of one service S by looking at a union of O_C on the concept nodes, which are the concepts related to the inputs of S. We will demonstrate this technique in Example 4.

ALGORITHM 14. Labeling Services on a taxonomy tree in $\mathcal{O}$

Input : $\mathcal{SR}$ and $\mathcal{O}$
Output: labeled $\mathcal{O}$

```
1:  foreach concept C in taxonomy tree in O do
2:      label two empty service set I_C and O_C in relation to inputs and
          output;
3:  foreach S in SR do
4:      foreach I_S of S do
5:          find concepts C of I_S on taxonomy tree in O;
6:          foreach C in C ∪ its child concepts do
7:              put S to I_C of C;
8:      foreach O_S of S do
9:          find concepts C of O_S on taxonomy tree in O;
10:         foreach C in C ∪ its parent concepts do
11:             put S to O_C of C;
12: return labeled O;
```

Example 4. *Suppose we have a service repository $\mathcal{SR}$ consisting of a single service $S_0 = (\{c, d\}, \{e\}, QoS_{S_0})$. Let us consider a* service request *$T = (\{a, b\}, \{i\})$. Two special services $Start = (\emptyset, \{a, b\}, \emptyset)$ and $End = (\{i\}, \emptyset, \emptyset)$ are defined by the given composition task T. Concepts related to a, b, c, d, e, and i are Dog, Artificial Data, Data, Canine, Animal Robot and Robot respectively. These concepts are represented and labeled with services in an taxonomy tree in Fig. 3.4. The predecessor of End is S_0, which is a service in O_{Robot} of concept $Robot$ related to i. The predecessor of S_0 is $Start$, which is a service in an union of O_{Data} and O_{Canine} related to c and d respectively.*

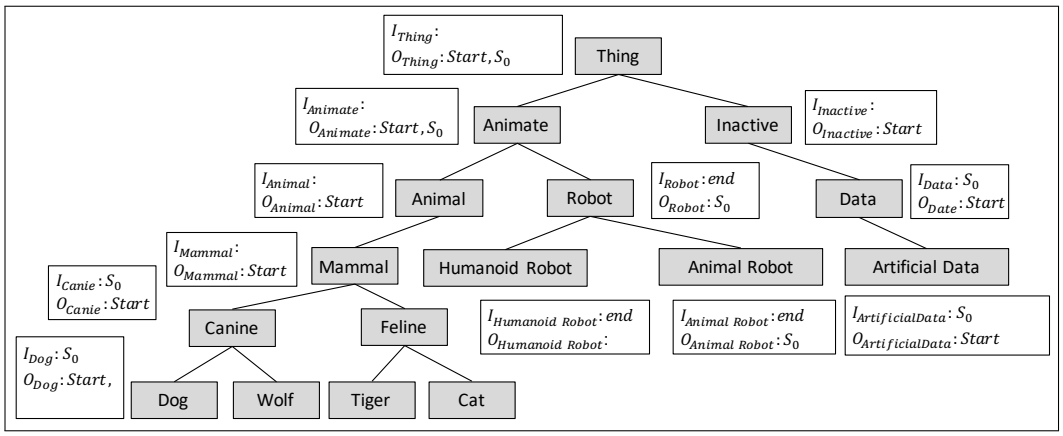


Fig. 3.4: An example of labeled $\mathcal{O}$.

3.6.3 Application of EHM Construction

Let $\mathcal{D} = \{S_i \rightarrow S_j\}$ be the set of all existing service dependencies among all possible pairs of services in $\mathcal{SR}$ (Note that the size of $\mathcal{D}$ is not big because only task-related services are considered, and not every pair of services has service dependencies). Let $\mathcal{G}$ be a DAG-based composite service consisting of a set of service dependencies, satisfying $\mathcal{G} \subset \mathcal{D}$. Consequently, $\mathcal{G}_k^g$ represents the k^{th} $(0 \leq k < m)$ DAG-based composite service, and $\mathcal{P}^g = [\mathcal{G}_0^g, \ldots, \mathcal{G}_k^g, \ldots, \mathcal{G}_{m-1}^g]$ is represented as a population of composite

services at generation g. We will use an example to demonstrate the construction of an EHM based on a set of $\mathcal{G}$ below.

Example 5. *Suppose we have a service repository $\mathcal{SR}$ consisting of five services $S_0 = (\{c, d\}, \{e\}, QoS_{S_0})$, $S_1 = (\{a\}, \{f, g\}, QoS_{S_1})$, $S_2 = (\{a, b\}, \{h\}, QoS_{S_2})$, $S_3 = (\{f, h\}, \{i\}, QoS_{S_3})$ and $S_4 = (\{a\}, \{f, g, h\}, QoS_{S_4})$. Let us consider the same* service request $T = (\{a, b\}, \{i\})$ *as in* **Example 4**.

A permutation-based population $\mathcal{P}^0$ can be created with m DAG-based solutions. Consider $m = 6$, $\mathcal{P}^0$ could be represented as follows:

$$\mathcal{P}^0 = \begin{bmatrix} \mathcal{G}_0^0 \\ \mathcal{G}_1^0 \\ \mathcal{G}_2^0 \\ \mathcal{G}_3^0 \\ \mathcal{G}_4^0 \\ \mathcal{G}_5^0 \end{bmatrix} = \begin{bmatrix} \{Start \to S_1, Start \to S_2, S_1 \to S_3, S_2 \to S_3, S_3 \to End\} \\ \{Start \to S_0, S_0 \to End\} \\ \{Start \to S_0, S_0 \to End\} \\ \{Start \to S_4, S_4 \to S_3, S_3 \to End\} \\ \{Start \to S_4, S_4 \to 3, S_3 \to End\} \\ \{Start \to S_1, Start \to 2, S_1 \to S_3, S_2 \to S_3, S_3 \to End\} \end{bmatrix}$$

The *edge histogram matrix* at generation g (denoted by $\mathcal{EHM}^g$) is a matrix with entries $e_{i,j}^g$ (where $i, j = Start, 0, 1, \cdots, m-1, End$) as follows:

$$e_{i,j}^g = \begin{cases} \sum_{k=0}^{m-1} \delta_{i,j}(\mathcal{G}_k^g) + \varepsilon_{i,j} & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$$\delta_{i,j}(\mathcal{G}_k^g) = \begin{cases} 1 & \text{if } S_i \to S_j \in \mathcal{G}_k^g \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

$$\varepsilon_{i,j} = \begin{cases} \frac{b_{ratio}}{|\mathcal{D}|} \sum_{k=0}^{m-1} |\mathcal{G}_k^g| & \text{if } S_i \to S_j \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

Herein, b_{ratio} is a predetermined constant (called bias ratio), $|\mathcal{G}_k^g|$ denotes the number of service dependencies in $\mathcal{G}_k^g$, while $|\mathcal{D}|$ denotes the number of all service dependencies in $\mathcal{SR}$. Roughly speaking, entry $e_{i,j}^g$ counts how

often a service dependency $S_i \rightarrow S_j$ occurs in all composition services in population $\mathcal{P}^g$.

Consider the $\mathcal{P}^0$ in **Example 5**, where the size of $\mathcal{P}^0$, m, equals 6, and the length of permutation, n, equals 5, b_{ratio} = 0.2, $|\mathcal{D}| = 9$, and $\sum_{k=0}^{m-1} |\mathcal{G}_k^g| = 20$. We calculate $\mathcal{EHM}^g$ as follows:

$$\mathcal{EHM}^0 = \begin{array}{c} \text{i} \setminus \text{j} \\ \textit{Start} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ \textit{End} \end{array} \begin{array}{c} \begin{array}{ccccccc} \textit{Start} & 0 & 1 & 2 & 3 & 4 & \textit{End} \end{array} \\ \left[\begin{array}{ccccccc} 0 & 2.44 & 2.44 & 2.44 & 0 & 2.44 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.44 \\ 0 & 0 & 0 & 0 & 2.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4.44 \\ 0 & 0 & 0 & 0 & 2.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

3.6.4 GEHBGSA for sampling

The proposal of GEHBGSA is inspired by the Edge Histogram-Based Sampling Algorithm (EHBSA) [169]. Instead of sampling permutations using EHBSA, GEHBGSA aims to sample functionally valid DAG-based composition services from a constructed EHM. To effectively sample DAGs, we utilize some useful information to guide the sampling process in GEHBGSA. This information includes: only row indexes of non-zero entries in $\mathcal{EHM}^g$ are to be sampled, and layer information is used to verify sampled predecessors for preventing cycles in solutions. GEHBGSA builds a DAG backwards as it has been suggested that backward graph building has its advantage over the forward graph building since it does not create dangling services (i.e., services do not contribute the required output of a service request) [47]. This sampling algorithm is summarized in ALGORITHM 15.

In ALGORITHM 15, we first initialize a DAG-based solution $\mathcal{G}$ with an empty set of service dependencies, and a set of service, $SerSet$, with End.

ALGORITHM 15. GEHBGSA for sampling a functionally valid DAG-based composite service

Input : $\mathcal{EHM}^g$, layers $\mathcal{L}_p$, where $p = 0, \ldots, q$
Output: a composite service $\mathcal{G}$

1: initial $\mathcal{G} = \{\ \}$ and $SerSet = \{End\}$;
2: **foreach** S_j *in* $SerSet$ **do**
3: **if** $SerSet$ *does not only contains* $start$ *and* S_j *is not fully satisfied* **then**
4: identify $\mathcal{L}_p\ s.t.\ S_j \in \mathcal{L}_p$;
5: determine a set SC of row indexes for non-zero entries in $\{e^g_{x,j}\}^{n-1}_{x=0}$;
6: **while** *inputs of* S_j *is not fully satisfied and* SC *is not empty* **do**
7: sample one predecessor x with probability $\frac{e^g_{x,j}}{\sum_{i \in SC} e^g_{i,j}}$;
8: identify $\mathcal{L}_{p'}\ s.t.\ S_x \in \mathcal{L}_{p'}$;
9: **if** $p' \leq p$ *and any unsatisfied input of* S_j *is fulfilled by* S_x **then**
10: put $S_x \rightarrow S_j$ into $\mathcal{G}$;
11: **foreach** $S_{j^\star}$ *in* $SerSet$ **do**
12: identify $\mathcal{L}_{p^\star}\ s.t.\ S_{j^\star} \in \mathcal{L}_{p^\star}$;
13: **if** $p' \leq p^\star$ *and any unsatisfied input of* $S_{j^\star}$ *is fulfilled by* S_x **then**
14: put $S_x \rightarrow S_{j^\star}$ into $\mathcal{G}$;
15: add S_x to $SerSet$;
16: remove x from SC;
17: remove S_j from $SerSet$;
18: **return** $\mathcal{G}$;

For $SerSet$, the satisfactions on the inputs of services in $SerSet$ are required to be checked. The following steps (i.e., Step 2 to 17) are repeated if

$SerSet$ does not only contains $Start$ or any service in $SerSet$ are not fully satisfied: for each service S_j in $SerSet$, we identify its layer $\mathcal{L}_p$. Meanwhile, we initialize a set, SC, consisting of row indexes of non-zero entries in $\{e^g_{x,j}\}_{x=0}^{n-1}$. Afterwards, another repeated sampling process is used to produce predecessors of S_j until S_j is fully satisfied (Step 6 to 16). During the sampling, let S_x be a sampled service, if the layer that contains S_x is ahead of or the same to that of S_j, and any unsatisfied inputs of S_j can be fulfilled by S_x (Step 9), we create a dependency $S_x \rightarrow S_j$ and put it into $\mathcal{G}$ (Step 10). Meanwhile, we also check the satisfaction of other services in $SerSet$ in the similar way that we create the dependency with S_j (Step 11 to 14). Later on, the sampled predecessor S_x is added to $SerSet$ and sampled x is removed from SC. Once S_j is fully satisfied, we remove it from $SerSet$, and repeat creating dependencies for newly added services in $SerSet$. Lastly, a $\mathcal{G}$ is returned after the iteration.

Consider $\mathcal{EHM}^0$ calculated in in **Example 5**, we present a process of sampling a candidate solution in Fig. 3.5 using GEHBGSA.

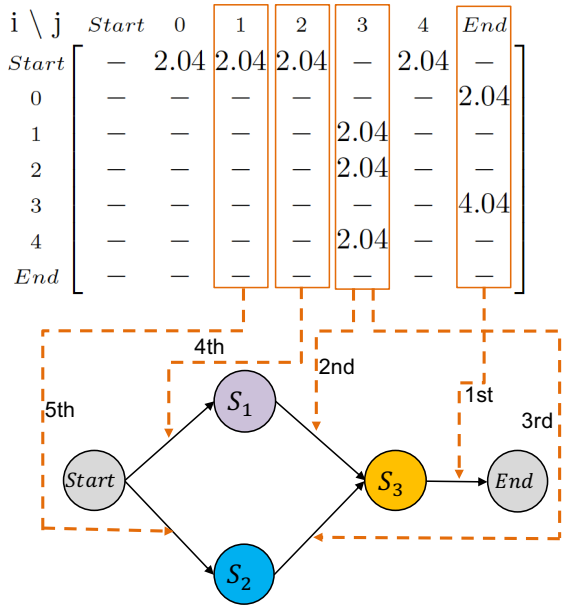


Fig. 3.5: An example of a DAG generated by GEHBGSA.

3.7 Experimental Evaluation

In this section, we conduct an experiment to study the performance of our proposed EDA-based methods for the WSC-CQ problem. Particularly, we compare EDA-EHM and EDA-NHM with the state-of-the-art automated service composition approaches discussed in Sect. 3.1, i.e., PSO [175], FL [47], PathSearch [32].

The population size is set to 200, the number of generations equals 100. The bias ratio, b_{ratio}, is set to 0.0002 for EDA-NHM and EDA-EHM. For other settings of the competing methods, we strictly follow their settings reported in their papers. In FL, the crossover rate is set to 0.95, and the mutation rate is set to 0.05. In PathSearch, the parameter K (i.e., number of services considered in the path construction at each step) associated with this algorithm is set to 7, which maximizes the performance of PathSearch in their paper. The weights of the fitness function in Eq. (3.5) are simply configured to balance the QoSM and QoS. In particular, we set both w_1 and w_2 to 0.25, and w_3, w_4, w_5 and w_6 all to 0.125. More experiments have been conducted and show that all our methods work consistently well under different weight settings. The p of $type_{link}$ in Eq. (3.1) is determined by the preference of users, and is recommended as 0.75 for the plugin match according to [101].

We run the experiment with 30 independent repetitions for all the EC-based approaches. All the methods are run on a PC with an Intel Core i7-4770 CPU (3.4GHz) and 8 GB RAM. This hardware configuration will also be used for all the methods presented in our thesis.

3.7.1 Comparing EDA-NHM and EDA-EHM with PSO, FL, SearchPath

Comparison of the Fitness

We employ an independent-sample T-test with a significance level of 5% to verify the observed differences in performance concerning fitness value. In particular, we use a pairwise comparison to compare all competing approaches, and then the top performances are highlighted in a green color in Table 3.2. Note that those methods that consistently find the best-known solutions over 30 runs with 0 standard deviations are also marked as top performances. The pairwise fitness comparison results are summarized in Table 3.3, where *win/draw/loss* shows the scores of one method compared to all the others, and displays the frequency that this method outperforms, equals or is outperformed by the competing method. This testing and comparison methods are also used in Sect. 3.9.1.

One objective of the experiment is to evaluate the effectiveness of EDA-NHM and EDA-EHM comparing to PSO, FL and PathSearch. By analyzing the results in Table 3.2 and 3.3, we can conclude the observations about the effectiveness of these methods as follows:

Firstly, EDA-NHM consistently find composite services with the highest quality among all the methods. This observation indicates that EDA-NHM is more competent at improving the quality of composite services by effectively utilizing the knowledge via NHMs, compared to other methods.

In contrast, PathSearch achieves the worst performance in finding high-quality solutions. This is because PathSearch is designed to make a locally best choice over the K services at each step, towards a gradually built path-based composite service.

In addition, FL significantly outperforms EDA-EHM in finding high-quality solutions. This observation agrees with the findings in a recent study [47] on the representations of composite services, i.e., graph-based

representations are less effective, compared to permutation-based representations.

Furthermore, regarding two different EDA-based approaches, EDA-NHM achieves better performance compared to EDA-EHM methods in finding high-quality solutions. These findings indicate that considering service dependencies can be less effective than considering service positions for enhancing the effectiveness of EDA. This might be due to the performance of our service composition problem may depend mainly on absolute positions of services in permutations.

Lastly, we can observe that EDA-EHM can outperform PSO in finding high-quality solutions. Different from the permutation-based representation utilized in EDA-EHM, EDA-NHM and FL, a vector-based representation made of real numbers (ranges from 0 to 1) is utilized in PSO for representing particles' positions. Instead of optimizing the order of services indexes in a permutation, PSO indirectly optimized particle's positions (i.e., continuous search space), which are mapped to the permutations (i.e., discrete search space). Apparently, optimizing positions is less effective, compared to optimzing the permutations directly.

In summary, we sort all the competing approaches based on the effectiveness in descending order: EDA-NHM $>$ FL $>$ EDA-EHM $>$ PSO $>$ PathSearch. Apparently, EDA-NHM can be the most suitable algorithm when the design stage can be ensured with sufficient computation time for running algorithms.

Table 3.2: Mean fitness values for EDA-NHM and EDA-EHM in comparison to PSO, FL and PathSearch. (Note: the higher the fitness the better)

Dataset	EDA-NHM	EDA-EHM	PSO	FL	PathSearch
WSC08-1	0.613745 ± 0	0.613745 ± 0	0.610182 ± 0.003748	0.613439 ± 0.000693	0.607149
WSC08-2	0.756812 ± 0	0.756481 ± 0.00178	0.756779 ± 0.000175	0.756812 ± 0	0.597588
WSC08-3	$0.477802 \pm 5.2ee05$	0.476864 ± 0.000401	0.476086 ± 0.000528	0.477447 ± 0.000228	0.473133
WSC08-4	0.557815 ± 0	0.557607 ± 0.000546	0.557416 ± 0.000666	$0.55781 \pm 3.1e-05$	0.549285
WSC08-5	0.524492 ± 0.000415	0.523312 ± 0.0032	0.517912 ± 0.00544	0.523463 ± 0.002078	0.455233
WSC08-6	0.482671 ± 0.000139	0.480823 ± 0.000224	0.481723 ± 0.000505	0.482142 ± 0.000452	0.414705
WSC08-7	$0.523584 \pm 2.2ee05$	0.522734 ± 0.002348	0.516141 ± 0.00519	0.521694 ± 0.002248	0.471800
WSC08-8	0.497638 ± 0.000126	0.483836 ± 0.002077	0.489581 ± 0.003219	0.496347 ± 0.000861	0.445169
WSC09-1	0.647988 ± 0.002939	0.654776 ± 0.014186	0.648605 ± 0.004328	0.649612 ± 0.003688	0.597177
WSC09-2	0.517561 ± 0.012522	0.497438 ± 0.005009	0.506701 ± 0.011045	0.521424 ± 0.003011	0.464576
WSC09-3	0.583978 ± 0	$0.58395 \pm 8.4e-05$	0.583358 ± 0.001182	$0.583954 \pm 9.1e-05$	0.455317
WSC09-4	0.484315 ± 0.000104	0.484168 ± 0.000224	0.481741 ± 0.000985	0.483277 ± 0.000367	0.467477
WSC09-5	$0.484818 \pm 4e-05$	0.481967 ± 0.000463	0.480539 ± 0.001308	0.483278 ± 0.001185	0.468230

Table 3.3: Summary of statistical significance tests for fitness, where each column shows win/draw/loss score of an approach against others.

Dataset	Method	EDA-NHM	EDA-EHM	PSO	FL	PathSearch
WSC-08 (8 tasks)	EDA-NHM	-	0/3/5	1/7/0	0/2/6	0/0/8
	EDA-EHM	3/5/0	-	3/2/3	3/2/3	0/0/8
	PSO	7/1/0	3/2/3	-	8/0/0	0/0/8
	FL	6/2/0	2/3/3	0/0/8	-	0/0/8
	PathSearch	8/0/0	8/0/0	8/0/0	8/0/0	-
WSC-09 (5 tasks)	EDA-NHM	-	1/0/4	0/1/4	0/3/2	0/0/5
	EDA-EHM	4/0/1	-	1/0/4	2/2/1	0/0/5
	PSO	4/1/0	4/0/1	-	2/2/1	0/0/5
	FL	2/3/0	1/2/2	1/2/2	-	0/0/5
	PathSearch	5/0/0	5/0/0	5/0/0	5/0/0	-

Comparison of the Execution Time

The second objective of our experiment is to study the efficiency of EDA-NHM and EDA-EHM comparing to PSO, FL, and PathSearch. Table 3.4 shows the mean value of the execution time and the standard deviation for all EC-based approaches over 30 repetitions, and execution time consumed by PathSearch over one run. The pairwise comparison results for the execution time are summarized in Table 3.5. By analyzing the results in these tables, we can conclude the observations about the execution time of these methods as follows:

First, PathSearch requires the least execution time because this method prunes pre-stored service dependency graphs, on which it only searches K best services at each step, towards a gradually built path (i.e., a composite service). Despite of the highest efficiency achieved by PathSearch, the effectiveness of this method is the worst.

Second, among all EC-based approaches (excluding PathSearch), we can observe that EDA-EHM requires the lowest execution time. The finding agrees with our expectation that the proposed ontology-based query technique and GEHBGSA in NHM-EHM can contribute its outstanding efficiency.

Third, EDA-NHM consumes less execution time compared to PSO and FL. This might be due to two reasons: (1) solutions evolved by EDA-NHM are likely to have all useful services required to build a suitable DAG placed at the very front of the service queue, and (2) the archive utilized in EDA-NHM stores promising solutions, saving execution time from preventing many promising solutions to be evaluated.

In summary, we sort all the competing approaches based on the execution time in ascending order: PathSearch $>$ EDA-EHM $>$ EDA-NHM $>$ PSO $>$ GA. Despite the lowest execution time consumed by PathSearch, PathSearch may not be suitable for practical use since it presents the lowest effectiveness for producing high-quality composite services at the design stage.

Table 3.4: Mean execution time (in s) for EDA-NHM and EDA-EHM in comparison to PSO, FL and PathSearch. (Note: the shorter the time the better)

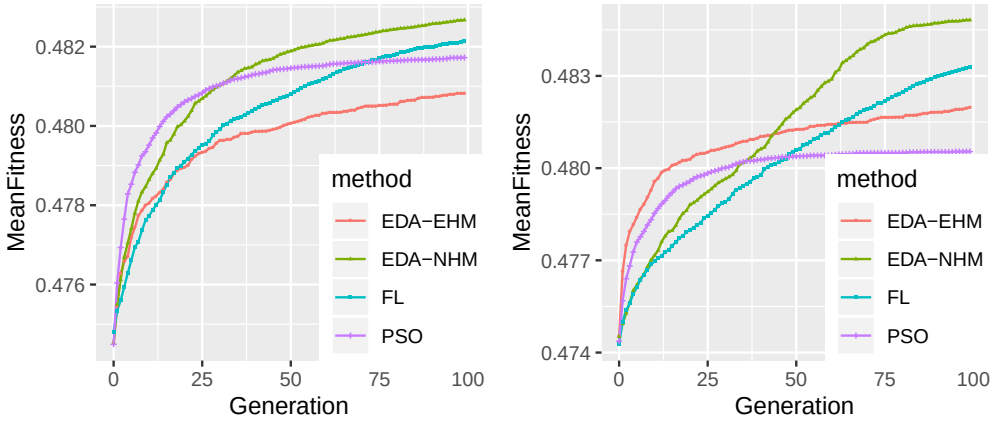
Dataset	EDA-NHM	EDA-EHM	PSO	FL	PathSearch
WSC08-1	72 ± 6	5 ± 0	111 ± 76	102 ± 11	3
WSC08-2	32 ± 4	5 ± 0	68 ± 48	21 ± 3	6
WSC08-3	1296 ± 137	29 ± 2	15789 ± 2602	12514 ± 1575	18
WSC08-4	38 ± 4	8 ± 1	211 ± 104	147 ± 36	6
WSC08-5	979 ± 96	18 ± 1	2549 ± 1517	3801 ± 1512	19
WSC08-6	13964 ± 1735	127 ± 65	33119 ± 12194	51287 ± 11561	101
WSC08-7	1181 ± 115	103 ± 64	4456 ± 2825	5499 ± 1526	90
WSC08-8	2677 ± 308	260 ± 28	6153 ± 1951	10931 ± 1667	273
WSC09-1	62 ± 7	8 ± 1	126 ± 139	65 ± 12	7
WSC09-2	2204 ± 378	141 ± 13	3652 ± 1516	4081 ± 1433	138
WSC09-3	819 ± 65	205 ± 21	2198 ± 2038	1713 ± 394	233
WSC09-4	70105 ± 6772	1192 ± 161	85813 ± 37895	176152 ± 46321	1154
WSC09-5	11117 ± 1150	969 ± 79	14807 ± 5605	29991 ± 4867	915

Table 3.5: Summary of statistical significance tests for execution time (in s), where each column shows win/draw/loss score of an approach against others.

Dataset	Method	EDA-NHM	EDA-EHM	PSO	FL	PathSearch
WSC-08 (8 tasks)	EDA-NHM	-	8/0/0	0/0/8	0/1/7	8/0/0
	EDA-EHM	0/0/8	-	0/0/8	0/0/8	7/0/1
	PSO	8/0/0	8/0/0	-	2/4/2	8/0/0
	FL	7/1/0	8/0/0	2/4/2	-	8/0/0
	PathSearch	0/0/8	1/0/7	0/0/8	0/0/8	-
WSC-09 (5 tasks)	EDA-NHM	-	5/0/0	3/2/0	1/2/2	5/0/0
	EDA-EHM	0/0/5	-	0/0/5	0/0/5	5/0/0
	PSO	5/0/0	5/0/0	-	1/2/2	5/0/0
	FL	5/0/0	5/0/0	2/2/1	-	5/0/0
	PathSearch	0/0/5	0/0/5	0/0/5	0/0/5	-

Comparison of the Convergence Curve

The third objective of our experiment is to study the convergence curves of EDA-NHM, EDA-EHM, FL, and PSO. We have used WSC08-6 and WSC09-5 as two examples to illustrate the performance of all the competing methods. Note that WSC-09-5 is a more challenging task than WSC08-06 as its search space is much bigger.



(a) WSC-08-6 (b) WSC-09-5

Fig. 3.6: A comparison of the convergence curves of EDA-NHM, EDA-EHM, PSO, FL over execution time on WSC08-6 (the left) and WSC09-5 (the right).

Fig. 3.6 exhibits the evolution of the mean fitness value of the best solution found along the generation over 30 independent runs for EDA-NHM, EDA-EHM, FL, and PSO. For both composition tasks, we observe a significant increase in the fitness value towards the optimum over all the methods. These methods eventually reach different levels of plateaus, given the same number of evaluations.

For the more challenging composition task (WSC09-5), EDA-NHM and FL converge much slower against EDA-EHM and PSO. Although EDA-EHM and PSO reach plateaus much faster than EDA-NHM and FL, EDA-NHM and FL eventually reach much higher plateaus.

For the less challenging composition task (WSC08-6), EDA-NHM and PSO start to converge faster at the early stage than EDA-EHM and FL, but EDA-NHM and FL eventually outperform EDA-EHM and PSO methods.

For both WSC08-6 and WSC09-5, EDA-NHM consistently reaches the highest plateaus among all the competing methods.

3.8 A Memetic Algorithm Based on EDA-NHM

In this section, we present our memetic algorithms based on EDA-NHM for the WSC-CQ problem. We start by giving an overview of our memetic algorithms. Subsequently, we will discuss a selection method based on a uniform distribution schema and stochastic local search operators in Sect. 3.8.2 and Sect. 3.8.3, respectively. In our memetic algorithms, several key ideas are jointly proposed to build our algorithms:

1. As discussed in Sect. 3.5.2, EDA-NHM represents composite services as permutations, which are mapped from their corresponding DAG-based form. Such permutations enable a suitable distribution to be learned via NHM and new promising permutations to be sampled from the learned NHM. Apart from that, it is straightforward to define the neighbourhood on permutations by so-called swap operators [47]. As discussed previously, the complexity of such a swap over a permutation of length n is $n - 1$ in [44]. Although this complexity is not high, this swap operator cannot effectively exploit the neighbours of a permutation. This is because many produced neighbours can be decoded into the identical DAG. Hence, we will propose more effective swap operators based on the domain knowledge of service composition. These swap operators aim to effectively exploit neighbours, which are likely to be decoded into different DAGs.

2. To significantly decrease the computation time of the local search procedure, it is crucial to select a restricted number of suitable can-

didate solutions. As we know, fitness can present the importance of different candidate solutions. Moreover, the local search should be performed on solutions with distinctive importance. Therefore, we strategically group candidate solutions based on their fitness values according to a uniform distribution scheme, which allows a random candidate solution from each group to be selected for the local search.

3. It is not efficient to exhaustively explore the whole neighbours in the conventional local search [47]. Instead, stochastically searching the neighbouring solutions not only can significantly reduce computation cost without exploring the whole neighbours but can also escape the local optimal easily using the randomness [189]. Therefore, we introduce a stochastic local search strategy to effectively and efficiently exploit the neighbourhood of the selected candidate composite services.

3.8.1 Outline of the Memetic Algorithm

Our memetic algorithm follows EDA-NHM in ALGORITHM 11 with an additional local search procedure outlined in ALGORITHM 16. This procedure is performed between Step 3 and 4, and between Step 13 and 14 in ALGORITHM 11. In a local search procedure, we apply a local search to a restricted number of suitable candidate solutions, which are selected via a fitness uniform selection scheme over the current population (see details in Sect. 3.8.2). Furthermore, for each selected solution, a stochastic local search operator is employed to create new permutations as its neighbours, where the best neighbour is identified based on the fitness value (see details in Sect. 3.8.3).

This local search procedure, illustrated in ALGORITHM 16, takes three inputs: the gth population $\mathcal{P}^g$, the number of selected individuals for local search n_{set}, and the number of neighbours n_{nb}. In this algorithm, we start

by selecting a small fixed number n_{set} of candidate solutions to form a subset $SelectedIndiSet$ of the current population $\mathcal{P}^g$ using ALGORITHM 17. The local search is performed on the solutions in $SelectedIndiSet$. For each solution Π in $SelectedIndiSet$, we produce n_{nb} neighbours from Π by local search, and then we identify the best neighbour Π_{best} from the produced neighbours. Consequently, we replace the solution Π with the best neighbour Π_{best}. Eventually, we return an updated $\mathcal{P}^g$.

ALGORITHM 16. A local search procedure

Input : $\mathcal{P}^g$, n_{nb} and n_{set}
Output: an updated $\mathcal{P}^g$

1: Select a small number n_{set} of individulals to form a subset $SelectedIndiSet$ of $\mathcal{P}^g$ using ALGORITHM 17;
2: **for each** Π *in* $SelectedIndiSet$ **do**
3: Generate a size n_{nb} of neighbours from Π by local search;
4: Identify the neighbour Π_{best} with the highest fitness;
5: replace Π with Π_{best};
6: **return** an updated $\mathcal{P}^g$;

3.8.2 Application of Uniform Distribution Schema

Two types of selection schemes (i.e., random selection scheme and statistical scheme) have been studied for selecting suitable individuals for local search [34]. The random selection scheme is a primary selection method, where a local search is potentially applied to all individuals with a predefined probability. However, it can be less effective as it does not assign local search to the most suitable candidate solutions, and it is more time-consuming when the population size is huge. This statistical scheme is capable of choosing suitable individuals, which can be determined based on the statistical information of the current population.

ALGORITHM 17. Fitness uniform selection scheme

Input : $\mathcal{P}^g$ and n_{set}

Output: selected solutions $SelectedIndiSet$

1: $SelectedIndiSet \leftarrow \{\}$;
2: Sort $\mathcal{P}^g$ in descending order based on the fitness;
3: Put the first individual in $\mathcal{P}^g$ into $SelectedIndiSet$;
4: Calculate fitness range for $n_{set} - 1$ groups based on a uniform $interval$ between $maxfitness$ and $minfitness$;
5: Assign each permutation in $\mathcal{P}^g$ to $n_{set} - 1$ groups based on the fitness value;
6: Random select one permutation from each group and put it in $SelectedIndiSet$;
7: **return** $SelectedIndiSet$;

Our selection scheme, inspired by [78], is proposed based on the statistical information to select a restricted number of suitable individuals based on the idea, discussed at the beginning of Sect. 3.8. This selection scheme is presented in ALGORITHM 17. This algorithm applied a local search to a set of selected individuals $SelectedIndiSet$. The size of $SelectedIndiSet$, n_{set}, is a pre-defined parameter. $SelectedIndiSet$ consists of one elite individual and $n_{set} - 1$ individuals from $n_{set} - 1$ groups of individuals in each generation. Particularly, we calculate a fitness interval based on the maximal fitness value, $maxfitness$, and minimal fitness value, $minfitness$, of the current population $\mathcal{P}^g$. Therefore, the population is divided into $n_{set}-1$ groups based on the calculated fitness interval. Consequently, each group represent distinct importance, and individuals in a group represent similar importance. Note that, for every generation, the actual number of selected individuals for local search could be less than n_{set}, because no individuals could fall into one group.

3.8.3 Stochastic Local Search Operators

To investigate an appropriate structure of neighbourhood, suitable local search operators must be proposed by utilizing domain knowledge to guide swap operators on permutations. We can then repeatedly assign these local search operators to $SelectedIndiSet$ for exploring their neighbouring solutions. Apart from that, to reduce the computation time and escape local optimal easily, a random subset of the entire large neighbourhood is explored by performing stochastic local search. Based on the proposed novel permutation-based representation (also called a tidy-up permutation) in Sect. 3.5.2, our swap operators inspired by the swap operator in [47] are developed. In particular, we propose four different stochastic swap operators:

1. **Constrained One-Point Swap**: For a tidy-up permutation $\Pi^{\star} = (\pi_0, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_{n-1})$, two service indexes π_a, where $0 \leq a \leq t$, and π_b, where $t+1 \leq b \leq n-1$, are selected and exchanged.

   The constrained one-point swap local search operator is inspired by [47], which swaps a pair of service indexes in a permutation. In [47], local search exclusively explores the neighbourhood based on one selected index of the permutation. Therefore, the size of the neighbourhood associated with the index is $n-1$. However, it can be very computationally expensive because the number of swaps becomes significant for a large n. Besides that, it can be less flexible as the neighbourhoods are just focusing on those neighbourhoods in relation to one selected index.

   To perform the constrained one-point swap, we pre-determine a fixed, relatively small number of neighbours n_{nb} to be produced for a considerable computational time for local search. Meanwhile, we randomly produce n_{nb} neighbours by swapping two randomly selected indexes, rather than by swapping $n-1$ indexes with one fixed index. We expect that swapping two randomly selected indexes is

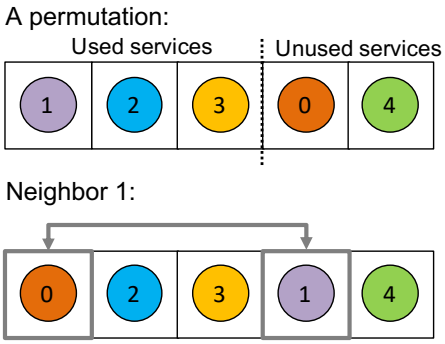


Fig. 3.7: An example of a constrained one-point swap on $[1, 2, 3 \mid 0, 4]$.

more effective within a budget computation time for making local improvements. In addition, we constrain the two randomly selected indexes that they must be before | and after | respectively in every swap because these swaps exclude those that have lower opportunities for local improvements. For example, one neighbour is created by swapping one pair of used service indexes. This swap operation has a high chance to produce the same DAG-based solution. Fig. 3.7 shows an example of constrained one-point swap for a selected permutation $[1, 2, 3 \mid 0, 4]$.

2. **Constrained Two-Point Swap**: For a tidy-up permutation $\Pi^{\star} = (\pi_0, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_{n-1})$, four service indexes π_{a_1}, π_{a_2}, π_{b_1}, and π_{b_2} are selected, where $0 \leq a_1 \leq t$, $0 \leq a_2 \leq t$, $t+1 \leq b_1 \leq n-1$, $t+1 \leq b_2 \leq n-1$, $a_1 \neq a_2$, and $b_1 \neq b_2$. π_{a_1} and π_{b_1} are exchanged. Likewise, π_{a_2} and π_{b_2} are exchanged.

   Based on the constrained one-point swap proposed above, we created a constrained two-point swap operator by combing two constrained one-point swap into a single operator. Particularly, this operator produces only one neighbour by two consecutive constrained one-point swaps. Compared to the constrained one-point swap, constrained two-point swap is more likely to make local changes on a candidate solution given the same number of swaps. Fig. 3.8 shows

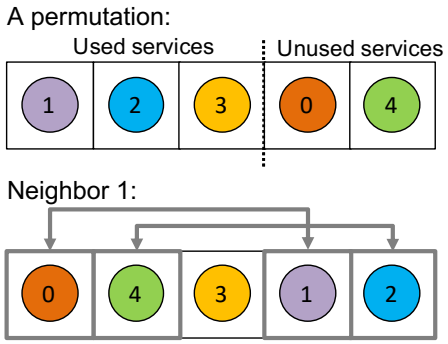


Fig. 3.8: An example of two-point swap on $[1, 2, 3 \mid 0, 4]$.

an example of a constrained two-point swap for a selected permutation $[1, 2, 3 \mid 0, 4]$.

3. **Constrained One-Block Swap**: For a tidy-up permutation $\Pi^{\star} = (\pi_0, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_{n-1})$, two sub-blocks $\{\pi_a, \ldots, \pi_t\}$, where $0 \leq a < t$ and $\{\pi_b, \ldots, \pi_{n-1}\}$, where $t + 1 \leq b < n - 1$, are selected and exchanged.

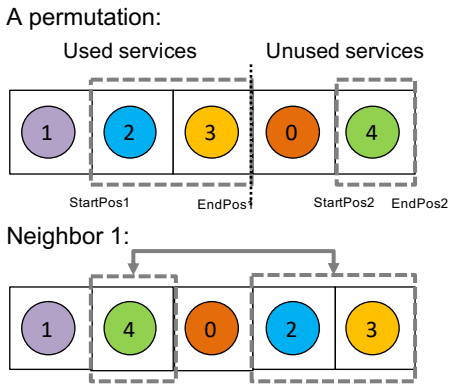


Fig. 3.9: An example of one constrained block-swap on $[1, 2, 3 \mid 0, 4]$.

A constrained one-block swap is proposed based on the concept of a block, i.e., consecutive service indexes in a permutation. In this swap, two blocks are built up, starting with two randomly selected points, π_a (i.e., a point must be selected before $|$) and π_b (i.e., a point

must be selected after |), on a permutation, respectively. Fig. 3.8 shows an example of a constrained one-block swap for a permutation $[1, 2, 3 \mid 0, 4]$, where one block is built up from the start position $StartPos1$ to the last position of used services, and another block is built up from the start position $StartPos2$ to the last position of the permutation.

4. **Layer-Based Constrained One-Point Swap**: For a tidy-up permutation $\Pi^{\star} = (\pi_0, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_{n-1})$, one service index π_a, where $0 \leq a \leq t$, is selected, and one layer $\mathcal{L}'$, where $\mathcal{L}'$ $s.t.$ $\pi_a \in \mathcal{L}'$, is identified. Afterwards, another service index π_b is randomly selected from the index set $\mathcal{L}' \cap \{\pi_{t+1}, \ldots, \pi_{n-1}\}$. Consequently, π_a and π_b are exchanged.

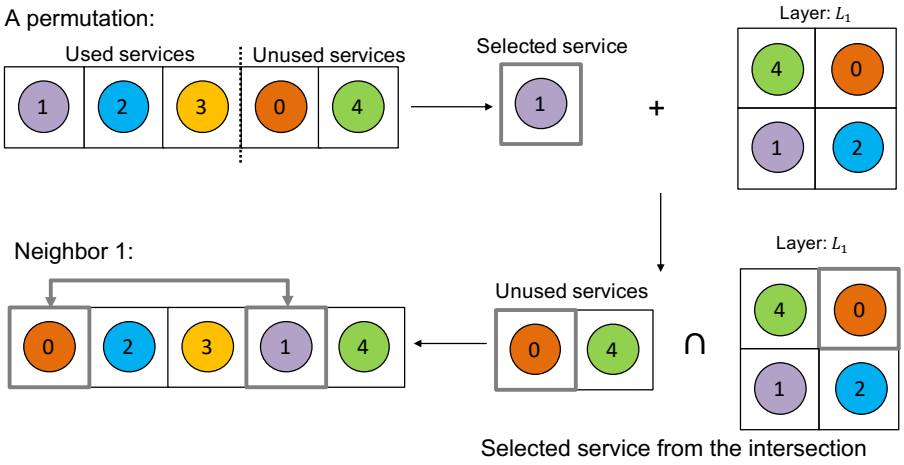


Fig. 3.10: An example of layer-based one-point swap operation on $[1, 2, 3 \mid 0, 4]$.

A layer-based one-point constrained swap is proposed by extending the constrained one-point swap while considering the layer information, introduced in Sect. 3.4. The benefit of considering layers allows us to identify a suitable pair of service indexes for a swap, compared to a constrained one-point swap. Such a pair of service indexes not

only comes from two different parts of a permutation (i.e., before and after |) but also comes from the same layer. By doing these, a layer-based one-point constrained swap operator is more likely to produce different neighbours in the DAG form. Herein, we propose a layer-based one-point swap operator: we first select one service index and identify its associated layer, and then select another service index randomly from a set of indexes, i.e., an intersection of the indexes of the identified layer and the indexes of unused services. (Note that when the intersection is an empty set, it indicates that the identified layer only contains one service index, so we re-select the first service index π_a). Consequently, two service indexes are exchanged. Fig. 3.10 shows an example of layer-based constrained one-point swap for creating one neighbour from a selected permutation $[1, 2, 3 \mid 0, 4]$.

Since the layer-based constrained one-point swap is proposed based on the constraint one-point swap, we use an example to analyze differences between them in Fig. 3.11. Fig. 3.11 exhibits an example of two corresponding neighbours produced by the constraint one-point swap. In the example, one identical solution can be decoded from both a permutation and two of its neighbours. This indicates that the constrained one-point swap does not properly exploit the neighbours of the permutations. In contrast, these two swaps are not permitted in the layer-based constraint one-point swap, where any produced neighbour must strictly follow the layer order on the permutation before |.

In the example, the permutation $[1, 2, 3 \mid 4, 0]$ is highlighted with two layers (i.e., $\mathcal{L}_1$ and $\mathcal{L}_2$) in ascending order. Particularly, $S_1, S_2 \in \mathcal{L}_1$ and $S_3 \in \mathcal{L}_2$. When the constrained one-point swap is performed, S_3 in the permutation are replaced with S_4 or S_0 in the produced neighbour 1 and neighbour 2 respectively. However, $\mathcal{L}_2$ is destroyed in the produced neighbours because of $S_4 \in \mathcal{L}_1$ and $S_0 \in \mathcal{L}_1$. Apparently, the layer-based constrained one-point swap can prevents these two

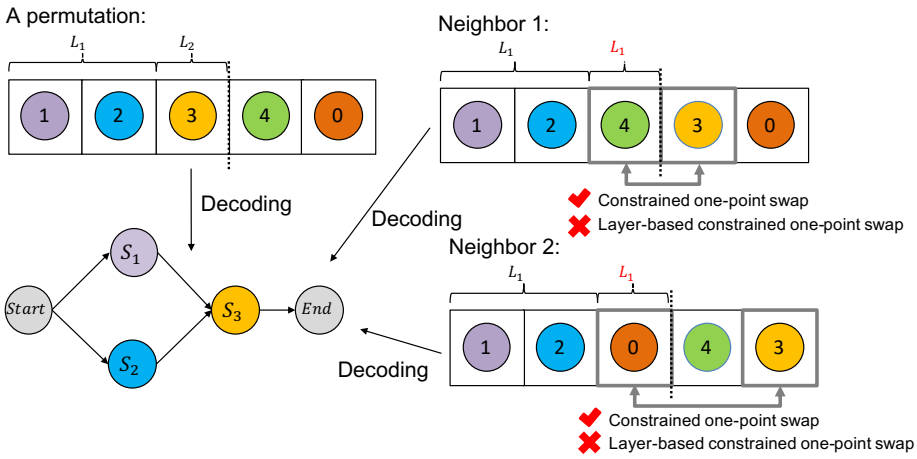


Fig. 3.11: An example of layer order breached by constrained one swap operation.

neighbours from being exploited.

3.9 Experimental Evaluation

In this section, we conduct experiments to evaluate the performances of our memetic EDA-based algorithms, i.e., memetic EDA with the constrained one-point swap (henceforth referred to as MEEDA-OP), memetic EDA with the constrained two-point swap (henceforth referred to as MEEDA-TP), memetic EDA with the constrained layer-based one-point swap (henceforth referred to as MEEDA-LOP) and memetic EDA with the constrained one-block swap (henceforth referred to as MEEDA-OB). These memetic EDA-based approaches are compared to MEFL [47], and the best performer found in Sect. 3.7, i.e., EDA-NHM, in finding high-quality solutions.

We follow the same experimental setting in Sect. 3.7 for the population size, the number of generations, b_{ratio}, for EDA-NHM, MEEDA-OP, MEEDA-TP, MEEDA-OB and MEEDA-LOP. For the local search proce-

dure, a pair of parameters, i.e., the number of selected solutions for local search, $Selected\ IndiSet$, and the number of swaps, n_{nb}, are set to 6 and 20, respectively. We have tuned this pair of parameters slighty by considering other pairs, such as $Selected\ IndiSet$ equals 4 and n_{nb} equals 30, and $SelectedIndiSet$ equals 8 and n_{nb} equals 15. We found that 6 and 20 present a better parameter setting for $SelectedIndiSet$ and n_{nb} than the others in finding high-quality solutions. For other competing methods, we strictly follow their settings reported in their papers. For example, the crossover rate and mutation rate of MEFL are set to 0.95 and 0.05, respectively, and the local search rate is 0.05. The weights in the fitness function in Eq. (3.5) and the p of $type_{link}$ in Eq. (3.1) follow our suggestion in Sect. 3.7.

3.9.1 Comparing Memetic EDA-based Methods with EDA-NHM and MEFL

Comparison of the Fitness

The second experiment aims to evaluate the effectiveness of all the proposed memetic EDA-based approaches comparing to MEFL and EDA-NHM. Note that, the same independent sample T-test and pairwise comparison methods utilized in Sect. 3.7.1 will be employed in this section. Table 3.6 shows the mean value of the fitness and the standard deviation over 30 repetitions for MEEDA-OP, MEEDA-TP, MEEDA-OB, MEEDA-LOP, EDA-NHM and MEFL. The pairwise comparison results of the fitness value are summarized in Table 3.7. Note that any "$-$" in the tables means that results cannot be collected when the corresponding method has been running for 7 days. By analyzing the results in these tables, we can conclude the observations about the effectiveness of all the competing methods as follows:

Firstly, all the memetic EDA algorithms, including MEEDA-OP, MEEDA-TP, MEEDA-OB, and MEEDA-LOP, significantly outperform the baseline method EDA-NHM. This observation corresponds well with our expecta-

tion that the exploitation ability of EDA can be enhanced by hybridizing it with local search.

Secondly, MEEDA-LOP is identified as the best performer in finding high-quality composite services. This observation corresponds well with our assumption that the layer-based constrained one-point swap is more effective than other swap operators. Meanwhile, MEEDA-LOP has achieved extremely stable performance in most runs with 0 standard deviations.

Lastly, MEEDA-OP, MEEDA-TP outperform MFEA, while MEEDA-OB and MFEA are comparable to each other. This is becasue EDA-NHM (i.e., the baseline method of MEEDA-OP, MEEDA-TP) is more effective than FL (i.e., the baseline method of MFEA), as discussed in Sect. 3.7.1. The effectiveness of MEEDA-OB does not meet our expectation. This is because swapping building blocks can potentially ruin the learned knowledge of promising solutions, resulting in poor searching behaviour.

In summary, we sort all the competing approaches based on the effectiveness in descending order: MEEDA-LOP $>$ MEEDA-TP $=$ MEEDA-OP $>$ MEEDA-OB $=$ MEFL $>$ EDA-NHM. Apparently, the layer-based constrained one-point swap operator is the most effective swap for enhancing EDA-NHM in terms of effectiveness.

Table 3.6: Mean fitness values for our memetic EDA algorithms in comparison to EDA-NHM and MEFL. (Note: the higher the fitness the better)

Dataset	MEEDA-OP	MEEDA-TP	MEEDA-OB	MEEDA-LOP	EDA-NHM	MEFL
WSC08-1	0.613745 ± 0	0.613745 ± 0	0.613745 ± 0	0.613745 ± 0	0.613745 ± 0	0.613745 ± 0
WSC08-2	0.756812 ± 0	0.756812 ± 0	0.756812 ± 0	0.756812 ± 0	0.756812 ± 0	0.756812 ± 0
WSC08-3	$0.477864 \pm 5.1e-05$	$0.477866 \pm 4e-05$	$0.477803 \pm 6.7e-05$	$0.47791 \pm 1.7e-05$	$0.477802 \pm 5.2ee05$	0.477768 ± 0.00015
WSC08-4	0.557815 ± 0	0.557815 ± 0	0.557815 ± 0	0.557815 ± 0	0.557815 ± 0	0.557815 ± 0
WSC08-5	0.52453 ± 0.00038	0.52474 ± 0.000388	0.524482 ± 0.000369	0.524412 ± 0.000329	0.524492 ± 0.000415	0.525586 ± 0.000639
WSC08-6	0.482698 ± 0.000107	0.482684 ± 0.000143	0.482622 ± 0.000131	0.482703 ± 0.000225	0.482671 ± 0.000139	0.482759 ± 0.000253
WSC08-7	0.523588 ± 0	0.523588 ± 0	$0.523577 \pm 5.8e-05$	0.523588 ± 0	$0.523584 \pm 2.2ee05$	0.523559 ± 0.00014
WSC08-8	$0.497676 \pm 1.3e-05$	$0.497676 \pm 1.3e-05$	$0.497646 \pm 9.4e-05$	0.497679 ± 0	0.497638 ± 0.000126	0.497455 ± 0.000357
WSC09-1	0.64902 ± 0.003747	0.650861 ± 0.00383	0.64852 ± 0.003005	0.650444 ± 0.00412	0.647988 ± 0.002939	0.650737 ± 0.003863
WSC09-2	0.521817 ± 0.003855	0.520562 ± 0.004943	0.520966 ± 0.008258	0.52309 ± 0.000443	0.517561 ± 0.012522	0.52257 ± 0.003246
WSC09-3	0.583978 ± 0	0.583978 ± 0	0.583978 ± 0	0.583978 ± 0	0.583978 ± 0	0.583978 ± 0
WSC09-4	0.484428 ± 0.000191	0.484427 ± 0.00032	0.484295 ± 0.00014	0.485533 ± 0.002073	0.484315 ± 0.000104	-
WSC09-5	$0.484832 \pm 2.6e-05$	$0.484818 \pm 9.5e-05$	0.484788 ± 0.000179	0.484825 ± 0.000117	$0.484818 \pm 4e-05$	0.484603 ± 0.000294

Table 3.7: Summary of statistical significance tests for fitness, where each column shows win/draw/loss score of an approach against others.

Dataset	Method	MEEDA-OP	MEEDA-TP	MEEDA-OB	MEEDA-LOP	EDA-NHM	MEFL
WSC-08 (8 tasks)	MEEDA-OP	-	0/8/0	0/6/2	1/7/0	0/6/2	1/4/3
	MEEDA-TP	0/8/0	-	1/5/2	1/6/1	0/6/2	1/4/3
	MEEDA-OB	2/6/0	2/5/1	-	1/7/0	0/8/0	2/5/1
	MEEDA-LOP	0/7/1	1/6/1	0/7/1	-	0/5/3	1/4/3
	EDA-NHM	2/6/0	2/6/0	0/8/0	3/5/0	-	1/6/1
	MEFL	3/4/1	3/4/1	1/5/2	3/4/1	1/6/1	-
WSC-09 (5 tasks)	MEEDA-OP	-	0/5/0	0/4/1	1/4/0	0/4/1	0/3/1
	MEEDA-TP	0/5/0	-	0/4/1	2/3/0	0/4/1	1/2/1
	MEEDA-OB	1/4/0	1/4/0	-	2/3/0	0/5/0	1/2/1
	MEEDA-LOP	0/4/1	0/3/2	0/3/2	-	0/2/3	0/3/1
	EDA-NHM	1/4/0	1/4/0	0/5/0	3/2/0	-	2/1/2
	MEFL	2/3/0	2/2/1	2/2/1	2/3/0	2/1/2	-

Comparison of the Execution Time

The second objective of this experiment is to study the efficiency of all the proposed memetic EDA-based algorithms comparing to MEFL and EDA-NHM. Table 3.8 shows the mean value of the execution time and the standard deviation over 30 repetitions for MEEDA-OP, MEEDA-TP, MEEDA-OB, MEEDA-LOP, EDA-NHM and MEFL. The pairwise comparison results for the execution time are summarized in Table 3.9. By analyzing the results in these tables, we can conclude the observations about the execution time of all the competing methods as follows:

First, EDA-NHM requires the least execution time because this method does not involve local search. Among all the memetic methods, we observe that MEEDA-LOP requires consistently less execution time than other memetic approaches. This remarkable observation further confirms the best effectiveness of MEEDA-LOP, resulting in sampled permutations that are likely to have useful services to be put in the front. Such permutations can be decoded in DAGs much faster than those produced by other approaches.

Second, MEFL requires the highest execution time because its local search are performed on all the candidate solutions based on a pre-defined probability. In addition, its swap operator exclusively searches the whole neighbourhood of candidate solutions. This poor execution time confirms that the local search strategy in MEFL is very time-consuming.

In summary, we sort all the competing methods based on the execution time in ascending order: EDA-NHM $>$ MEEDA-LOP $>$ MEEDA-OP $>$ MEEDA-TP $>$ MEEDA-OB $>$ MEFL. Despite the least execution time consumed by EDA-NHM, EDA-NHM cannot outperform any memetic method. Therefore, MEEDA-LOP becomes the most suitable method since it not only consumes the second least execution time but also achieves the best effectiveness.

Table 3.8: Mean execution time (in s) for our memetic EDA algorithms in comparison to EDA-NHM, MEFL.
(Note: the shorter the time the better)

Dataset	MEEDA-OP	MEEDA-TP	MEEDA-OB	MEEDA-LOP	EDA-NHM	MEFL
WSC08-1	156 ± 12	211 ± 19	422 ± 70	112 ± 8	72 ± 6	622 ± 74
WSC08-2	72 ± 9	98 ± 13	133 ± 14	72 ± 6	32 ± 4	118 ± 17
WSC08-3	8470 ± 462	8807 ± 621	12849 ± 740	8329 ± 346	1296 ± 137	68382 ± 13142
WSC08-4	87 ± 6	112 ± 7	177 ± 7	84 ± 6	38 ± 4	766 ± 253
WSC08-5	1705 ± 148	2136 ± 142	4363 ± 154	1742 ± 122	979 ± 96	47603 ± 47104
WSC08-6	17524 ± 843	19954 ± 1352	43621 ± 2062	17303 ± 1569	13964 ± 1735	947368 ± 157828
WSC08-7	2025 ± 138	2869 ± 1258	8096 ± 448	1918 ± 119	1181 ± 115	81847 ± 20610
WSC08-8	4375 ± 371	5066 ± 350	11341 ± 666	4283 ± 368	2677 ± 308	148133 ± 29304
WSC09-1	159 ± 23	239 ± 38	314 ± 27	159 ± 18	62 ± 7	506 ± 104
WSC09-2	3314 ± 551	4311 ± 686	7573 ± 553	3362 ± 505	2204 ± 378	49455 ± 17831
WSC09-3	1643 ± 146	2303 ± 191	4638 ± 343	1614 ± 124	819 ± 65	18998 ± 3300
WSC09-4	92342 ± 7584	103433 ± 6847	214067 ± 12358	86543 ± 6046	70105 ± 6772	-
WSC09-5	16160 ± 1123	18446 ± 1776	45039 ± 4534	15249 ± 978	11117 ± 1150	635637 ± 151975

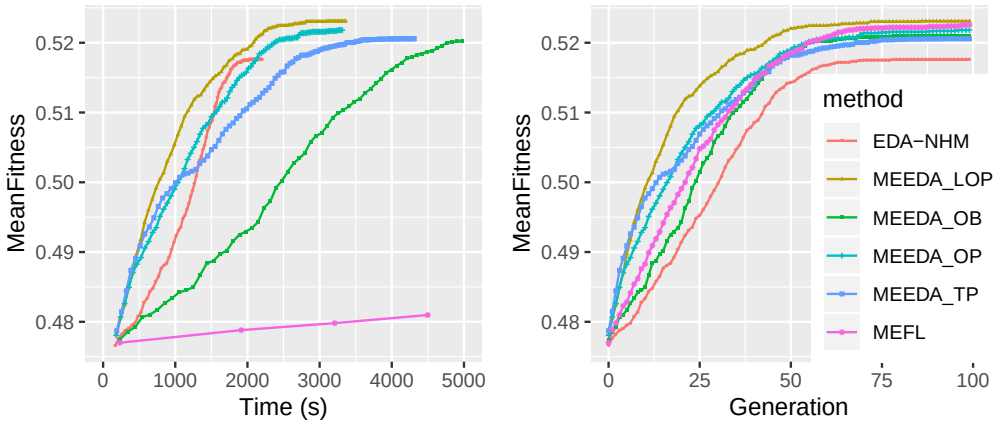
Table 3.9: Summary of statistical significance tests for execution time (in s), where each column shows win/draw/loss score of an approach against others.

Dataset	Method	MEEDA-OP	MEEDA-TP	MEEDA-OB	MEEDA-LOP	EDA-NHM	MEFL
WSC-08 (8 tasks)	MEEDA-OP	-	0/0/8	0/0/8	6/2/0	8/0/0	0/0/8
	MEEDA-TP	8/0/0	-	0/0/8	8/0/0	8/0/0	0/0/8
	MEEDA-OB	8/0/0	8/0/0	-	8/0/0	8/0/0	0/0/8
	MEEDA-LOP	0/2/6	0/0/8	0/0/8	-	8/0/0	0/0/8
	EDA-NHM	0/0/8	0/0/8	0/0/8	0/0/8	-	0/0/8
	MEFL	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	-
WSC-09 (5 tasks)	MEEDA-OP	-	0/0/5	0/0/5	2/3/0	5/0/0	0/0/4
	MEEDA-TP	5/0/0	-	0/0/5	5/0/0	5/0/0	0/0/4
	MEEDA-OB	5/0/0	5/0/0	-	5/0/0	5/0/0	0/0/4
	MEEDA-LOP	0/3/2	0/0/5	0/0/5	-	5/0/0	0/0/4
	EDA-NHM	3/2/0	2/1/2	0/0/5	3/2/0	-	0/0/4
	MEFL	4/0/0	4/0/0	4/0/0	4/0/0	5/0/0	-

Comparison of the Convergence curve

The third objective of our experiment is to study the convergence curves of all the algorithms. We use WSC09-2 as an example to illustrate the convergence curves of all the compared methods.

Fig. 3.12 depicts the evolution of the mean fitness values of the best solutions found by MEEDA-OP, MEEDA-TP, MEEDA-OB, MEEDA-LOP, EDA-NHM and MEFL with respect to both the execution time and generation over 30 independent runs. As MEFL requires much longer time for execution, we set a threshold for the execution time scale for WSC09-2 to easily observe their differences in Fig. 3.12a.



(a) WSC-09-2 (b) WSC-09-2

Fig. 3.12: A comparison of the average convergence rate of EDA-NHM, EDA-EHM, PSO, FL over execution time (the left) and generation (the right) for WSC09-2.

First, we observe a significant increase in the fitness value towards the optimum over all the algorithms excluding MEFL in Fig. 3.12a. Particularly, given the same budget of execution time, memetic EDA-based algorithms happen to converge significantly faster than MEFL. In Fig. 3.12b, all algorithms happen to converge fast along the generations. These algorithms eventually reach different levels of plateaus.

Second, MEFL suffers from the scalability issue when the size of the service repository is doubled in our new benchmark. The complexity of the local search in MEFL strongly depends the length of the permutation. In Fig. 3.12a, we can observe that MEFL does not even converge at all when the same amount of execution time is assigned.

Lastly, MEEDA-LOP is consistently ranked as a top performer among all the competing algorithms. The convergence rate of MEEDA-OP and MEEDA-TP presents a very similar pattern. MEEDA-OB happens to converge slower than the others, but it eventually reaches comparable results compared to MEEDA-OP and MEEDA-TP.

Comparison of local search operators

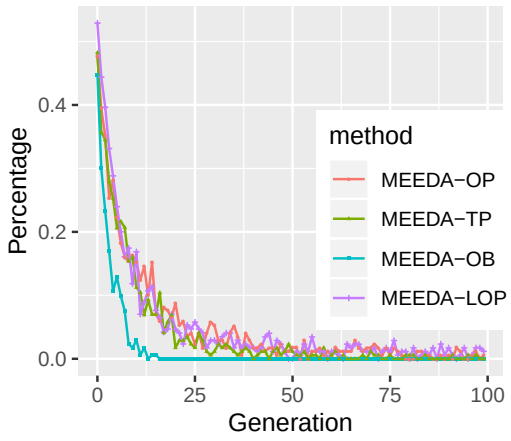


Fig. 3.13: A comparison of the percentage of better neighbours produced by four memetic algorithms along generations over 30 runs for WSC08-03.

To demonstrate which swap-based local search operator is more likely to produce better solutions, we investigate how often the mean fitness of neighbours is better than the fitness of their original permutation in MEEDA-OP, MEEDA-TP, MEEDA-LOP, and MEEDA-BP. Herein we use

the composition task WSC0803 as an example to demonstrate the percentage of better neighbours produced by our four memetic EDA-based approaches along generations over 30 runs in Fig. 3.13. The result shows that MEEDA-BP and MEEDA-TP are less likely to produce better solutions, while MEEDA-OP and MEEDA-LOP are highly comparable to each other.

3.10 Summary

The overall goal of this chapter is to propose EDA-based fully automated service composition approaches to generate composite services with near-optimized comprehensive quality in terms of QoSM and QoS. We have achieved this goal with the following contributions: (1) A comprehensive quality model is proposed to represent users' requirements in terms of both QoSM and QoS. (2) An EDA-NHM algorithm is proposed along with a novel permutation-based representation, which allows learning a suitable distribution via NHM. This distribution learns the positions of component services over varied structures of composite services. In addition, NHBSA is employed to effectively sample new promising permutations. (3) An EDA-EHM algorithm is proposed along with a DAG-based representation that is presented as a set of service dependencies. These service dependencies contribute to learning a suitable distribution via EHM. This distribution can capture the essential ingredients (i.e., distribution of service dependencies) for building DAG-based composite services. In addition, a GEHBGSA is proposed to sample functionally valid and promising DAG-based composite services. (4) To demonstrate the performance of EDA-NHM and EDA-EHM, EDA-NHM and EDA-EHM are compared to some state-of-the-art service composition approaches, i.e., PSO [175], FL [47], and SearchPath [32]. (5) Memetic EDA-based approaches (that includes MEEDA-OP, MEEDA-TP, MEEDA-OB and MEEDA-LOP) are proposed based on EDA-NHM, which is chosen because of its better effective-

ness in finding high-quality composite services, compared to PSO, FL, and SearchPath. Particularly, an effective and efficient local search procedure is proposed with an application of uniform distribution schema for selecting suitable solutions for local search and stochastic local search operators for effectively and efficiently exploiting neighbours. We investigate different local search operators, each of which is combined with EDA-NHM to form different variations of memetic EDA-based approaches. These approaches are compared to a state-of-the-art memetic GA, i.e., MEFL [47] and their baseline EDA-NHM.

The development of new algorithms and their experimental study leads to several major findings: (1) EDA-NHM achieves significantly better performance in finding composite services with high comprehensive quality, compared to EDA-EHM and other algorithms, such as PSO, FL, and SearchPath. This result indicates that NHM that captures positions of component services can be more effective than EHM that captures service dependencies. (2) EDA-EHN could deliver moderate effectiveness in finding high-quality composite services with the highest efficiency among all the EC-based approaches. (3) The uniform distribution schema and the layer-based constrained one-point swap jointly form a very effective and efficient memetic EDA algorithm, i.e., MEEDA-LOP, achieving cutting-edge performance in terms of both effectiveness and efficiency.

Chapter 4

Multi-Objective Fully Automated Web Service Composition

4.1 Introduction

The previous chapter presented EC-based approaches to single-objective fully automated semantic web service composition, which aims to optimize QoSM and QoS using an aggregated fitness function with different weights associated to different quality criteria. Such weights are defined based on the importance of these quality criteria, and are set by users. In other words, we assume the users have clear preferences on these quality criteria. Although the single-objective optimization is a very effective strategy, users often do not have clear preferences on trade-off solutions before they see the trade-offs of the solutions [46, 40]. This chapter focuses on **M**ulti-objective **C**omprehensive **Q**uality-aware semantic web service composition **P**roblem (MOCQP, for short). Particularly, the two categories of MOCQP, discussed in Chapter 1.2, are to be studied in this chapter. The first one refers to semantic **W**eb **S**ervice **C**omposition problem for **M**ultiple independent **O**jectives (henceforth referred to as **WSC-MO**). WSC-MO aims to simultaneously optimize multiple conflicting quality criteria in our comprehensive quality model and produces a set of ap-

proximated Pareto-optimal composite services. The second one refers to semantic **W**eb **S**ervice **C**omposition problem for **M**ultiple user segments with distinctive **Q**oSM **P**references (henceforth referred to as **WSC-MQP**). WSC-MQP aims to simultaneously optimize the comprehensive quality with distinctive QoSM preferences for different user segments and produces a set of near-optimal composite services, each of which serves one user segment. The introduction to WSC-MO and WSC-MQP are discussed in Sect. 4.1.1 and Sect. 4.1.2, respectively.

4.1.1 Introduction to the WSC-MO Problem

WSC-MO aims to find a set of approximated Pareto-optimal solutions by simultaneously considering both QoSM and QoS. For example, some users may be willing to trade QoS for QoSM. In the literature, similar multi-objective service composition problems [40, 46] have been coped with via evolutionary multi-objective algorithms, such as NSGA-II [54]. For example, a very recent work [40] proposed two hybridized methods (called Hybrid and Hybrid-L) to tackle two objectives related to QoS criteria (i.e., one objective combines time and cost, another objective combines availability and reliability). Particularly, Hybrid and Hybrid-L effectively combine the use of two evolutionary optimization algorithms, i.e., NSGA-II and MOEA/D. They takes advantage of the fast non-dominated sorting strategy in NSGA-II and "*divide and conquer*" strategy in MOEA/D. Moreover, Hybrid-L allows local search to be performed on numerous decomposed single-objective optimization subproblems. Hybrid and Hybrid-L have successfully enhanced the performance of its baseline NSGA-II via local search, achieving outstanding effectiveness in finding Pareto solutions.

Despite this recent success, a large number of decomposed subproblems is predefined (e.g., 500 subproblems) in Hybrid and Hybrid-L [40], and a simple form of local search (i.e., so-called one-point "swap") is less effective and efficient to make local improvements in Hybrid-L be-

cause it is randomly applied to every subproblem without focusing on the most suitable candidate solutions in each generation. Furthermore, each "swap"-based local search procedure exploit the neighbourhood of one solution (i.e., a subproblem representative), ignoring any information of other promising candidate solutions that could be jointly used to guide the local search. Therefore, new memetic approaches must be developed to address these limitations. Besides that, to the best of our knowledge, existing EC-based multi-objective fully automated approaches only focus on QoS, overlooking QoSM of composite services. In practice, some customers often demand highly accurate outputs of composite services (i.e., the QoSM of this composite service is high), and are willing to trade QoS for QoSM. In contrast, a proportion of customers may prefer highly responsible composite services at an affordable cost (i.e., the QoS of this composite service is high). To address the limitations above, *we propose a novel memetic NSGA-II with EDA-based local search (henceforth referred to as MNSGA2-EDA) for multi-objective fully automated semantic service composition*, achieving substantially high performances in both effectiveness and efficiency. In addition, MNSGA2-EDA tackles two practical objectives, i.e., QoSM and QoS, with respect to the functional and non-functional quality aspects. The following objectives are sought in the first part of this chapter:

1. To avoid pre-determining a large number of single-objective subproblems in advance, we propose a new clustering technique to select candidate Pareto-optimal solutions for local search. This technique is performed separately and concurrently in different regions of the Pareto front, contributing to wide and uniformly distributed near-optimal Pareto solutions produced by our MNSGA2-EDA.

2. To perform an effective local search using the useful information of good candidate solutions in each generation, we propose a model-guided local search. Such a local search does not rely on any man-

ually pre-defined local search operator to form the neighborhood of the selected solutions. Instead, our local search first constructs distribution models from suitable Pareto front solutions and other good candidate solutions selected by our proposed clustering technique, and then samples effective solutions from the distribution models.

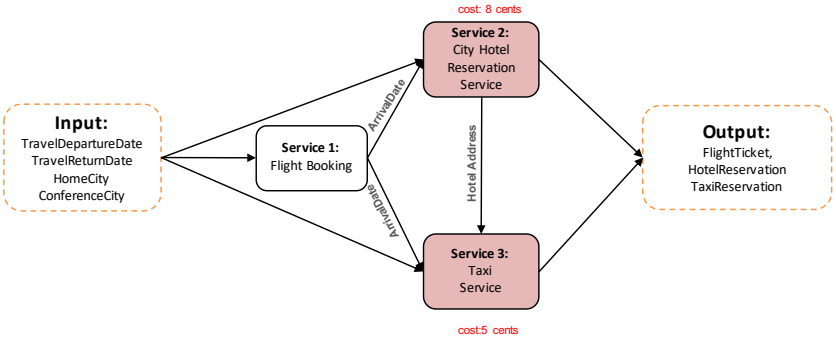
3. Considering two independent objectives on QoSM and QoS, we empirically evaluate NHSGA2-EDA by comparing it with NSGA-II, Hybrid and Hybrid-L [40]. Our experiment results show that NHSGA2-EDA is much more effective and efficient. We keep using the new benchmark dataset proposed in Chapter 3 to show that MNSGA2-EDA can maintain high performance on the WSC-MO problem with significantly larger sizes.

4.1.2 Introduction to the WSC-MQP Problem

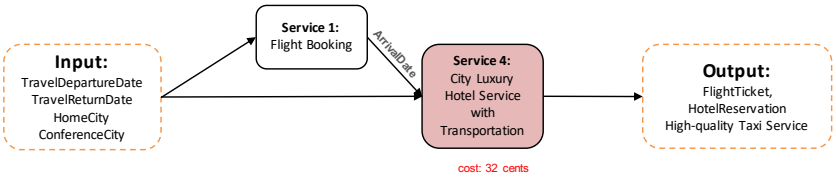
WSC-MQP aims to optimize QoSM and QoS of a set of composite services subject to multiple user segments with distinctive QoSM preferences. As discussed in Sect. 1.2, to distinguish different types of users, service composers often strategically group relevant service composition requests into several user segments (e.g., platinum, gold, silver, and bronze user segments), and each user segment is associated with distinctive preferences on QoSM. Therefore, one composite service for a user segment can comfortably satisfy requirements from all users belonging to the same segment. In other words, any new service requests arising from the same segment will be immediately served by the same composite service designed a priori for that segment.

Herein we use an example to demonstrate composites services for two user segments (Note that the number of segments could be more than two in the real world). TripPlanner is a service composition design system that produces composite booking services for many travelling companies. See an example of two composite booking services (i.e., composite book-

Fig. 4.1: Two composite booking services produced by TripPlanner.



(a) Composite booking service A



(b) Composite booking service B

ing service A and composite booking service B) produced by TripPlanner in Figure. 4.1. Both two composite services can be used to book airlines, hotels, and local transportation for travellers. In Fig. 4.1, some component services in composite booking service A (i.e., Service 2: City Hotel Reservation Service and Service 3: Taxi Service) are different from those in composite booking service B (i.e., Service 4: City Luxury Hotel Service with Transportation). Apparently, Service 4 aggregates the functionalities of Service 2 and Service 3, providing high-quality hotel and taxi services. Apart from that, the cost of executing Service 4 (i.e., 32 cents) is much higher than that of Service 2 and Service 3 (i.e., 8 + 5 = 13 cents). Therefore, these two composite booking services differ in QoSM and QoS. It is important to cater for different users with varied QoSM and QoS requirements. For example, large international travel companies (i.e., platinum segment users of TripPlaner) often care about their customers' needs more

than small local travelling companies (i.e., bronze segment users of TripPlaner), by providing high-quality services. These high-quality services contribute to reliable and accurate information required by users. In other words, composite services with high QoSM, such as the composite booking service B in Fig. 4.1, are preferably considered by the platinum segment users. In contrast, composite services with low QoSM with a trade-off in cost, such as the composite booking service A, is preferred by bronze segment users. From the perspective of service developers, they should distinguish different types of companies and provide different segment offers (i.e., composite services) to different segment.

We perceive the problem discussed above as a multi-tasking problem, i.e., multiple similar service composition requests from different user segments with distinctive QoSM preferences. Our goal is to solve these multiple requests simultaneously.

Existing service composition algorithms are designed primarily to solve each service composition request independently by using either single-objective [152, 42, 221] or multi-objective approaches [35, 214, 40, 46], ignoring similarities between different requests that could be dealt with collectively.

Recently, Gupta et al. [71] introduce a new EC computing paradigm, namely, multi-factorial evolutionary algorithm (MFEA) [71] with a unified random-key representation to search solutions for multiple tasks (or optimization problems) simultaneously. MFEA transfers implicit knowledge of promising solutions through the use of simple genetic operators across multiple tasks. These genetic operators allow two randomly selected parents to undergo crossover or mutation with certain conditions on the tasks. This genetic mechanism is called *assortative mating* [71]. Besides, the offspring is only evaluated on one selected task determined by its parents based on *vertical cultural transmission*. MFEA has shown its efficiency and effectiveness in several problem domains [17, 59, 225, 233].

Very recently, [17] reported the first attempt to search solutions with

high QoS for two unrelated composition tasks concurrently using MFEA, outperforming some basic single-objective EC techniques. Despite this recent success, this work [17] can only handle semi-automated service composition problems, i.e., a specific service workflow must be given in advance and strictly obeyed. Meanwhile, QoSM requirements are not considered in their work. In addition, the number of composition tasks that are optimized concurrently is very small (e.g., two tasks in [17]). Lastly, test cases used for the experiments are small (e.g., each test case only contains 2507 atomic web services in [17]). The findings of their experiments are based on comparisons with some basic EC techniques, overlooking state-of-the-art service composition approaches. *To address the limitations above, we propose a novel **P**ermutation-based **M**ultifactorial **E**volutionary **A**lgorithm (called **PMFEA**) to solve the WSC-MQP problem.*

Based on PMFEA, we conduct a further study on improving the effectiveness of assortative mating in promoting constructive inter-task knowledge sharing. Different from the implicit learning and sharing via crossover in PMFEA, we will propose a new technique to explicitly learn and share knowledge via a series of NHMs. The learned NHMs can be further utilized by EDA to search for multiple near-optimal solutions simultaneously with respect to multiple service requests. Note that existing EDA-based approaches for service composition have never been designed to extract and utilize knowledge from multiple tasks. In this chapter, we will propose the first **P**ermutation-based **M**ulti-factorial **E**volutionary **A**lgorithm based on EDA (called **PMFEA-EDA**) to simultaneously solve the WSC-MQP problem. PMFEA-EDA features the use of innovative inter-task knowledge sharing techniques and sampling techniques for producing promising composite services for multiple tasks. The following objectives are sought in the second part of this chapter:

1. To handle the WSC-MQP problem, we are the first in the literature to formulate this problem as a multi-tasking and fully automated service composition problem. To effectively and efficiently solve this

problem, we propose PMFEA with a permutation-based representation and an assortative mating with domain-specific crossover and mutation introduced in [47]. This assortative mating allows implicit knowledge to be learned and shared across tasks. Furthermore, we introduce a neighbourhood structure over multiple tasks to allow newly evolved solutions to be additionally evaluated on the neighbouring tasks. The use of this neighbourhood structure has a severe impact on the effectiveness as well as the efficiency of PMFEA for optimizing more than two tasks concurrently.

2. To explore the performance of PMFEA and two of its variations (i.e., PMFEA with evaluations on **N**eighboring **T**asks, called PMFEA-NT, and PMFEA with evaluations on **A**ll **T**asks, called PMFEA-AT), we compare them against a state-of-the-art single-tasking single-objective approach, i.e., FL [47]. For the experiments, we use the benchmark datasets proposed in Chapter 3. The evaluation shows that all PMFEA approaches are performed at the cost of only a fraction of time. In particular, PMFEA-NT achieves the best performance in terms of both effectiveness and efficiency.

3. Different from the previously proposed PMFEAs, a new method, namely PMFEA-EDA, is proposed with an explicit knowledge learning and sharing technique for solutions across multiple service requests. Particularly, PMFEA-EDA iteratively builds a set of single-tasking NHMs. Each NHM captures the knowledge of good solutions with respect to one task. Meanwhile, to facilitate knowledge sharing across different tasks, PMFEA-EDA also learns multi-tasking NHMs in association with every two tasks with similar preferences on QoSM (i.e. neighbouring tasks). To balance the exploration and exploitation of the evolutionary search process in a multi-tasking context, we propose a new sampling mechanism, inspired by the principle of assortative mating [71], to construct new composite ser-

vices based on single-tasking NHMs and multi-tasking NHMs. By using this mechanism, we can also effectively prevent our method from pre-mature convergence.

4. To demonstrate the effectiveness and efficiency of our PMFEAs using either implicit or explicit knowledge learning and sharing technique, we conduct experiments to evaluate the performance of four algorithms, i.e., PMFEA, PMFEA-EDA, and two single-tasking single-objective algorithms (i.e., EDA-NHM proposed in Chapter 3, and FL [47]). Meanwhile, we also study the effectiveness of explicit knowledge sharing across neighbouring tasks in PMFEA-EDA in terms of its impact on the quality of obtained solutions for all the tasks. This is achieved by experimentally comparing PMFEA-EDA with PMFEA-EDA-WOT that does not permit knowledge sharing.

4.2 Chapter Organization

The remainder of this chapter is organized as follows: Sect. 4.3 describes the WSC-MO problem. Sect. 4.4 presents our MNSGA2-EDA algorithm for the WSC-MO problem. Sect. 4.5 outlines the experimental design and results for evaluating the performance of MNSGA2-EDA. Sect. 4.6 describes the WSC-MQP problem. Sect. 4.7 and 4.8 present our two methods, namely, PMFEA and PMFEA-EDA, for solving the WSC-MQP problem. Sect. 4.9 outlines the experimental design and results for different comparisons with PMFEA and PMFEA-EDA. Lastly, Sect. 4.10 summarises this chapter.

4.3 The WSC-MO Problem

WSC-MO concerns the quality of composite services in both functional (i.e., QoSM) and non-functional (i.e., QoS) aspects. However, according to

our knowledge, no attempts have ever been reported in literature to address this problem in a multi-objective setting where QoSM and QoS are optimized separately. Herein, we formulate WSC-MO based on two objectives that reflect the functional (i.e., QoSM) and non-functional quality criteria (i.e., QoS) as follows:

$$\begin{gathered}\text{Minimize } \vec{f}(C) = (f_1(C), f_2(C)) \\ \text{subject to } C \in \mathcal{Z}\end{gathered} \tag{4.1}$$

$$f_1(C) = w_1(1 - \hat{MT}) + w_2(1 - \hat{SIM}) \tag{4.2}$$

$$f_2(C) = w_3(1 - \hat{A}) + w_4(1 - \hat{R}) + w_5\hat{T} + w_6\hat{CT} \tag{4.3}$$

where $\mathcal{Z}$ denotes the set of all composite services over a given repository of atomic services, and f_1, f_2 are two objective functions that capture QoSM and QoS, respectively, for every service C in $\mathcal{Z}$. In particular, QoSM is calculated based on the normalized semantic matching type $\hat{MT}$ and the semantic similarity $\hat{SIM}$ while QoS is calculated based on the normalized availability $\hat{A}$, reliability $\hat{R}$, response time $\hat{T}$, and execution cost $\hat{CT}$, see calculations in Sect. 3.3. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$ and $\hat{R}$ are offset by 1, so that lower scores correspond to better quality.

WSC-MO is to find the set of Pareto optimal composite services $PF^\star = \{C^\star \in \mathcal{Z}\}$, where $C^\star$ is Pareto optimal if $\nexists C' \in \mathbf{C}$, such that $C^\star \prec C'$. Note that $C^\star \prec C'$ means C' dominates $C^\star$ if $f_1(C^\star) \geq f_1(C')$ and $f_2(C^\star) > f_2(C')$ or if $f_1(C^\star) > f_1(C')$ and $f_2(C^\star) \geq f_2(C')$.

4.4 The MNSGA2-EDA Algorithm

In this section, we present MNSGA2-EDA for solving the WSC-MO problem, starting with an overview of MNSGA2-EDA in Sect. 4.4.1. Subsequently, we discuss the outlines of MNSGA2-EDA in Sect. 4.4.2. Furthermore, we present the genetic operators of MNSGA2-EDA in Sect. 4.4.3.

Lastly, we discuss two important components in our EDA-based local search, consisting of identifying a cluster representative and learning an NHM based on the representative in Sect. 4.4.4 and 4.4.5, respetively.

4.4.1 An overview of MNSGA2-EDA

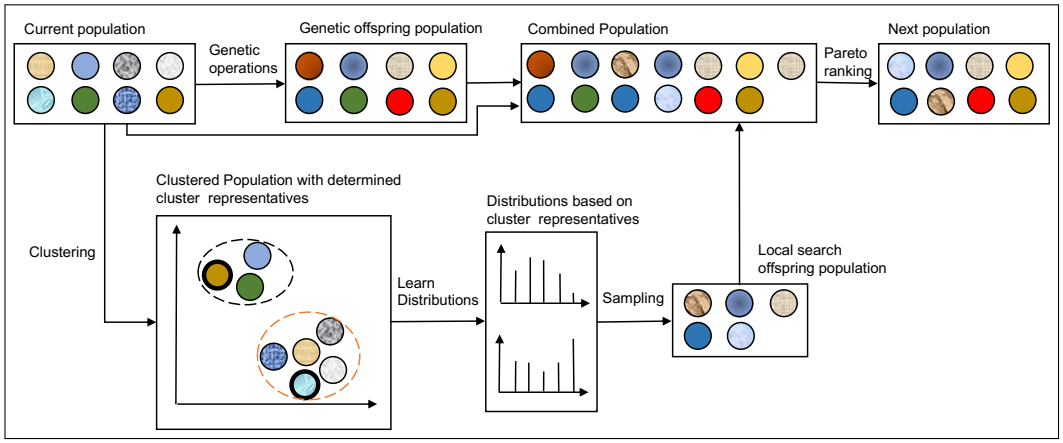


Fig. 4.2: Generation updates in MNSGA2-EDA.

MNSGA2-EDA enhances NSGA-II by EDA-based local search, where EDA is used to exploit better solutions based on several non-dominated solutions generated by NSGA-II. These solutions are selected separately and concurrently in different regions of the Pareto front for each generation. These regions are created by grouping the current Pareto front into multiple clusters.

The generation updates in MNSGA2-EDA is illustrated in Fig. 4.2. From the current population, two offspring populations are produced — genetic offspring population and local search offspring population. In particular, genetic offspring population is produced by genetic operators, including crossover and mutation (see details in Sect. 4.4.3); local search offspring population is produced by sampling from the distribution models constructed from the most suitable cluster representatives of the Pareto front (see details in Sect. 4.4.4 and 4.4.5).

4.4.2 Outline of MNSGA2-EDA

MNSGA2-EDA is outlined in Algorithm 18. Initially, we randomly generate m permutations Π_k^g as composite services for population $\mathcal{P}^g$ of generation g, where $0 \leq k < m$ and $g = 0$. Each permutation is a randomly ordered sequence of task-related service indexes. For example, let $\Pi = (\pi_0, \ldots, \pi_t, \ldots, \pi_{n-1})$ be a permutation-based composite service of service indexes $\{0, \ldots, t, \ldots, n-1\}$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. f_1, f_2 in Eq. (4.2) and Eq. (4.3) of any newly produced permutations will be evaluated by decoding each permutation into a DAG-based solution, $\mathcal{G}_k^g$. Subsequently, the following steps (Step 3 to 15) are repeated until a maximum number of generation g_{max} is reached. Particularly, the production of the first offspring population starts with a tournament selection in favor of winners with higher dominance regarding ranks and sparsity suggested in NSGA-II. The tournament winners will be processed by genetic operators (see details in Sub. 4.4.3) to produce genetic offspring population $\mathcal{P}_a^g$ based on a predefined probability. Afterwards, offspring $\mathcal{P}^g$ will be clustered into d clusters based on the values of f_1, f_2. For each cluster, we start by identifying its cluster representative Rep_{cl}^g, and then encode each cluster member $\mathcal{G}_k^g$ into a different permutation $\Pi^{\star g}_k$, element of which are ordered based on $\mathcal{G}_k^g$ (see details in Sect. 4.4.5). The process of producing this different permutation is exactly the same as that of producing the novel permutation in Sect. 3.5.2. As discussed in Sect. 3.5.2, this permutation allows more reliable and accurate learning of a NHM of each cluster, i.e., $\mathcal{NHM}_{cl}^g$. The contribution of each cluster member to the NHM is adjusted according to the Euclidean distance in the objective space between the cluster member and Rep_{cl}^g. Subsequently, $\mathcal{NHM}_{cl}^g$ is used to sample new local search offspring population $\mathcal{P}_b^g$ (see details in Sect. 4.4.5). Consequently, we produce the next population $\mathcal{P}^{g+1}$ by combining the current population $\mathcal{P}^g$, genetic offspring population $\mathcal{P}_a^g$ and local search offspring population $\mathcal{P}_b^g$. After evaluating newly generated solutions, we perform the fast non-dominated sorting on $\mathcal{P}^{g+1}$, and the top m individuals are re-

served to form the next generation $\mathcal{P}^{g+1}$. When the stopping criterion is met, the non-dominated solutions in $\mathcal{P}^{g_{max}}$ are returned as the output of MNSGA2-EDA.

ALGORITHM 18. MNSGA2-EDA for the WSC-MO problem.

Input : T, $\mathcal{SR}$, d and g_{max}
Output: A set of solutions

1: Randomly initialize population $\mathcal{P}^g$ of m permutations Π_k^g as solutions (where $g = 0$ and $k = 1, \ldots, m$);
2: Evaluate f_1, f_2 of the permutations by decoding them into DAGs $\mathcal{G}_k^g$;
3: **while** $g < g_{max}$ **do**
4: Use tournament selection based on the dominance;
5: Apply genetic operators to the tournament winners to form genetic offspring population $\mathcal{P}_a^g$;
6: Divide the whole population $\mathcal{P}^g$ into d clusters;
7: Set cluster counter $cl \leftarrow 0$;
8: **while** $cl < d$ **do**
9: Identify the cl^{th} cluster representative Rep_{cl}^g;
10: Encode each $\mathcal{G}_k^g$ in the cl^{th} cluster into a different permutation $\Pi^{\star g}_k$;
11: Learn $\mathcal{NHM}_{cl}^g$ over the cl^{th} cluster based on the representative Rep_{cl}^g to form sampling local search offspring population $\mathcal{P}_b^g$;
12: $\mathcal{P}^{g+1} = \mathcal{P}^g \cup \mathcal{P}_a^g \cup \mathcal{P}_b^g$;
13: Evaluate f_1, f_2 of each permutation in $\mathcal{P}^{g+1}$ by decoding it into $\mathcal{G}_k^g$;
14: Perform a fast non-dominated sorting on $\mathcal{P}^{g+1}$;
15: Keep top m solutions in $\mathcal{P}^{g+1}$;
16: Return non-dominated solutions in $\mathcal{P}^{g_{max}}$;

4.4.3 Genetic Operators

One order crossover [49] and one one-point swap mutation [40, 98] are employed to produce the genetic offspring population. An example of this crossover and mutation operator is illustrated in Fig. 4.3. The crossover operator produces two children. Each child preserves part of the elements of one parent, while elements of another parent (excluding those preserved elements by the child) fill the remaining parts of this child from left to right. The mutation randomly swaps two elements of one parent to produce a new permutation.

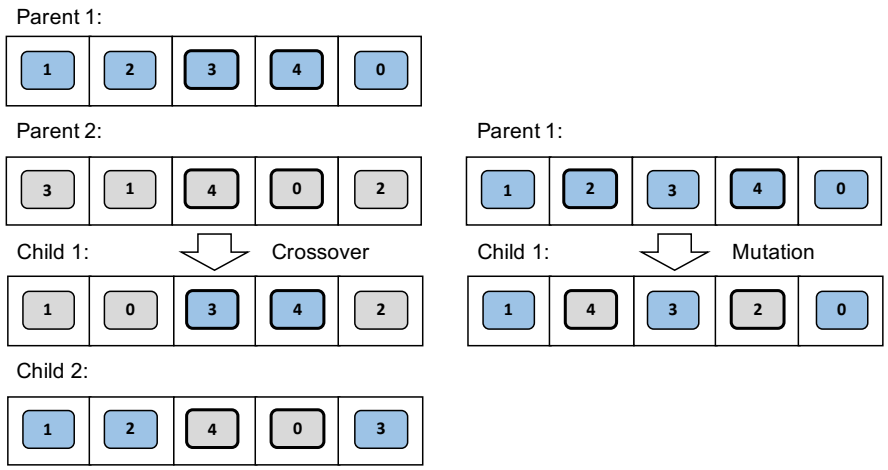


Fig. 4.3: Examples of crossover and mutation for parents.

We produce genetic offspring population $\mathcal{P}_a^g$ more efficiently than that in Hybrid and Hybrid-L [40]. Although two children are produced by one crossover, only one child associated with a higher Tchebycheff score [229] will be added to the offspring population in both Hybrid and Hybrid-L [40]. Compared to [40], we put both children in population $\mathcal{P}_a^g$. Therefore, to produce an offspring population with an equal size, our method is more efficient than Hybrid and Hybrid-L [40].

4.4.4 Identify a Cluster Representative of Each Cluster

Unlike single-objective optimization problems discussed in Chapter 3, it is not straightforward to determine promising solutions for learning NHM in EDA under the multi-objective optimization setting since they often have two objectives. To select a small number of promising solutions, we propose to define one cluster representative as a promising solution based on the dominance relationships among all solutions in the cluster. In particular, we cluster d groups of close individuals in one generation using an existing clustering technique, i.e., K-means++ [10]. The sensitivity of parameter d is studied in Sect. 4.5.1. We infer a group of individuals that represents close similarities measured by fitness values, f_1 and f_2 in Eq. (4.2) and (4.3). With equal probability, we choose one solution that is not dominated by any other solutions of the same cluster as a cluster representative. Consequently, we can learn an NHM based on the cluster representative, see details in Sect. 4.4.5.

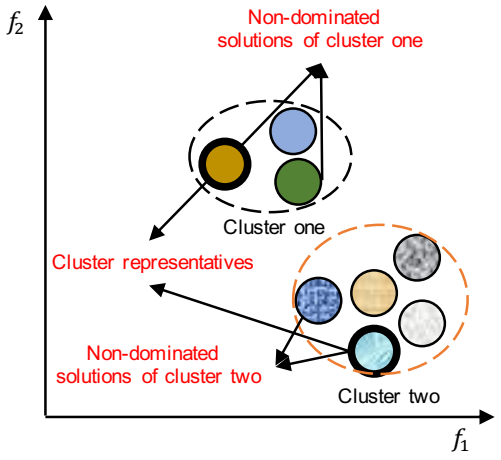


Fig. 4.4: An example of identifying two cluster representatives.

An example of identifying promising solutions via a clustering technique is illustrated in Fig. 4.4. In Fig. 4.4, we consider one population of eight individuals, which are clustered into two groups of individuals

based on their objective values using K-mean++. Subsequently, for each cluster, we can randomly pick up one non-dominant solution as the cluster representative, see the marked cluster representatives in Fig. 4.4.

4.4.5 Learn a NHM Based on Cluster Representatives

In this subsection, we propose a method to learn a suitable distribution model (i.e. NHM) based on the cluster representative with respect to each cluster. This method consists of two main steps: permutation transformation and NHM learning.

We transfer every cluster member Π_k^g into a new permutation $\Pi^{\star g}_k$ based on its decoded DAG form $\mathcal{G}_k^g$. The transformation process is exactly the same as the process of producing the novel permutation in Sect. 3.5.2. After the transformation, we can now learn a NHM based on the cluster representative formed in this permutation. Different from the way of learning a NHM in Sect. 3.5.3, we propose a new way of learning a NMH, improving the chance of local improvements on the cluster representatives through sampling. In particular, we use the Euclidean distances between the cluster representative and other members of this cluster to weight the influences of every cluster member on the NHM. This is because cluster members far from cluster representative contribute less to the NHM.

The *node histogram matrix* (NHM) for the cl^{th} cluster with the cluster representative Rep^{cl} in generation g is denoted as $\mathcal{NHM}_{cl}^g$, which is an $n \times n$-matrix with entries $e_{i,j}$ as follows:

$$e_{i,j} = \sum_{k=0}^{m-1} \delta_{i,j}(\Pi^{\star g}_k) + \varepsilon \tag{4.4}$$

$$\delta_{i,j}(\Pi^{\star g}_k) = \begin{cases} w(\Pi^{\star g}_k) & \text{if } \pi_i = \text{j} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

$$w(\Pi^{\star g}_k) = 1 - ||\vec{f}(\Pi^{\star g}_k) - \vec{f}(Rep^g_{cl})||_2 \tag{4.6}$$

where $i, j = 0, 1, \ldots, n-1$, $\varepsilon = \frac{m}{n-1} b_{ratio}$ is a predetermined bias, and $||\vec{f}(\Pi^{\star g}_k) - \vec{f}(Rep^g_{cl})||_2$ measures a Euclidean distance between one cluster member and the cluster representative. This distance value is offset by 1, so the higher values correspond to less weights in learning an NHM. Roughly speaking, entry $e_{i,j}$ counts how often service index π_i appears in position j of all the permutations in the cl^{th} cluster. Meanwhile, the weight of the frequency is penalized by Eq. (4.6). Afterwards, we can use NHBSA [168] to sample local search offspring population $\mathcal{P}^g_b$ for generation $g+1$, using ALGORITHM 12 in Sect. 3.5.3

4.5 Experimental Evaluation

We conduct experiments for studying the performance of our MNSGA2-EDA approach using the new benchmarks, WSC-08 and WSC-09, proposed in Chapter 3 to demonstrate that MNSGA2-EDA can maintain its high performance on our problem with significantly larger sizes. We first investigate the sensitivity of parameters, i.e., the number of clusters d and their sampling size. In particular, we investigate three groups of settings with increasing size of $\mathcal{P}^g_b$ (see details in Sect. 4.5.1). The following experiment further investigates the effectiveness and efficiency of MNSGA2-EDA in comparison to the baseline method NSGA-II and to Hybrid and Hybrid-L [40]. These approaches have recently been proposed to solve a similar multi-objective fully automated service composition problem. Note that Hybrid-L has employed a so-called swap operator on numerous decomposed subproblems for local serach. However, Hybrid-L exhibit very poor convergence rate in practice. We use two tasks WSC09-3 and WSC09-5 to exemplify the poor performance of Hybrid-L in Fig. 4.5 and 4.6. Therefore, we do not further report on the performance of Hybrid-L for the remaining tasks, when compared to MNSGA2-EDA.

We follow the settings in [40] for all approaches, where the size of both $\mathcal{P}^g$ and $\mathcal{P}^g_a$ are set to 500. The maximum generation g is 51, and the prob-

ability rates of crossover, mutation, and reproduction are 0.8, 0.1 and 0.1. For EDA settings in MNSGA2-EDA, b_{ratio} of ε is set to 0.0002 according to our suggestion in Sect 3.7. The weights in the fitness function Eq. (4.2) and Eq. (4.3) are set to balance quality criteria in both QoSM and QoS, i.e., w_1 and w_2 are set to 0.5, and w_3, w_4, w_5 and w_6 to 0.25. We have also conducted tests with other weights and parameters and generally observed the same behavior.

IGD and *hypervolume* are commonly used performance evaluation metrics for multi-objective optimization algorithms [84]. IGD measures the distance from the nearest point of the non-dominated set produced by an approach to an approximated true Pareto front obtained by using all approaches. Hypervolume measures the dominated volume covered by a reference point (e.g., a point (1,1) is chosen in our case) and the front evolved by each algorithm. In particular, we highlight IGD and hypervolume values of all the top performances in a green colour for all the competing algorithms.

4.5.1 Parameters sensitivity

To determine suitable parameters of EDA-based local search in MNSGA2-EDA, we use task WSC08-3 to perform parameters sensitivity tests over a set of parameters with an increasing size of local search offspring population $\mathcal{P}_b^g$ in MNSGA2-EDA.

We use Wilcoxon rank-sum testing with a significance level of 5% to verify the observed differences in *IGD* and *hypervolume* over 30 runs. This test method is also used consistently to detect any noticeable differences in the experiment results in Sect. 4.5.2.

In Table 4.1 and 4.2, the first column shows the size of $\mathcal{P}_b^g$. The second and third column of Table 4.1 and 4.2 show a pair of parameters used in EDA, which are the number of clusters d and their sampling size. The fourth column of Table 4.1 and 4.2 show the mean values of IGD and hy-

Table 4.1: Mean IGD of MNSGA2-EDA with three groups of parameter settings over WSC08-3 (Note: the lower the IGD the better).

Size of $\mathcal{P}_b^g$	d	Sampling size	MNSGA2-EDA
160	2	80	$6e-04 \pm 3e-04$
	4	40	$2e-04 \pm 0$
	6	27	$2e-04 \pm 1e-04$
200	2	100	$4e-04 \pm 2e-04$
	4	50	$1e-04 \pm 0$
	6	34	$2e-04 \pm 1e-04$
240	2	120	$4e-04 \pm 3e-04$
	4	60	$1e-04 \pm 1e-04$
	6	40	$1e-04 \pm 0$

Table 4.2: Mean Hypervolume of MNSGA2-EDA with three groups of parameter settings over WSC08-03 (Note: the higher the hypervolume the better).

Size of $\mathcal{P}_b^g$	d	Sampling size	MNSGA2-EDA
160	2	80	$0.2302 \pm 1e-04$
	4	40	$0.2304 \pm 1e-04$
	6	27	$0.2304 \pm 1e-04$
200	2	100	$0.2303 \pm 1e-04$
	4	50	0.2305 ± 0
	6	34	$0.2305 \pm 1e-04$
240	2	120	$0.2303 \pm 1e-04$
	4	60	$0.2305 \pm 1e-04$
	6	40	0.2305 ± 0

pervolume and the corresponding standard deviation over 30 repetitions.

Table 4.1 and 4.2 show that 200 local search offspring population size based on 4 clusters with 50 sampling size is the best-found parameter set-

ting over all designed parameter settings for task WSC08-3. As shown in Table 4.1 and 4.2, MNSGA2-EDA with this setting is highlighted as one top performing algorithm in terms of both mean IGD and hypervolume. We will use this setting in Sect. 4.5.2.

4.5.2 Comparing MNSGA2-EDA with NSGA-II, Hybrid and Hybrid-L

Comparison of the Execution Time

Table 4.3 shows the mean execution times (in seconds) and the standard deviation observed for the three methods MNSGA2-EDA, NSGA-II and Hybrid over 30 repetitions. More specifically, Table 4.4 summarizes the results of pairwise comparisons of the three methods without Bonferroni correction. The table displays win/draw/loss of one method compared to all other methods. That is, it is reported how often one method outperforms, equals or is outperformed by the competing method.

The mean execution time for MNSGA2-EDA and NSGA-II are very comparable (but not equal) to each other for tasks in WSC08 and WSC09. In comparison, Hybrid consistently takes twice the execution time for each task. This observation does not agree with the findings in [40] that Hybrid and NSGA-II achieve competitive execution time. This is because they do not point out the assumption that the evaluation time of every candidate solution is indistinct. In our thesis, a more challenging benchmark is utilized for testing, and high computation resources are required for computing QoSM of each solution. In addition, every crossover operator in Hybrid requires two evaluations of two produced children in order to keep a child with a higher Tchebycheff score. Instead, MNSGA2-EDA and NSGA-II keep both of the two children. For example, 500 children are required for the next generation using crossover, then Hybrid requires 1000 evaluations while MNSGA2-EDA and NSGA-II only require 500 evaluations. Therefore, Hybrid consumes much more execution time than MNSGA2-EDA and NSGA-II.

Table 4.3: Mean execution time (in seconds) for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the shorter the time the better).

Task	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08-1	224 ± 12	190 ± 48	418 ± 65
WSC08-2	81 ± 17	58 ± 14	139 ± 32
WSC08-3	5539 ± 464	8095 ± 1437	20793 ± 4149
WSC08-4	210 ± 17	317 ± 58	805 ± 147
WSC08-5	4242 ± 562	6090 ± 1704	14735 ± 5166
WSC08-6	62966 ± 10943	65051 ± 8592	158737 ± 27171
WSC08-7	5489 ± 814	9132 ± 2578	23074 ± 6030
WSC08-8	9917 ± 3788	12443 ± 1818	33077 ± 6164
WSC09-1	198 ± 67	155 ± 76	327 ± 90
WSC09-2	5634 ± 679	6139 ± 1678	14634 ± 2816
WSC09-3	2968 ± 301	2820 ± 714	6527 ± 2403
WSC09-4	269207 ± 23542	255195 ± 28813	646897 ± 117538
WSC09-5	39370 ± 5125	35338 ± 8350	86281 ± 19944

Table 4.4: Summary of statistical significance tests for the execution time, where each column shows the win/draw/loss score of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08 (8 tasks)	MNSGA2-EDA	-	2/1/5	0/0/8
	NSGA-II	5/1/2	-	0/0/8
	Hybrid [40]	8/0/0	8/0/0	-
WSC09 (5 tasks)	MNSGA2-EDA	-	3/2/0	0/0/5
	NSGA-II	0/2/3	-	0/0/5
	Hybrid [40]	5/0/0	5/0/0	-

Comparison of the IGD and Hypervolume

Tables 4.5 to 4.8 show the mean IGD and the mean hypervolume with the standard deviations over 30 repetitions, and the corresponding summarized results of pair comparisons for MNSGA2-EDA, NSGA-II, and Hybrid. We note that MNSGA2-EDA achieves the best-known values of IGD for all tasks except for one task (i.e., WSC09-1) and the best-known values of hypervolume for all tasks. On the other hand, NSGA-II only achieves the best-known values of both IGD and hypervolume for 2 of the 13 tasks, while Hybrid only obtained the best-known values of IGD and hypervolume for 4 out of the 13 tasks and 3 out of the 13 tasks respectively. Therefore, MNSGA2-EDA significantly outperforms NSGA-II and Hybrid measured by IGD and hypervolume.

Table 4.5: Mean IGD for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the lower the IGD the better).

Task	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08-1	0 ± 0	$1e-04 \pm 7e-04$	$1e-04 \pm 5e-04$
WSC08-2	0 ± 0	0 ± 0	0 ± 0
WSC08-3	$1e-04 \pm 0$	$0.001 \pm 4e-04$	$0.001 \pm 3e-04$
WSC08-4	0 ± 0	$3e-04 \pm 3e-04$	$1e-04 \pm 1e-04$
WSC08-5	0.0029 ± 0.0014	0.0043 ± 0.0015	0.0027 ± 0.0011
WSC08-6	$7e-04 \pm 3e-04$	$0.0014 \pm 3e-04$	$0.0012 \pm 3e-04$
WSC08-7	$1e-04 \pm 2e-04$	$0.002 \pm 9e-04$	0.0015 ± 0.001
WSC08-8	$0 \pm 1e-04$	$9e-04 \pm 5e-04$	$6e-04 \pm 3e-04$
WSC09-1	0.0701 ± 0.0132	$0.0731 \pm 6e-04$	0.0654 ± 0.0199
WSC09-2	0.0055 ± 0.001	0.0065 ± 0.0011	$0.0061 \pm 9e-04$
WSC09-3	$0.002 \pm 9e-04$	0.0126 ± 0.0085	0.0107 ± 0.0076
WSC09-4	0.0025 ± 0.001	$0.0061 \pm 7e-04$	0.0056 ± 0.0012
WSC09-5	0.0025 ± 0.0014	0.0052 ± 0.0011	$0.0045 \pm 7e-04$

Table 4.6: Summary of statistical significance tests for IGD, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08 (8 tasks)	MNSGA2-EDA	-	0/2/6	0/3/5
	NSGA-II	6/2/0	-	4/4/0
	Hybrid [40]	5/3/0	0/4/4	-
WSC09 (5 tasks)	MNSGA2-EDA	-	0/0/5	1/0/4
	NSGA-II	5/0/0	-	2/3/0
	Hybrid [40]	4/0/1	0/3/2	-

Table 4.7: Mean Hypervolume for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the higher the hypervolume the better).

Task	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08-1	0.3825 ± 0	$0.3824 \pm 4e-04$	$0.3825 \pm 1e-04$
WSC08-2	0.5798 ± 0	0.5798 ± 0	0.5798 ± 0
WSC08-3	0.2305 ± 0	$0.2298 \pm 2e-04$	$0.23 \pm 1e-04$
WSC08-4	0.3217 ± 0	$0.3213 \pm 7e-04$	$0.3215 \pm 5e-04$
WSC08-5	$0.278 \pm 7e-04$	0.2752 ± 0.0022	0.2767 ± 0.0014
WSC08-6	$0.2341 \pm 1e-04$	$0.2338 \pm 2e-04$	$0.2341 \pm 2e-04$
WSC08-7	$0.2808 \pm 2e-04$	0.278 ± 0.0014	0.2788 ± 0.0014
WSC08-8	$0.2475 \pm 1e-04$	$0.2465 \pm 7e-04$	$0.2471 \pm 4e-04$
WSC09-1	0.4435 ± 0.0028	$0.4424 \pm 9e-04$	0.4434 ± 0.0031
WSC09-2	$0.2751 \pm 1e-04$	0.2742 ± 0.0016	$0.2747 \pm 7e-04$
WSC09-3	$0.3693 \pm 1e-04$	0.361 ± 0.0064	0.3618 ± 0.0054
WSC09-4	0.239 ± 0.0014	$0.2346 \pm 9e-04$	0.2355 ± 0.0017
WSC09-5	0.2376 ± 0.001	$0.235 \pm 5e-04$	$0.2353 \pm 5e-04$

Table 4.8: Summary of the statistical significance tests for hypervolume, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [40]
WSC08 (8 tasks)	MNSGA2-EDA	-	0/2/6	0/3/5
	NSGA-II	6/2/0	-	5/3/0
	Hybrid [40]	5/3/0	0/3/5	-
WSC09 (5 tasks)	MNSGA2-EDA	-	0/0/5	0/0/5
	NSGA-II	5/0/0	-	2/3/0
	Hybrid [40]	5/0/0	0/3/2	-

Comparison of the Convergence Curve

To investigate the effectiveness and scalability of MNSGA2-EDA, NSGA-II, Hybrid and Hybrid-L, we further investigate the convergence curves for IDG and hypervolume over 30 repetitions using WSC09-3 and WSC09-5 as two examples.

Fig. 4.6 and 4.5 depict the evolution of the mean values of the IGD and hypervolume over the mean execution time for MNSGA2-EDA, NSGA-II, Hybrid, and Hybrid-L. We cut the mean execution time to fit the maximal required time of Hybrid because Hybrid-L results in a much higher order of magnitude in execution time, and it does not have a chance to catch up with MNSGA2-EDA. For Hybrid, it converges much better than Hybrid-L, but does not converge better than the baseline NSGA-II. In contrast, our MNSGA2-EDA approach achieves significantly better IGD and hypervolume values with the fastest convergence rate.

Comparison of the Pareto Optimal Solutions

In Fig. 4.7, we present a plot of the Pareto optimal solutions of WSC09-3 and WSC09-5 obtained by MNSGA2-EDA, NSGA-II and Hybrid over 30 independent runs. The Pareto optimal solutions are identified based

Fig. 4.5: Mean hypervolume over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right)
(Note: the larger the hypervolume the better).

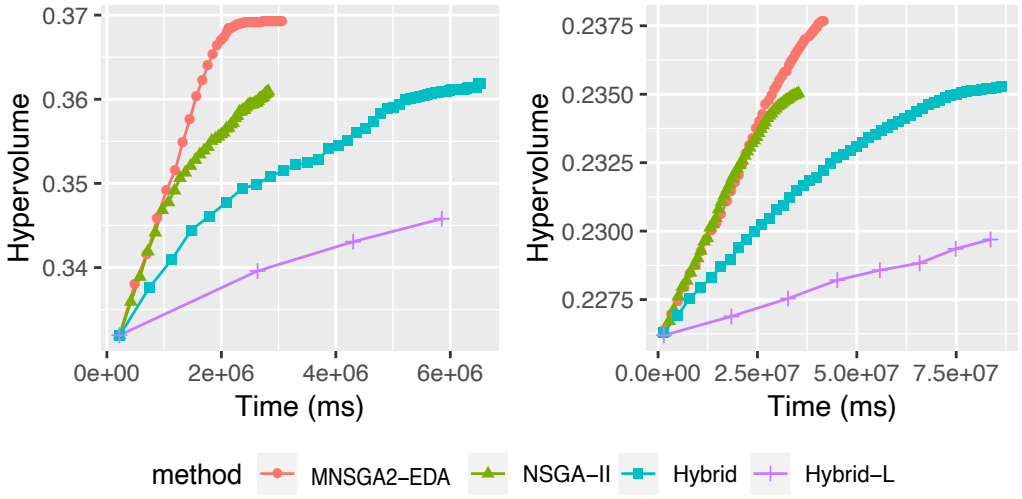
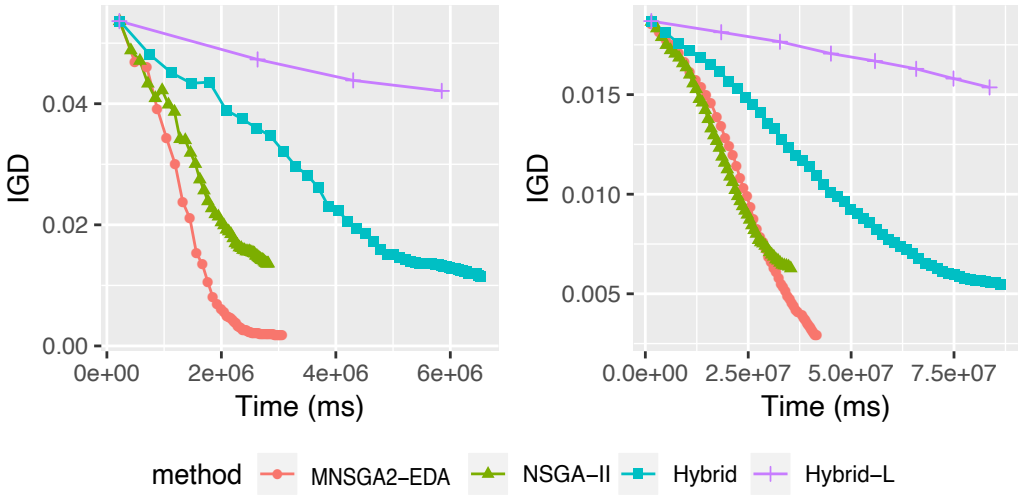
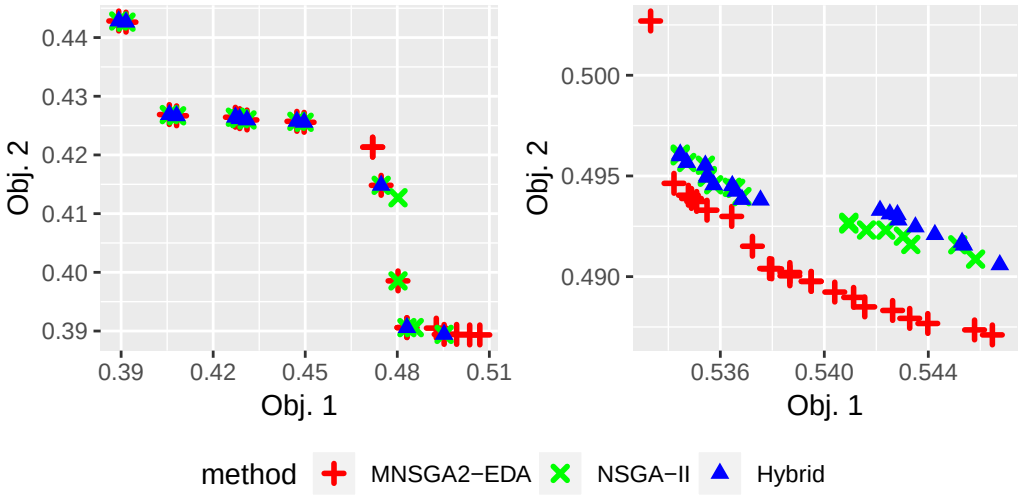


Fig. 4.6: Mean IGD over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right) (Note: the smaller the IGD the better).



on the combined results of all 30 runs of each method. It is easy to observe that the Pareto front generated by MNSGA2-EDA is much more widely distributed. In other words, extreme solutions are more likely to be found by MNSGA2-EDA. For task WSC09-3, a trade-off solution at the knee point of the Pareto front is found by MNSGA2-EDA. We hasten to point out that it is highly important and desirable to discover a solution like this. The other two methods (NSGA-II and Hybrid) fail to discover this solution, which may be regarded as a weakness. For task WSC09-05, much better Pareto optimal solutions are obtained by MNSGA2-EDA, and these solutions consistently dominate all solutions obtained by other methods.

Fig. 4.7: Pareto optimal solutions obtained for tasks WSC09-3 (left) and WSC09-5 (right).



4.6 The WSC-MQP Problem

We perceive WSC-MQP as a multitasking problem that aims to optimize K composition tasks concurrently with respect to the K user segments.

WSC-MQP has not been explicitly studied before, according to the literature review discussed in Sect. 2.2. Some concepts related to MFEA, such as *factorial cost* f_j^{Π}, *factorial rank* r_j^{Π}, *scalar fitness* φ^{Π} and *skill factor* of a individual Π, introduced in Sect. 2.1.3, are also utilized in this chapter.

We extend *composition task* defined in Sect. 3.3 to cover distinctive QoSM preferences from K user segments. The preferences of one user segment is defined as an interval, such as $QoSM \in (0.75, 0.1]$. Therefore, a *composition task* (also called a *service request*) over a given *service repository* is a tuple $T_j = (I_T, O_T, cons_j)$ where I_T is a set of task inputs, and O_T is a set of task outputs. The inputs in I_T and outputs in O_T are parameters that are semantically described by concepts in a ontology $\mathcal{O}$. $cons_j$ is a QoSM preference, where $cons_j$ is in the range of $(QoSM_j^a, QoSM_j^b]$, $j \in \{1, 2, \ldots, K\}$ and $QoSM_j^a, QoSM_j^b$ are lower and upper bounds of QoSM that are unique for each user segment.

It is essential to include infeasible individuals (i.e., solutions that violate the preference of one task) into a population, since infeasible composite services may help to find optimal solutions of other tasks. For example, one solution is infeasible for T_1 as it violates $cons_1$, but it is feasible to T_2 as it complies with $cons_2$. This solution should be included for finding optimal solutions for T_2. Therefore, we allow infeasible individuals in the population, but their fitness must be penalized. According to the fitness function in Eq. (4.7) with respect to T_j, we guarantee that f_j^{Π} of an infeasible individual Π stays below 0.5 while f_j^{Π} of a feasible individual falls above 0.5. Eq. (4.8) measures the comprehensive quality of a individual Π (see our previous discussion in Sect. 3.3). Eq. (4.9) measures QoSM of a individual Π. Based on Eq. (4.9), Eq. (4.10) measures the degree of violation of $cons_j$ by calculating how far it is from $QoSM(\Pi)$. In particular, an infeasible individual that violates $cons_j$ more should be penalized more.

$$f_j^{\Pi} = \begin{cases} 0.5 + 0.5 * F(\Pi) & \text{if } QoSM(\Pi) \in interval_j, \\ 0.5 * F(\Pi) - 0.5 * V_j(\Pi) & \text{otherwise.} \end{cases} \tag{4.7}$$

$$F(\Pi) = w_1\hat{MT} + w_2\hat{SIM} + w_3\hat{A} + w_4\hat{R} + w_5(1-\hat{T}) + w_6(1-\hat{CT}) \tag{4.8}$$

$$QoSM(\Pi) = w_7\hat{MT} + w_8\hat{SIM} \tag{4.9}$$

$$V_j(\Pi) = \begin{cases} QoSM_j^a - QoSM(\Pi) & \text{if } QoSM(\Pi) \leq QoSM_j^a, \\ QoSM(\Pi) - QoSM_j^b & \text{otherwise.} \end{cases} \tag{4.10}$$

with $\sum_{k=1}^{6} w_k = 1$ $(w_k \geqslant 0)$ and $\sum_{k=7}^{8} w_k = 1$ $(w_k \geqslant 0)$. We can adjust the weights according to the preferences of each user segment. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$, $\hat{R}$, $\hat{T}$, and $\hat{CT}$ are normalized values calculated within the range from 0 to 1 using Eq. (3.6) (see details in Sect. 3.3). The goal of WSC-MQP is to find the K best possible solutions with one for each task, our goal is to maximize the objective function in Eq. (4.7) concerning K tasks.

4.7 The PMFEA Algorithm

In this section, we present PMFEA to solve the **WSC-MQP** problem. We begin with an outline of PMFEA in Sect. 4.7.1. Afterwards, we discuss some critical components of PMFEA from Sect. 4.7.2 to 4.7.4.

Our proposed PMFEA is characterized by three novel aspects. Firstly, we employ a permutation-based representation for composite services to establish a common search space over K composition tasks (see details in Sect. 4.7.2). As discussed in FL [47], their permutation-based representation has shown its promises in solving single-tasking single-objective service composition problems. Therefore, we will study the effectiveness of this permutation in solving the WSC-MQP problem. In addition, the permutation-based crossover and mutation operators, discussed in Sect. 4.4.3, are studied for their effectiveness in the assortative mating (see details in Sect. 4.7.3).

Secondly, we introduce a neighbourhood structure over multiple tasks for more effectively evolving solutions in PMFEA. By evaluating evolved solutions on neighbouring tasks, we increase the chance for a solution evolved for one inherited task (determined through vertical culture transmission) to participate in building the solutions of related tasks. In our problem, the related tasks are tasks whose QoSM preferences are adjacent to those of the inherited task. It is through this way that knowledge can be exchanged effectively across multiple tasks, enabling our algorithm to effectively cope with a problem with more than two concurrent composition requests (see details in Sect. 4.7.4).

Thirdly, we show that fitness evaluations of a solution on neighboring tasks are efficient in **WSC-MQP** because once the calculation of $F(\Pi)$ in Eq. (4.8) (i.e., a time-consuming calculation) is completed for the evaluation on the inherited task, it is not required to be calculated again for the evaluations on the neighboring tasks. Moreover, we expect the overall execution time of PMFEA can be reduced by performing the evaluations on neighbouring tasks. Due to the effective knowledge transformation across different tasks through the introduced neighborhood structure, we increase the chances of evolving effective solutions through assortative mating. Therefore, the process of decoding permutations could be accelerated by the better knowledge transformation (see details in Sect. 4.7.2).

4.7.1 Outline of PMFEA

The overview of PMFEA is summarized in ALGORITHM 19: we initially generate m random permutations Π_k^g, where $0 \leq k < m$ and $g = 0$. Each permutation will be decoded into a DAG-based solution, $\mathcal{G}_k^g$ (see the details in Sect. 4.7.2). We evaluate $f^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over T_j, where $j \in \{1, 2, \ldots, K\}$. Subsequently, the following steps (Step 4 to 9) are repeated until a maximum generation g_{max} is reached. During the iteration, we apply assortative mating with crossover and mutation to breed offspring population $\mathcal{P}_a^g$ (see details in Sect. 4.7.3). Once $\mathcal{P}_a^g$ is generated,

ALGORITHM 19. PMFEA for WSC-MQP.

Input : T_j, K, and g_{max}
Output: $\{\Pi_j^{opt}\}_{j=1}^{K}$

1: Randomly initialize population $\mathcal{P}^g$ of m permutations Π_k^g as solutions (where $g = 0$ and $k = 1, \ldots, m$);
2: Decode each Π_k^g into DAG $\mathcal{G}_k^g$ using a forward graph-building technique;
3: Evaluate $f^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over T_j, where $j \in \{1, 2, \ldots, K\}$;
4: **while** $g < g_{max}$ **do**
5: Apply assortative mating to the randomly selected individuals to generate offspring population $\mathcal{P}_a^{g+1}$;
6: Assign offspring in $\mathcal{P}_a^{g+1}$ to the selected tasks and evaluate $f^{\Pi_k^{g+1}}$ on the tasks;
7: $\mathcal{P}^{g+1} = \mathcal{P}^g \cup \mathcal{P}_a^{g+1}$;
8: Update $r_j^{\Pi_k^{g+1}}$, $\varphi^{\Pi_k^{g+1}}$ and $\tau^{\Pi_k^{g+1}}$ of offspring in $\mathcal{P}^{g+1}$;
9: Keep top half of the fittest individuals in $\mathcal{P}^{g+1}$ based on $\varphi^{\Pi_k^{g+1}}$;
10: Let $\{\Pi_j^{opt}\}_{j=1}^{K}$ be a set of best solutions over all the generations for the K tasks;

individuals in $\mathcal{P}_a^g$ will be assigned to tasks based on vertical cultural transmission (see detail in ALGORITHM 8 in Sect. 2.1.3) and the corresponding neighbouring tasks for evaluations (see details in Sect. 4.7.4). Consequently, we produce the next population $\mathcal{P}^{g+1}$ by combining the current population $\mathcal{P}^g$ and assortative mating offspring population $\mathcal{P}_a^g$. We update $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of the combined population in $\mathcal{P}^{g+1}$, and keep the top half of fittest individuals in $\mathcal{P}^{g+1}$ based on $\varphi^{\Pi_k^g}$. In each generation, we keep track of the fittest Π_j^{opt} for each task T_j. When the maximal generation g_{max} is met, we return a set of best solutions $\{\Pi_j^{opt}\}_{j=1}^{K}$ for the K tasks over all the generations .

4.7.2 Permutation-based representation

Different from the novel permutation introduced in Sect. 3.5.2, we simply utilize random permutations introduced in FL [47]. Let $\Pi = (\pi_1, \ldots, \pi_t, \ldots, \pi_n)$ be a permutation-based composite service of service indexes $\{1, \ldots, t, \ldots, n\}$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. Permutations must be decoded into DAG-based solutions for easy of calculating the factorial cost and presenting users a final execution workflow.

Fig. 4.8 illustrates an example of producing a DAG-based solution decoded from a permutation using a forward graph-building technique [175]. In the example, we take an arbitrary permutation $[4, 3, 5, 1, 2]$ as an example with composition task inputs I_T and outputs O_T. We check the permutation from left to right, looking for services whose inputs can be fulfilled by I_T, so we remove them from the permutation and add them to the graph. Afterwards, we go through the permutation from left to right again and add services whose inputs can be fulfilled by I_T and any outputs of services in the graph. We continue this process until we can add End to the graph (i.e., O_T can be produced). Note that this process may result in graphs that contain some services whose outputs does not contribute to the satisfactions of the inputs of End, such as service S_4. These services will be removed.

4.7.3 Assortative Mating

PMFEA employs assortative mating to generate offspring for K tasks. In particular, two randomly selected parent candidates undergo crossover if they have the same skill factors. Otherwise, a pre-defined probability $rand$ is used to decide when knowledge should be implicitly shared across tasks: crossover is performed over the parent candidates with different skill factors or mutation is performed on each parent, see ALGORITHM 7 in Sect. 2.1.3 for technical details. Note that the order crossover and one-point swap mutation, discussed in Sect. 4.4.3, are employed for the purpose of

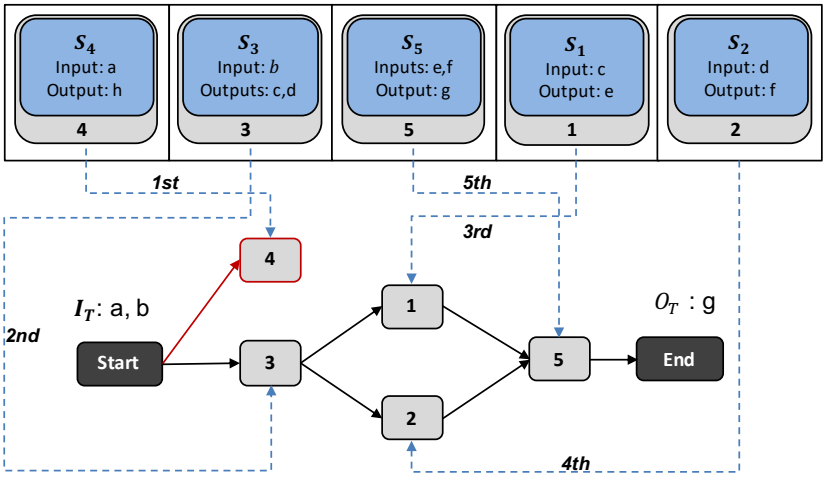


Fig. 4.8: An example of a DAG-based solution decoded from a given permutation.

assortative mating to generate permutations for K tasks.

4.7.4 Task Selection for Evaluations

MFEA [71] suggest that candidate solutions are only evaluated on the task that is inherited from parents based on the vertical cultural transmission, see details in ALGORITHM 8 in Sect. 2.1.3.

To effectively deal with more than two optimization tasks, we propose PMFEA-NT for evolving more effective solutions through careful selections of tasks. In particular, we introduce a neighbourhood structure over a set of tasks. We suggest identifying the neighbouring tasks of each child's inherited task, which is decided by vertical culture transmission. Subsequently, we will assign each child to the neighbouring tasks for additional evaluations.

Fig. 4.9 illustrates an example of neighborhood structure over four composition tasks T_1, T_2, T_3 and T_4 with respect to four user segments. These four composition tasks have the same input and output (i.e., I_T and I_O) but different $cons_j$, where $j \in \{1, 2, 3, 4\}$. In particular, $cons_1 \in (0, 0.25]$,

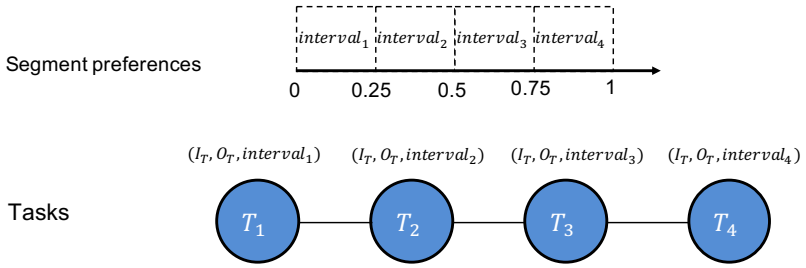


Fig. 4.9: An example of neighborhood structure over four tasks.

$cons_2 \in (0.25, 0.5]$, $cons_3 \in (0.5, 0.75]$ and $cons_4 \in (0.75, 1]$, respectively. The neighborhood structure is manually determined by following a rule, i.e., tasks whose segment preferences on QoSM are adjacent to each other are neighbours. For example, the neighboring tasks of task T_2 are T_1 and T_3 whose segment preference on QoSM (i.e., $cons_1$ and $cons_3$) are adjacent to that of T_2 (i.e., $cons_2$).

We continue to demonstrate the benefits of our proposed neighbourhood structure in PMFEA-NT using the example in Fig. 4.9. Consider a child derived from a parent that satisfies $cons_1$ of T_1, and this child can also lead to the satisfaction of a neighbouring segment preference, i.e., $cons_2$. If this child is only evaluated on task T_1 based on the vertical cultural transmission, resulting in a poor fitness value, it is likely to be discarded. In contrast, if we give this child a chance to be evaluated on the neighbouring task, i.e., T_2, resulting in a good fitness value, it can survive to the next generation due to its good performance on T_2. We hope such a situation will help to diversify solutions in the population and make the evolution process more effective.

4.8 The PMFEA-EDA Algorithm

We first present an outline of PMFEA-EDA for WSC-MQP in Sect. 4.8.1. Subsequently, we will discuss the two main innovations of this method: (1) constructing single-tasking and multitasking NHMs for effective ex-

ploration of the solution space over multiple tasks in every generation, and (2) a new sampling mechanism designed to balance the trade-off between exploration and exploitation in a multitasking context.

To learn a single-tasking NHM with respect to each task, we assign composite services to different solution pools based on their skill factors. Therefore, every solution pool stores promising solutions for one task. As discussed in Sect. 4.7.4, solutions that are promising for one task can be used to evolve new solutions for its adjacent tasks (i.e., neighbouring tasks whose QoS preferences are close to each other). Due to this reason, we also prepare additional solution pools to store solutions that are promising for every two adjacent tasks. In other words, every pool contains solutions whose skill factors are associated with two adjacent tasks. We assume that every two adjacent tasks are the most suitable tasks for the knowledge sharing. Therefore, learning multitasking NHMs of these additional pools allow knowledge to be shared across adjacent tasks (see details in Sect. 4.8.2).

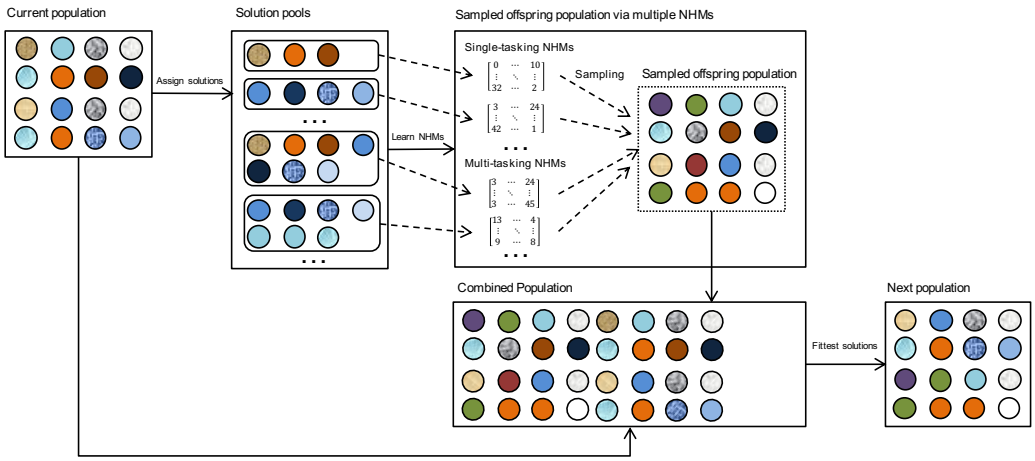


Fig. 4.10: Generation updates in PMFEA-EDA.

Moreover, we propose a sampling mechanism to balance exploration and exploitation. Particularly, a random sampling probability (rsp) is predefined to determine which NHM will be used to build new solutions.

This mechanism is inspired by assortative mating [71], where a different random probability is defined as the occurrence of crossover on two parent solutions from the same skill factor or different skill factors.

The generation updates in PMFEA-EDA are illustrated in Fig. 4.10. From the current population in Fig. 4.10, one sampled offspring population is created and further combined with the current population to produce the next population by only keeping the fittest solutions. Particularly, this sampled offspring population is formed from new solutions that are sampled from both single-tasking and multitasking NHMs. These NHMs are learned from multiple solution pools consisting of solutions assigned based on their skill factors.

4.8.1 Outline of PMFEA-EDA

The outline of PMFEA-EDA is shown in ALGORITHM 20. We first randomly initialize m permutation-based Π_k^g solutions, where $0 \leq k < m$ and $g = 0$. Each permutation-based solution will be decoded into a DAG-based solution $\mathcal{G}_k^g$ for interpreting its service execution workflow. Based on $\mathcal{G}_k^g$, we can easily determine $f_j^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over task T_j, where $j \in \{1, 2, \ldots, K\}$. Afterwards, we encode each solution Π_k^g in $\mathcal{P}^g$ into another permutation $\Pi^{\star g}_k$ based on its decoded DAG form $\mathcal{G}_k^g$. Note that the decoding and encoding process for producing permutations is exactly the same as that discussed in Sect.3.5.2. This newly decoded representation enables reliable and accurate learning of a NHM over varied composition workflows in the fully automated service composition. The iterative part of PMFEA-EDA comprises of Step 6 to 12, which is repeated until a maximum generation g_{max} is reached. During each iteration, we generate an offspring population $\mathcal{P}_a^{g+1}$ via multiple NMHs using ALGORITHM 21 (see details in Sect. 4.8.2). Again, the same decoding and encoding process is employed on the newly sampled solutions in $\mathcal{P}_a^{g+1}$. Afterwards, we evaluate the factorial cost $f^{\Pi_k^{g+1}}$ of solutions in $\mathcal{P}_a^{g+1}$ on the task related to the imitated tasks skill factor. In particular, the skill factor of every newly

ALGORITHM 20. PMFEA-EDA for WSC-MQP.

Input : T_j, K, and g_{max}
Output: $\{\Pi_j^{opt}\}_{j=1}^{K}$

1: Randomly initialize population $\mathcal{P}^g$ of m permutations Π_k^g as solutions (where $g = 0$ and $k = 1, \ldots, m$);
2: Decode each Π_k^g into DAG $\mathcal{G}_k^g$ using a decoding method;
3: Calculate $f_j^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over T_j, where $j \in \{1, 2, \ldots, K\}$;
4: Encode each solution Π_k^g in $\mathcal{P}^g$ with another permutation $\Pi^{\star g}_k$;
5: **while** $g < g_{max}$ **do**
6: Generate offspring population $\mathcal{P}_a^{g+1}$ via multiple NHMs learning and sampling using ALGORITHM 21;
7: Decode solutions in $\mathcal{P}_a^{g+1}$ into DAG $\mathcal{G}_k^{g+1}$ using a decoding method;
8: Calculate $f^{\Pi_k^{g+1}}$ of solutions in $\mathcal{P}_a^{g+1}$ on the selected tasks related to the skill factors determined in its corresponding NHM;
9: Encode each solution Π_k^g in $\mathcal{P}^g$ with an another permutation $\Pi^{\star g}_k$;
10: $\mathcal{P}^{g+1} = \mathcal{P}^g \cup \mathcal{P}_a^{g+1}$;
11: Update $r_j^{\Pi_k^{g+1}}$, $\varphi^{\Pi_k^{g+1}}$ and $\tau^{\Pi_k^{g+1}}$ of offspring in $\mathcal{P}^{g+1}$;
12: Keep top half the fittest individuals in $\mathcal{P}^{g+1}$ based on $\varphi^{\Pi_k^{g+1}}$;
13: Let $\{\Pi_j^{opt}\}_{j=1}^{K}$ be a set of best solutions over all the generations for the K tasks;

sampled solution is determined based on its corresponding NHM using ALGORITHM 22 (see details in Sect. 4.8.4). We then produce the next population $\mathcal{P}^{g+1}$ by combining the current population $\mathcal{P}^g$ and the offspring population $\mathcal{P}_a^g$. Consequently, we update $r_j^{\Pi_k^{g+1}}$, $\varphi^{\Pi_k^{g+1}}$ and $\tau^{\Pi_k^{g+1}}$ of the combined population $\mathcal{P}^{g+1}$, and only half of the fittest combined popula-

tion $\mathcal{P}^{g+1}$ are kept based on $\varphi^{\Pi_k^{g+1}}$. In each generation, we keep track of the fittest Π_j^{opt} for each task T_j. When the maximal generation g_{max} is met, the algorithm returns a set of best $\{\Pi_j^{opt}\}_{j=1}^{K}$ over all the generations for the K tasks.

4.8.2 NHMs Learning and Sampling Solutions

Considering K composition tasks in PMFEA-EDA, we learn $2K-1$ NHMs from promising solutions for sampling new candidate solutions. Among the NHMs, there are K single-tasking NHMs and K-1 multitasking NHMs. With respect to each NHM, a separate solution pool will be maintained by PMFEA-EDA to keep track of useful solutions for building the corresponding NHM. For example, considering the example of the four composition tasks discussed in Sect. 4.7.4, i.e., T_1, T_2, T_3 and T_4, seven pools must be initialized for the four composition tasks and three adjacent task pairs (i.e., T_1 and T_2, T_2 and T_3, and T_3 and T_4).

Moreover, a parameter rsp is used to determine whether multitasking or single-tasking NHMs are selected for sampling. Particularly, a value of rsp close to 0 implies that single-tasking NHMs are more frequently used to build new solutions, while a value close to 1 implies that multitasking NHMs are used with high probability to build new solutions for two adjacent tasks.

The outline of multiple NHMs learning and sampling over K tasks is summarized in ALGORITHM 21. We first initialize a set of empty solution pools $\mathcal{A}_q$, where $1 \leq q \leq (2K-1)$. Afterwards, we assign these encoded solutions to these pools based on the solutions' skill factors $\tau^{\Pi_k^{\star g}}$. For example, if $\tau^{\Pi_k^{\star g}}$ = 1, this solution $\Pi_k^{\star g}$ is assigned to two pools, one for task T_1, and the other is for both task T_1 and T_2. Afterwards, we learn $2K-1$ NHMs from the $2K-1$ pools respectively (see details in Sect. 4.8.3). The iteration part comprises of Step 4 to 11. This iteration will not stop until m new solutions are constructed to form the offspring population $\mathcal{P}_a^{g+1}$. Dur-

ALGORITHM 21. Multiple NHMs learning and sampling over K tasks.

Input : $\mathcal{P}^g$
Output: $\mathcal{P}_a^{g+1}$

1: Initialize a set of empty $\mathcal{A}_q$ for each task and every two adjacent tasks;
2: Assign each solution $\Pi_k^{\star g}$ in $\mathcal{P}^g$ to $\mathcal{A}_q$ based on its skill factor $\varphi^{\Pi_k^{\star g}}$;
3: Learn $2K-1$ NHMs $\mathcal{NHM}_q^g$ from the $2K-1$ $\mathcal{A}_q$;
4: **while** $|\mathcal{P}^{g+1}| \leq m$ **do**
5: $\quad rand \leftarrow Rand(0,1)$;
6: $\quad$ **if** $rand < rsp$ **then**
7: $\quad\quad$ Select one NHM from multitasking NHMs randomly;
8: $\quad$ **else**
9: $\quad\quad$ Select one NHM from single-tasking NHMs randomly;
10: $\quad$ Sample one solution Π_k^{g+1} from the selected NHM and put the solution into $\mathcal{P}^{g+1}$;
11: $\quad$ Π_k^{g+1} inherts the skill factor based on the selected NHM;
12: Return offspring population $\mathcal{P}_a^{g+1}$;

ing the iteration, rsp is used to determine whether one NHM is randomly selected from the $2K-1$ single-tasking NHMs or multitasking NHMs. The selected NHM is used to build one solution. Hence, the skill factor of the newly created solution will also be determined by the associated tasks with the chosen NHM, inspired by the principal of vertical culture transmission [71]. After all iterations have been completed, ALGORITHM 21 returns the newly produced population $\mathcal{P}_a^{g+1}$ for Step 6 of ALGORITHM 20.

4.8.3 NHBSA for Multitasking Evolutionary Search

We employ the node histogram-based sampling [168] as a tool to create new permutations from the selected NHMs in Step 7 and 9 in ALGO-

RITHM 21.

An NHM learned from solutions in each pool $\mathcal{A}_q$ at generation g, denoted by $\mathcal{NHM}_q^g$, is an $n \times n$-matrix with entries $e_{i,r}^g$ as follows:

$$e_{i,r}^g = \sum_{k=0}^{m-1} \delta_{i,r}(\Pi_k^{\star g}) + \varepsilon \tag{4.11}$$

$$\delta_{i,r}(\Pi_k^{\star g}) = \begin{cases} 1 & \text{if } \pi_i = \mathsf{r} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where $i, r = 0, 1, \ldots, n-1$, $\varepsilon = \frac{m}{n-1} b_{ratio}$ is a predetermined bias, and $n = |\mathcal{SR}|$. Roughly speaking, entry $e_{i,r}^g$ counts how often the service index π_i appears in position r of the permutation over all solutions in pool $\mathcal{A}_q$.

Once we have computed $\mathcal{NHM}_q^g$, we use NHBSA to sample new candidate solutions Π_k^{g+1} for the population $\mathcal{P}_a^{g+1}$, see technical details in ALGORITHM 12 in Sect. 3.5.3.

4.8.4 Skill Factor Transmission

Vertical transmission [71] allows every new solution to be evaluated only on the selected tasks that it is most likely to perform well on. This design significantly reduces the total number of fitness evaluations required. To continue to support this design in PMFEA-EDA, we propose a simple skill factor transmission method shown in ALGORITHM 22.

In ALGORITHM 22, every newly created solution assumes the skill factor according to the corresponding NHM. In particular, when a multitasking NHM is selected to create a new solution, this solution will randomly assume one of the two related tasks associated with the NHM as its skill factor. On the other hand, if a solution is created based on a single-tasking NHM, the skill factor of this solution is determined by the task associated to the NHM.

ALGORITHM 22. Skill factor transmission based on the NHM.

if Π_k^g *is created by any multitasking NHM* **then**
  Let T_a and T_b be two selected adjacent tasks of a multitasking NHM;
  $rand \leftarrow Rand(0, 1)$;
  if $rand < 0.5$ **then**
    Π_k^g inherts the skill factor from task T_a;
    Π_k^g is only evaluated on T_a;
  else
    Π_k^g inherts the skill factor from task T_b;
    Π_k^g is only evaluated on T_{a+1};
else
  Let T_c be one selected task associated with a single-tasking NHM;
  Π_k^g inherts the skill factor from task T_c;
  Π_k^g is only evaluated on T_b;

4.9 Experimental Evaluation

The evaluations of our PMFEAs consists of two parts. In the first part, we conduct experiments to evaluate the effectiveness and efficiency of PMFEA, and its two variations, PMFEA-NT and PMFEA-AT based on a quantitative evaluations in Sect. 4.9.1. These three approaches are compared to one state-of-art single-tasking single-objective EC-based method, i.e., Fixed Length Genetic Algorithm (FL) [47], which is reported as a very effective method to find high-quality solutions. In the second part, we study the performance of PMFEA-EDA. Particularly, we compare three multitasking methods: PMFEA-EDA, PMFEA-WTO, PMFEA, and two single-tasking methods (i.e., EDA-NHM proposed in Chapter 3 and FL [47]) in Sect. 4.9.2. We keep using the new benchmark, WSC09, proposed in Chap-

ter 3. This benchmark is extended with four pre-defined QoSM preferences with respect to four user segments: $(0, 0.25]$, $(0.25, 0.5]$, $(0.5, 0.75]$, and $(0.75, 1]$. Therefore, each multitasking method is utilized to find solutions for the four composition tasks concurrently. Meanwhile, each single-tasking method is utilized to find a solution for each task at a time, and its execution time is the aggregation of the total time spent on all tasks. We perform 30 independent runs of each algorithm for all the datasets.

The maximum generation g_{max} is 200. The size of the population m is set to 30, which strictly follow the population size reported in FL [47]. For PMFEA, PMFEA-NT, and PMFEA-AT, the assortative mating $rand$ is set to 0.3, following the popular evolutionary multitasking setting in [225]. For PMFEA-EDA, we define rsp as 0.2 so that every single-tasking NHM and every multitasking NHM are expected to create 6 and 2 solutions respectively for the population size of 30 (i.e., 6×4 single-tasking NHMs $+2 \times 3$ multitasking NHMs$= 30$). For any use of EDA, b_{ratio} is set to 0.0002, following the setting suggested in Sect. 3.7. For other settings of our EDA-NHM and the weights in the fitness function Eq. (4.8), we also follow the suggestions in Sect. 3.7. Similar to the weights settings in Eq. (4.8), the weights in Eq. (4.9) are set to balance all quality criteria in QoSM, i.e., w_7 and w_8 are set to 0.5. We have conducted tests with other weights and parameters, and observe similar results to those reported in Sect. 4.9.1 and Sect. 4.9.2.

4.9.1 Comparing PMFEAs with FL

Comparison of the Fitness

We use an independent sample T-test with a significance level of 5% to verify the observed differences in mean fitness over 30 runs. In particular, a pairwise comparison of different methods have been carried to rank the performances of all the approaches based on the number of times they have been found to be better, similar, or worse than other methods. We

highlight the best performances and the worst performances in green and red colors, respectively, for related values in the tables. Note that when multiple values in a row are highlighted in green that implies that no significant differences among all the approaches can be detected.

Table 4.9: Mean fitness values for our approach in comparison to FL [47] (Note: the higher the fitness the better).

Task 1				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [47]
WSC09-1	0.192631 ± 0.00475	0.19291 ± 0.003977	0.192864 ± 0.003543	0.193947 ± 0.003276
WSC09-2	0.146518 ± 0.003685	0.146975 ± 0.005269	0.148709 ± 0.00488	0.146254 ± 0.002946
WSC09-3	0.152277 ± 0.003385	0.152809 ± 0.003959	0.150053 ± 0.003885	0.15355 ± 0.002688
WSC09-4	0.141319 ± 0.000747	0.140892 ± 0.000836	0.140255 ± 0.00073	0.141451 ± 0.000515
WSC09-5	0.144593 ± 0.001096	0.144193 ± 0.00102	0.143447 ± 0.000946	0.144942 ± 0.000705
Task 2				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [47]
WSC09-1	0.810555 ± 0.00638	0.808541 ± 0.006982	0.809369 ± 0.007696	0.807483 ± 0.005398
WSC09-2	0.748537 ± 0.006183	0.749583 ± 0.00816	0.752848 ± 0.006956	0.74895 ± 0.006425
WSC09-3	0.765014 ± 0.007071	0.764333 ± 0.007089	0.760746 ± 0.006192	0.761924 ± 0.006194
WSC09-4	0.739807 ± 0.000696	0.73975 ± 0.000825	0.739866 ± 0.000853	0.739826 ± 0.000692
WSC09-5	0.73927 ± 0.00081	0.739328 ± 0.00073	0.73936 ± 0.001217	0.739467 ± 0.000735
Task 3				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [47]
WSC09-1	0.820082 ± 0.00571	0.820097 ± 0.004829	0.820107 ± 0.007241	0.819418 ± 0.003768
WSC09-2	0.230114 ± 0.006103	0.231639 ± 0.007691	0.234557 ± 0.00651	0.22968 ± 0.004616
WSC09-3	0.788258 ± 0.003952	0.788829 ± 0.00307	0.789012 ± 0.002968	0.788726 ± 0.002576
WSC09-4	0.224035 ± 0.001628	0.224026 ± 0.001827	0.224278 ± 0.001957	0.224127 ± 0.001467
WSC09-5	0.221114 ± 0.001512	0.221319 ± 0.001286	0.221169 ± 0.002244	0.221102 ± 0.001248
Task 4				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [47]
WSC09-1	0.222976 ± 0.007486	0.223659 ± 0.008279	0.219863 ± 0.013342	0.221582 ± 0.00946
WSC09-2	0.105114 ± 0.006103	0.10656 ± 0.007747	0.109708 ± 0.00659	0.10468 ± 0.004616
WSC09-3	0.215947 ± 0.00718	0.215877 ± 0.007496	0.217783 ± 0.005575	0.216698 ± 0.00533
WSC09-4	0.099035 ± 0.001628	0.098941 ± 0.001889	0.099276 ± 0.001935	0.099127 ± 0.001467
WSC09-5	0.096114 ± 0.001512	0.096312 ± 0.001287	0.096085 ± 0.002181	0.096102 ± 0.001248

First, all the multitasking approaches, i.e., PMFEA, PMFEA-NT, and PMFEA-AT, outperform FL [47] since the most values related to FL [47] in Tables 4.9 are marked in red color. This observation shows that multitask-

ing is more competent at improving the quality of solutions by utilizing the knowledge of other tasks through assortative mating. Such an observation further agrees with the findings in MFEA [71]

Second, the quality of solutions produced by PMFEA-NT is consistently high since all the solutions are marked as the best performance except one that is marked with an average performance in Table 4.9. This corresponds well with our expectation that the careful selections of the neighbourhood structure in PMFEA-NT will lead to effective search of good solutions. This indicates that useful knowledge of solutions (i.e., the order of services used for composition) on one task can be effectively transferred to the neighbouring tasks, and the skill factor of these solutions have a chance to be updated with respect to the neighbouring tasks.

Third, PMEFA and PMFEA-AT are comparable to each other, and both are less favourable to PMFEA-NT. We can see that the performances of both algorithms vary on different tasks, i.e., 16 out of 20 tasks as the best performance and 4 out of 20 tasks are marked as the worst performance in Table 4.9. MEFA strictly follows the vertical cultural transmission, so it loses the chance to transfer knowledge to other tasks. On the other hand, although PMFEA-AT assigns candidate solutions to all tasks for evaluations, it does not outperform PMFEA. It may be due to that candidate solutions only inherit the most effective task of all the tasks and make a locally optimal choice each time. Such a greedy strategy can easily lead to local optima, resulting in no improvement in terms of the effectiveness.

Comparison of the Execution Time

Table 4.10 shows the execution times observed for PMFEA-NT, PMFEA-NT, MFEA and FL [47] on the four tasks as a whole. Again, an independent sample T-test has been conducted over 30 runs of these algorithms.

PMFEA, PMFEA-NT, and PMFEA-AT require significantly less execution time while FL [47] consistently takes four times the execution time of PMFEA, PMFEA-NT and PMFEA-AT approximately in the worst cases

Table 4.10: Mean execution time (in s) for our approaches in comparison to FL (Note: the shorter the time the better).

All tasks (Task 1, Task 2, Task 3 and Task 4)				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [47]
WSC09-1	54 ± 52	44 ± 32	79 ± 87	150 ± 151
WSC09-2	1900 ± 1032	1925 ± 702	2371 ± 804	8479 ± 3002
WSC09-3	1479 ± 1257	1542 ± 1159	1821 ± 740	5926 ± 3199
WSC09-4	64311 ± 16843	60925 ± 16311	71903 ± 19042	250146 ± 55355
WSC09-5	12943 ± 6615	12456 ± 6094	13689 ± 6723	47879 ± 16126

(e.g., the mean execution time for WSC09-3). It is due to that FL [47] is a single-tasking EC technique that optimizes four tasks separately.

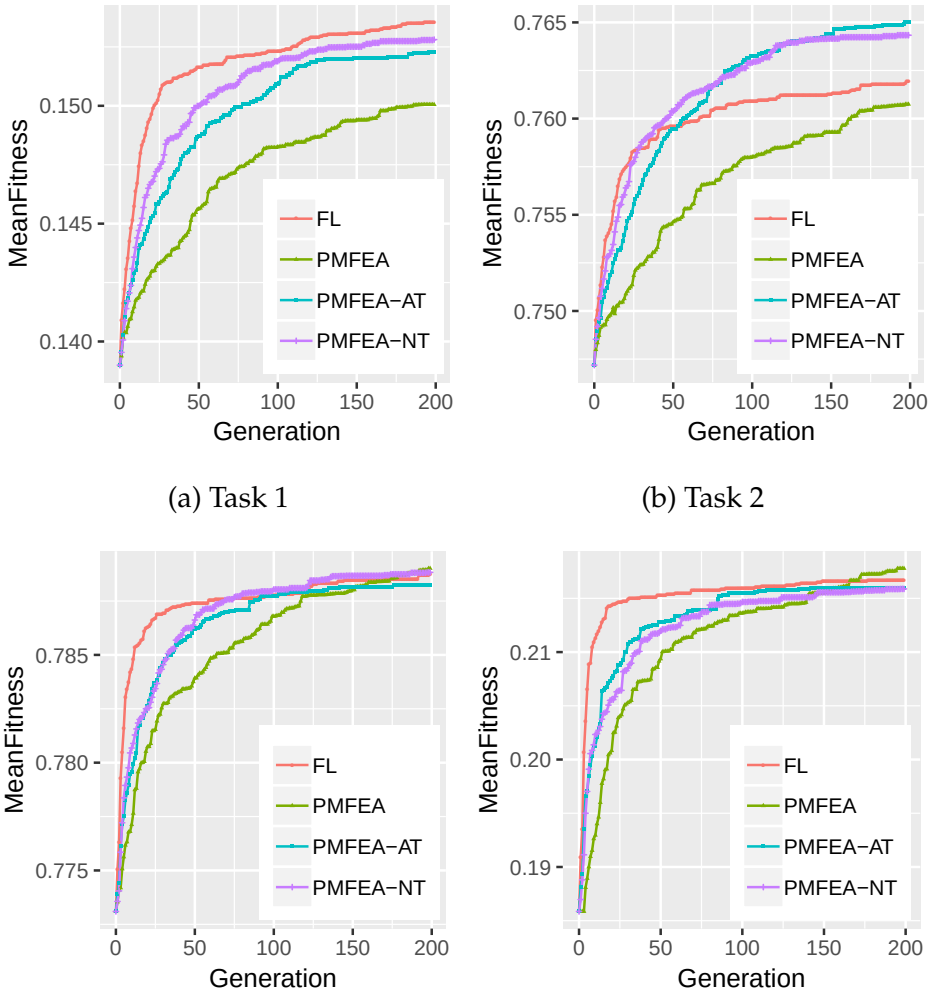
PMFEA-NT achieves the shortest execution time for each dataset consistently. Meanwhile, PMFEA and PMFEA-AT are very comparable to each other. These observations correspond well with our expectation that the execution time of PMFEA can be reduced due to the effective knowledge transformation across different tasks through the introduced neighborhood structure over multiple tasks.

Comparison of the Convergence Curve

We investigate the convergence curves of PMFEA-AT, PMFEA-NT, PMFEA, and FL [47] on the four tasks over 30 runs, and use WSC09-3 as an example to illustrate the performance of all the compared methods.

Fig. 4.11 shows the evolution of the mean fitness value of the best solutions found along 200 generations for all the approaches. Among all the four tasks, we observe a significant increase in the fitness values towards the optimum for all the approaches, which eventually reach a plateau with more stable improvements. In particular, PMFEA consistently converges slower than all the other approaches. In contrast, FL [47] consistently converge very fast than all the other approaches. In addition, PMFEA-NT

Fig. 4.11: Mean fitness over generations for tasks 1-4, for WSC09-3 (Note: the larger the fitness the better).



(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4

and PMFEA-AT converge faster than PMFEA and are comparable to each other. This observation indicates that evaluating an offspring on neighbour tasks, or all tasks can contribute to more effective assortative mating, accelerating the convergency speed.

4.9.2 Comparing PMFEA-EDA with PMFEA-EDA-WTO, PMFEA, EDA-NHM, and FL

Comparison of the Fitness

An independent sample T-test is employed at a significance level of 5% to verify the observed differences in mean fitness of PMFEA-EDA, PMFEA-WTO, PMFEA, EDA-NHM and FL, over 30 runs. Particularly, pairwise comparisons of all the competing methods are carried out to count the number of times they were found to be better, similar, or worse than the others. Consequently, we can rank all the competing methods and highlight the top performance in a green color.

Table 4.11 shows the mean value of the solution fitness and the standard deviation over 30 runs for each task. We observe that the comprehensive quality of the solutions produced by using PMFEA-EDA, and EDA-NHM are generally higher than those obtained by PMFEA and FL [47]. This corresponds well with our expectation that learning the knowledge of promising solutions explicitly can increase the chance of finding high-quality composite services.

Furthermore, PMFEA-EDA performs better than single-tasking EDA-NHM. This observation indicates that addressing multiple tasks collectively is often more effective than addressing each task individually. Particularly, compared to single-tasking EDA, multitasking methods are more likely to evolve a well diversified population of solutions because solutions with different skill factors are kept in such a population. Consequently, we can easily prevent the evolutionary process from converging prematurely.

Lastly, PMFEA-EDA also significantly outperforms PMFEA-EDA-WTO and achieves consistently top performance among all the tasks. This corresponds well with our expectation that knowledge sharing through multitasking NHMs can significantly increase the chance of finding high-quality solutions.

Table 4.11: Mean fitness values of solutions per task for our approaches in comparison to PMFEA, EDA-NHM and FL [47] (Note: the higher the fitness the better).

Task T_1					
Method	PMFEA-EDA	PMFEA-EDA-WTO	PMFEA [185]	EDA [183]	FL [47]
WSC09-1	0.196195 ± 0.000547	0.193173 ± 0.002266	0.192864 ± 0.003543	0.196316 ± 0.000314	0.193947 ± 0.003276
WSC09-2	0.15621 ± 0.000806	0.141811 ± 0.000646	0.148709 ± 0.00488	0.145844 ± 0.001347	0.146254 ± 0.002946
WSC09-3	0.158664 ± 0.001038	0.147505 ± 0.001923	0.150053 ± 0.003885	0.156733 ± 0.001425	0.15355 ± 0.002688
WSC09-4	0.142097 ± 0.000467	0.139684 ± 0.000359	0.140255 ± 0.00073	0.140256 ± 0.000673	0.141451 ± 0.000515
WSC09-5	0.145391 ± 0.000337	0.143159 ± 0.000389	0.143447 ± 0.000946	0.145045 ± 0.000278	0.144942 ± 0.000705
Task T_2					
Method	PMFEA-EDA	PMFEA-EDA-WTO	PMFEA [185]	EDA [183]	FL [47]
WSC09-1	0.815627 ± 0.002673	0.808235 ± 0.003689	0.809369 ± 0.007696	0.816144 ± 0	0.807483 ± 0.005398
WSC09-2	0.761226 ± 0.00064	0.74019 ± 0.001354	0.752848 ± 0.006956	0.74704 ± 0.005348	0.74895 ± 0.006425
WSC09-3	0.77392 ± 0.003505	0.755821 ± 0.003864	0.760746 ± 0.006192	0.772646 ± 0.001529	0.761924 ± 0.006194
WSC09-4	0.741242 ± 0.000261	0.738214 ± 0.000567	0.739866 ± 0.000853	0.739946 ± 0.000233	0.739826 ± 0.000692
WSC09-5	0.74173 ± 0.000756	0.738334 ± 0.000293	0.73936 ± 0.001217	0.739431 ± 0.000255	0.739467 ± 0.000735
Task T_3					
Method	PMFEA-EDA	PMFEA-EDA-WTO	PMFEA [185]	EDA [183]	FL [47]
WSC09-1	0.806034 ± 0.097563	0.820217 ± 0.003495	0.820107 ± 0.007241	0.823483 ± 0.000978	0.819418 ± 0.003768
WSC09-2	0.242136 ± 0.000241	0.222519 ± 0.001753	0.234557 ± 0.00651	0.232177 ± 0.00283	0.22968 ± 0.004616
WSC09-3	0.791989 ± 0	0.787464 ± 0.002324	0.789012 ± 0.002968	0.791938 ± 0.000146	0.788726 ± 0.002576
WSC09-4	0.227768 ± 0.000446	0.221226 ± 0.000789	0.224278 ± 0.001957	0.224629 ± 0.00063	0.224127 ± 0.001467
WSC09-5	0.224546 ± 0.000719	0.219729 ± 0.000897	0.221169 ± 0.002244	0.2218 ± 0.000243	0.221102 ± 0.001248
Task T_4					
Method	PMFEA-EDA	PMFEA-EDA-WTO	PMFEA [185]	EDA [183]	FL [47]
WSC09-1	0.226227 ± 0.011127	0.22145 ± 0.009191	0.219863 ± 0.013342	0.22731 ± 0.003251	0.221582 ± 0.00946
WSC09-2	0.117137 ± 0.000241	0.097486 ± 0.001454	0.109708 ± 0.00659	0.107177 ± 0.00283	0.10468 ± 0.004616
WSC09-3	0.222379 ± 0	0.215091 ± 0.005245	0.217783 ± 0.005575	0.222212 ± 0.00034	0.216698 ± 0.00533
WSC09-4	0.102637 ± 0.000634	0.096245 ± 0.001065	0.099276 ± 0.001935	0.099674 ± 0.000596	0.099127 ± 0.001467
WSC09-5	0.099487 ± 0.000716	0.094344 ± 0.000876	0.096085 ± 0.002181	0.096785 ± 0.000271	0.096102 ± 0.001248

Comparison of the Execution Time

An independent sample T-test at a significance level of 5% is also employed to verify the observed differences in mean execution time (in seconds) over 30 runs.

Table 4.12: Mean execution time (in s) over all the tasks for our approaches in comparison to PMFEA [185], EDA-NHM and FL [47] (Note: the shorter the time the better).

Tasks T_1, T_2, T_3 and T_4					
Method	PMFEA-EDA	PMFEA-EDA-WTO	PMFEA [185]	EDA-NHM	FL [47]
WSC09-1	54 ± 8	52 ± 11	79 ± 87	184 ± 12	150 ± 151
WSC09-2	1571 ± 181	1533 ± 218	2371 ± 804	7058 ± 369	8479 ± 3002
WSC09-3	1085 ± 186	975 ± 122	1821 ± 740	5057 ± 885	5926 ± 3199
WSC09-4	57788 ± 6902	50310 ± 7535	71903 ± 19042	202464 ± 9366	250146 ± 55355
WSC09-5	9671 ± 1092	8834 ± 819	13689 ± 6723	39257 ± 1885	47879 ± 16126

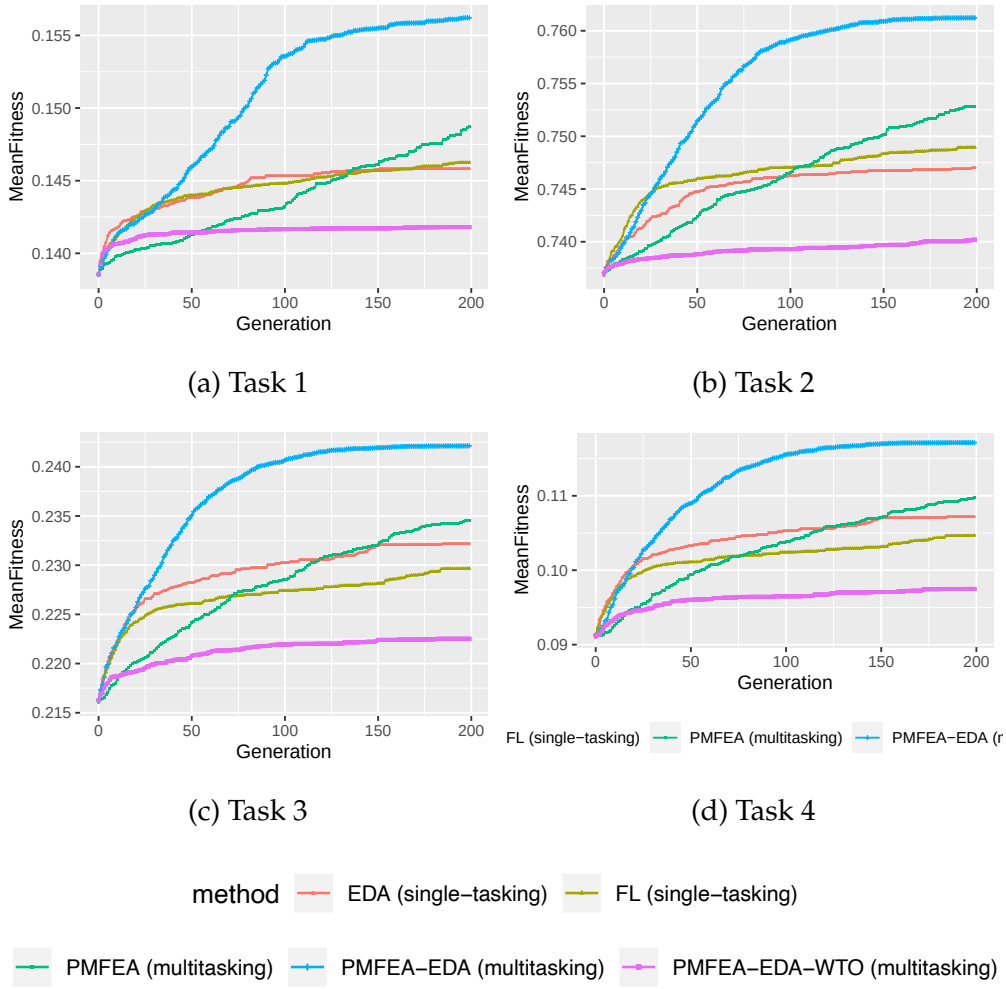
Table 4.12 shows the mean value of the execution time and the standard deviation over 30 repetitions for all the tasks. PMFEA-EDA, PMFEA-EDA-WTO, and PMFEA appear to be very efficient than EDA-NHM and FL [47]. This is because EDA-NHM and FL [47] are single-tasking methods that have to solve each composition task one at a time.

Moreover, compared to PMFEA-EDA, PMFEA-EDA-WTO requires slightly less execution time for WSC09-03 to WSC09-05. This is because PMFEA-EDA demands more time in learning NHMs when the service repository $\mathcal{SR}$ becomes larger. However, the extra time incurred in PMFEA-EDA is not substantial compared to PMFEA-EDA-WTO.

Comparison of the Convergence Curve

We also studied the convergence curves of PMFEA-EDA, PMFEA-EDA-WTO, PMFEA, EDA-NHM, and FL [47]. In Fig. 4.12, we show the behaviours of effectiveness of all the methods using WSC09-2 as an example.

Fig. 4.12: Mean fitness over generations for tasks 1-4, for WSC09-2 (Note: the larger the fitness the better).



(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4

Fig. 4.12 shows the evolution of the mean fitness value of the best solutions found along 200 generations for all the approaches. We can see that PMFEA-EDA converges much faster than other methods in most tasks (Task 1, 2 and 3). Besides that, PMFEA-EDA converges much faster than PMFEA-EDA-WTO, and eventually reaches the highest plateau. This observation matches well with our expectation that the explicit knowledge

sharing across tasks is very effective.

4.10 Summary

The overall goal of this chapter is to propose effective EC-based methods to solve two categories of MOCQP: WSC-MO and WSC-MQP. We have achieved the first goal of solving WSC-MO by completing the following: (1) We model the WSC-MO problem with two objectives, which are related to the functional and non-functional aspects of composite services, i.e., QoSM and QoS. (2) One memetic multi-objective fully automated web service composition approach, namely, MNSGA2-EDA, was proposed. Particularly, NSGA-II algorithm is combined with a novel EDA-based local search, which is performed separately and concurrently in different regions of the Pareto front via multiple NHMs. These NHMs are learned based on suitable Pareto front solutions selected by a clustering technique and other good candidate solutions evolved by NSGA-II. (3) Using the challenging benchmark datasets proposed in Chapter 3, MNSGA2-EDA was compared to the-state-of-the-art methods, such as Hybrid [40], Hybrid-L [40], and a baseline algorithm, NGSA-II, in terms of both effectiveness and efficiency. Our experimental results show that MNSGA2-EDA can efficiently produce composite services with much better Pareto optimal solutions than Hybrid, Hybrid-L and NGSA-II.

The development of MNSGA2-EDA and the experimental study leads to several major findings for the WSC-MO problem: (1) The clustering technique in MNSGA2-EDA can select less, but more suitable candidate solutions for local search, compared to the randomly selected subproblem representatives in the Hybrid and Hybrid-L. (2) Our EDA-based local search can make substantial performance improvement for NSGA-II, compared to the single-objective local search operator introduced in Hybrid-L.

We have also achieved the second goal of solving WSC-MQP by completing the following: (1) We formulate the WSC-MQP problem as a multi-

tasking service composition problem, which collectively handles multiple requests for multiple user segments. These requests have the same functionality (i.e., inputs and outputs), but have distinctive quality preferences on QoSM. (2) Two multi-factorial evolutionary algorithms, namely, PMFEA and PMFEA-EDA are proposed in this chapter. The first algorithm, PMFEA, is proposed based on MFEA with a permutation-based representation and corresponding genetic operators in the assortative mating, allowing implicit knowledge of promising solutions to be learned and shared. In addition, we explored two variations of PMFEA, i.e., PMFEA-NT and PMFEA-AT, which permit evaluations on the neighbourhood tasks and all the task, respectively. The experimental results show that all the PMFEAs performed significantly better than one single-tasking single-objective EC-based method, FL [47], using only a fraction of time. Meanwhile, PMFEA-NT achieves the best effectiveness and efficiency among all the competing method by performing additional evaluations on the neighbouring tasks. (4) The second algorithm, PMFEA-EDA, was proposed with a different permutation-based representation. Meanwhile, a set of single-tasking and multitasking NHMs are constructed to allow explicit knowledge of promising solutions to be learned and shared. Furthermore, a sampling mechanism is proposed to effectively sample new promising candidate solutions via a set of NHMs for multiple tasks. (5) PMFEA-EDA was compared with PMFEA-EDA-WT to examine its effectiveness of the knowledge sharing of solutions across different composition requests, and with PMFEA and the state-of-the-art single-tasking single-objective algorithm (i.e., FL and NHM-EDA proposed in Chapter 3) to test its performance in terms of effectiveness and efficiency.

The development of PMFEAs and PMFEA-EDA and their experimental study leads to several major findings for the WSC-MQP problem: (1) Our multitasking approaches can produce solutions with much higher quality than the solutions produced by the single-tasking single-objective approaches. This finding indicates that the solutions evolved for one re-

quest can help the solutions to be evolved for the other requests. (2) The proper use of the neighbourhood structure can enhance the effectiveness of MFEAs. Such finding motivates us to propose more effective PMFEA-EDA that allows knowledge to be explicitly shared among neighbouring tasks. (3) The explicit learning and sharing mechanism in PMFEA-EDA is more effective than the implicit knowledge learning and sharing in PMFEAs. Particularly, we found that PMFEA-EDA can consistently achieve outstanding performance in both effectiveness and efficiency, compared to other PMFEAs and single-tasking EC approaches (i.e., FL [47] and EDA-NHM).

Chapter 5

Evolving Robust Composite Services for Dynamic Semantic Web Service Composition

5.1 Introduction

The previous chapter studied two categories of MOCQP: WSC-MO and WSC-MQP. WSC-MO aims to find a set of approximated Pareto composite services by simultaneously considering QoSM and QoS. WSC-MQP aims to concurrently find a set of composite services with optimized comprehensive quality for multiple user segments with distinctive QoSM preferences. We also propose novel EMO and MFEA algorithms to effectively and efficiently solve WSC-MO and WSC-MQP, respectively. All the previous studies reported in this thesis share a common assumption that QoS of web services seldom changes or does not change at all. Researchers in this group often follow the average values of the historical QoS to find high-quality composite services. However, the composite services generated by these approaches may not perform well when QoS of their component services changes. In a worst-case scenario, they can become inexecutable due to the failures of their component services.

As discussed in Sect. 1.1, QoS is changing dynamically in the real world, due to various reasons, e.g., software/hardware failures, and network issues [81]. Although QoS criteria (such as response time, throughput, failure probability, availability, price and popularity [48]) advertised by service providers are often used to match the needs of service requesters, it can be very risky because no service provider can guarantee the advertised QoS under all circumstances [89]. In fact, a composite service may suffer from various changes. In Particular, stochastic service failures constitute the most critical uncertainty because component services can fail unexpectedly, causing unforeseeable interruptions to a composite service created at the design stage. For this reason, stochastic service failures are the central focus of this chapter.

To design composite services, we must take potential service failures into account to avoid abandoning a composite service completely. Some existing works [9, 125, 190] propose to use re-optimization techniques at the service execution stage (see the differences between execution stage and design stage in Sect. 2.1.2). Particularly, the frequency of re-optimization is scheduled to cope with changes of QoS, which are assumed to happen periodically (e.g., every a few generations [190] or every period of time [9, 125]). In fact, [6, 108] recommends proactive use of re-optimization techniques in response to anticipated future QoS changes based on sufficient historical data. These re-optimization techniques discussed above can also be used to handle stochastic service failures.

While re-optimization techniques can help to cope with stochastic service failures to some extent, these approaches ignore the importance of building robust composite services at the design stage. Moreover, the assumption of periodical changes in QoS or sufficient historical QoS data poses noticeable feasibility challenges. In reality, periodical re-optimization techniques may not be fast enough to cope with QoS changes, since services often fail sporadically in a highly unpredictable manner. Meanwhile, newly registered services may not have sufficient

historical QoS data for training a reliable prediction model. *To address these limitations, we will introduce a* **R**obust **W**eb **S**ervice **C**omposition *problem for handling stochastic* ***S****ervice* ***F****ailures (henceforth referred to as* ***RWSC-SF****) via two stages, namely, the design stage and the execution stage in this chapter*. In the design stage, RWSC-SF constructs baseline composite services by explicitly considering stochastic service failures. At the execution stage, the baseline composite services can cope with unexpected service interruptions in a robust manner with an efficient and effective local search to resume the high quality of the composite services.

It is difficult to measure the robustness of any composite service in terms of the expected QoS and QoSM subject to stochastic service failures, due to the difficulty of enumerating all the possible failure scenarios (see definition of scenario in the following Sect. 5.3). As discussed in Sect. 2.2.2 fitness approximation methods must be developed to estimate the robustness. Particularly, problem approximation, data-driven functional approximation and fitness inheritance are usually used for the estimation [86, 87].

Problem approximation is preferable, compared to the data-driven functional approximation and fitness inheritance. This is because: (1) adopting data-driven functional approximation that trains explicit models (also called meta-models or surrogates), requires sufficient historical data that maps between the design parameters and the quality of the design [86]. However, we often do not have sufficient historical data for training reliable models, and (2) fitness inheritance estimates the fitness of one individual by the fitness of other similar individuals. However, it is hard to define a similarity or distance measure between any two composite services in our problem. This is because composite services that serve the same functionality can differ in both the component services and workflow structures. Moreover, when the dimension of decision variables increases, distance measures can become less useful.

Monte Carlo sampling is one popular method of problem approximation and it is often employed to approximate the robustness of candidate

solutions effectively via a number of simulated dynamic scenarios. However, it remains an important research question regarding how to use a small number of simulated scenarios on service failure to accurately estimate the robustness of any composite service. Moreover, GA is a popular EC technique that has enabled many researchers to tackle several challenging service composition problems [44, 47]. Therefore, in this chapter, GA will be utilized to construct composite services with optimized robustness at the design stage.

In this chapter, we will first investigate a Monte Carlo sampling-based technique [153] for estimating the robustness of candidate composite services. Subsequently, we will propose a GA-based method based on the Monte Carlo sampling technique (henceforth referred to as ***GA-MC****) and conduct our initial study on a small-scale RWSC-SF problem using a small benchmark (i.e., OWLS-TC [97] that contains a maximum of 946 web services in a service repository).*

When dealing with large-scale RWSC-SF problem over a large service repository (e.g., the maximum size of a service repository is 15211 using WSC09 [92] benchmark), the complexity of the fitness landscape will increase dramatically [55]. It may lead to the deterioration of the performance in robustness estimation. For example, this complexity makes it hard to accurately estimate the robustness of composite services through the Monte Carlo sampling-based technique. This is because Monte Carlo sampling requires a large number of sampling size to ensure the estimated robustness is sufficiently accurate.

To cope with the robustness measure for the large-scale RWSC-SF problem, we will propose an alternative robustness approximation method. We expect this method can work efficiently and accurately when the size of the service repository increases. In addition, evolutionary control is used to decide whether an actual or approximation method should be utilized for robustness evaluations [87]. As discussed in Sect. 2.2.2, generation-based evolutionary control can achieve good performance with fewer control interventions on generations, compared to the individual-based one. Mean-

while, an adaptive frequency over the generations should be considered because the fidelity of the approximate model of the evaluations may vary significantly during the optimization [87]. Therefore, *we will propose a two-stage GA algorithm with an adaptive generation-based evolutionary control over two consecutive evolutionary stages for finding composite services with high robustness. Particularly, the two evolutionary stage employ different methods for the robustness estimation. We will conduct our study on a large-scale RWSC-SF problem using large service composition benchmarks, i.e., WSC-08 and WSC-09*. The following objectives are sought in this chapter:

1. We introduce a new dynamic service composition problem for handling stochastic service failures. This problem is formulated as a novel two-stage service composition, consisting of a design stage and a execution stage. Particularly, the robustness of composite services in terms of expected QoS and QoSM is explicitly considered and optimized at the design stage. These composite services can be repaired easily with the help of an efficient local search technique so that their execution can be resumed at the execution time with negligible impact on QoS and QoSM.

2. We propose a GA-MC method to solve the small-scale RWSC-SF problem. Particularly, we introduce two key techniques that jointly form an effective fitness function for searching robust composite services. The first technique is to adopt the Monte Carlo sampling technique [153] to accurately approximate the robustness in terms of the expected QoS and QoSM of any given composite services. The second technique is an application of re-optimization technique (i.e., local search) that effectively repairs composite services in response to arbitrary service failures.

3. To conduct experiments to explore the performance of GA-MC on a small-scale RWSC-SF problem, GA-MC is compared to a Fixed

Length GA [47] (henceforth referred to as FL), which achieves outstanding performance in finding high-quality solutions. Our experimental results show that GA-MC can produce baseline composite services with significantly higher robustness, compared to FL.

4. To improve the efficiency and accuracy of Monte Carlo sampling-based method for measuring the robustness, we introduce a new robustness estimation method based on a lower bound of the expected QoS and QoSM. Notably, this method aims to reduce the variance of the estimated robustness by carefully selecting scenarios based on service repositories. Moreover, to leverage the importance of each selected scenario, the probability of the selected scenario to be sampled is considered as the weight.

5. To further reduce the computation time of GA and maintain its effectiveness, a two-stage GA, denoted GA-2Stage, is proposed with an adaptive evolution control mechanism. Particularly, in stage one, we do not consider service failures, and an efficient evaluation (i.e., the comprehensive quality) is employed to find good solutions that are likely to present high robustness for the preparation of stage two. In stage two, these solutions can be further evolved to improve their robustness using our proposed lower bound robustness estimation method. Moreover, to balance the computation resources assigned to the two stages, an adaptive archive-based evolutionary control is proposed. Particularly, it determines at which generation stage two should start. Meanwhile, the first generation of stage two is initialized with the archive, which stores good solutions found in stage one.

6. To demonstrate the effectiveness and efficiency of our GA-2Stage algorithm for the large-scale RWSC-SF problem, we conduct experiments to compare GA-2Stage against three GA-based approaches: one method only employs GA with the lower bound robustness esti-

mation throughout all the generations (henceforth referred to as GA-RE), the GA-MC algorithm and FL [47]. Our experiment results show that GA-2Stage can produce highly robust composite services consistently in the case of stochastic service failures, regardless of the size of the service repository.

5.2 Chapter Organization

The remainder of this chapter is organized as follows. Sect. 5.3 introduces the RWSC-SF problem. Sect. 5.4 and 5.5 present our new methods, GA-MC and GA-2Stage, to solve the RWSC-SF problem. Sect. 5.6 outlines the experimental design and results for evaluating the performances of GA-MC and GA-2Stage. Sect. 5.7 provides a summary of this chapter.

5.3 The RWSC-SF Problem

In this section, we extend the definition of a *semantic web service* (i.e., a tuple $S = (I_S, O_S, QoS_S)$, introduced in Sect. 3.3) by considering *service failure probability*, pr_S. In particular, I_S is a set of service inputs that are consumed by S, O_S is a set of service outputs that are produced by S, and $QoS_S = \{t_S, c_S, r_S, a_S, pr_S\}$ refer to the response time, cost, reliability, availability, *service failure probability* of S.

In practice, the execution of a composite service is usually confronted with stochastic service failures [48]. A *service failure probability* pr_S can be approximated by dividing the number of failed invocations by the total number of invocations conducted in the past on service S [232]. Also, pr_S of newly published web services can be estimated as the pr_S of web services hosted by the same service providers in the same location. Moreover, for any service in the service repository, its failure probability is assumed to be independent of each other.

Herein we model the RWSC-SF problem as a two-stage web service composition problem, consisting of the design stage and execution stage. At the design stage, it aims to construct baseline composite services (i.e., composite services to be executed) with optimized robustness by explicitly considering stochastic service failures. The baseline solution is further executed at the execution stage. Such a solution is expected to continue working reliably or be easily repaired during its execution.

To explicitly consider robustness at the design stage, problem approximation is utilized to estimate the robustness via a set of simulated scenarios with respect to service failures. It has been employed in many other problems for measuring the robustness via the use of a continuous or discrete scenarios set [105]. We define the *robustness* of a composite service in the presence of stochastic service failures captured by a set of scenarios $\mathbb{Q}$. A *scenario* $Q \in \mathbb{Q}$ corresponds to a set of services $\{S_j\}$ that remain accessible during the execution of a composite service, where $\sum_{Q\in\mathbb{Q}} Pr(Q) = 1$. Let $\mathscr{L}(\Pi, Q)$ be a local search operator (i.e., an efficient re-optimization technique) that produces a new feasible composite service Π' for Q through applying local changes to Π. The robustness is defined as the expected quality of a composite service across all possible scenarios as follows:

$$r(\Pi) = \sum_{Q\in\mathbb{Q}} f_{cq}(\mathscr{L}(\Pi, Q))Pr(Q) \tag{5.1}$$

$$f_{cq}(\Pi') = \begin{cases} w_1\hat{MT} + w_2\hat{SIM} + w_3\hat{A} + w_4\hat{R} + \\ \quad w_5(1-\hat{T}) + w_6(1-\hat{CT}) & \text{if } \Pi' \Rightarrow \mathcal{G} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where $\Pi' \Rightarrow \mathcal{G}$ denotes that a permutation Π' that corresponds to $\mathscr{L}(\Pi, Q)$ can be decoded into a functionally valid $\mathcal{G}$. Otherwise, no composite service can be decoded from the permutation Π'. In other words, the comprehensive quality, f_{cq}, of Π' equals 0.

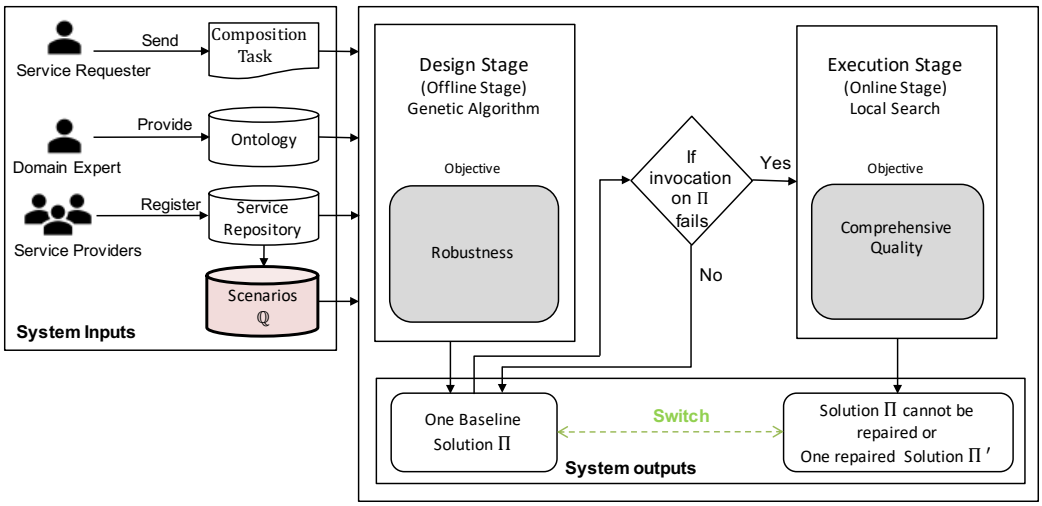


Fig. 5.1: Two-stage robust web service composition system.

Our two-stage robust web service composition system is illustrated in Fig. 5.1. This composition system requires three inputs: a composition task initialised by the service requester, a service repository provided by the service providers, and an ontology defined by the domain experts. At the design stage, a global searching technique, such as GA, can be utilised to search for a baseline solution Π with high robustness measured by Eq. (5.1). This baseline service Π serves as an output of the robust service composition system. At the execution stage, the baseline solution will be executed if none of its component services fails. Otherwise, this baseline solution will be repaired through a local search technique to resume its feasibility thereafter. This repairing process does not guarantee that the solution Π can always be repaired because a service request might not be satisfied with the remaining available services. Note that the robustness measure in Eq. 5.1 requires a local search for every scenario. The same local search is also suggested to be used at the execution stage.

5.4 GA-MC Algorithm to RWSC-SF

In this section, we first present our new robustness estimation method in Sect. 5.4.1. Subsequently, we present our GA-MC algorithm for solving the RWSC-SF problem in Sect. 5.4.2. Consequently, we demonstrate the details of the robustness estimation in Sect. 5.4.3.

Due to the difficulty of enumerating all the possible failure scenarios for measuring the robustness, Monte Carlo sampling-based technique [153] can be adopted to estimate the robustness. One advantage of this robustness estimation method is that it can perform a cheap but not necessarily accurate fitness estimation in the frame of evolutionary computation. In other words, the estimated robustness should be accurate enough for reliably ranking individuals with respect to the robustness.

GA has been successfully utilized as a global searching technique for effectively searching service composition with optimized QoS and/or QoSM [47]. However, the success of GA must rely on an "accurate enough" fitness function for ranking evolved composite services by GA.

5.4.1 Robustness Estimation

As shown in Eq. (5.1), the robustness is defined as the expected quality of a composite service across all possible scenarios and can be directly estimated through Monte Carlo sampling [153] as follows:

$$\begin{aligned} r(\Pi) &= \sum_{Q \in \mathbb{Q}} f_{cq}(\mathscr{L}(\Pi, Q)) Pr(Q) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} f_{cq}(\mathscr{L}(\Pi, Q_i)) \end{aligned} \tag{5.3}$$

where N is the sample size. Particularly, in Eq. (5.3), Π is evaluated N times based on N sampled scenarios $\{Q_i\}_{i=1}^{N}$.

5.4.2 Outline of GA-MC

ALGORITHM 23. GA-MC method for robust service composition.

Input : composition task T, Ontology $\mathcal{O}$, service repository $\mathcal{SR}$, sample size N, and the number of neighbors n_{nb}

Output: a baseline solution

1: Initialize $\mathcal{P}^0$ with m randomly permutations, each represented as a Π_k^g (where $k = 1, \ldots, m$);
2: **Evaluate each permutation in $\mathcal{P}^0$ against the stochastic service failures based on N simulations in Eq. (5.3)**;
3: Set generation counter $g \leftarrow 0$;
4: **while** $g < g_{max}$ **do**
5: Populate $\mathcal{P}^{g+1}$ with m permutations from $\mathcal{P}^g$ through the use of genetic operators;
6: **Evaluate each permutation in $\mathcal{P}^{g+1}$ against the stochastic service failures based on N simulations in Eq. (5.3)**;
7: Set $g \leftarrow g + 1$;
8: Select the best solution Π^{opt} in $\mathcal{P}^g$ as a baseline;

GA-MC for evolving robust composite services is outlined in ALGORITHM 23. It follows a state-of-the-art service composition approach, i.e., FL [47] except in Step 2 and Step 6. This algorithm takes five inputs: service composition task $T = (I_T, O_T)$, an ontology $\mathcal{O}$ that describes all the parameters of the web services, a service repository $\mathcal{SR}$, sampling size N, and the number of neighbours n_{nb} to be exploited for repairing each solution in a scenario. We begin with initializing population $\mathcal{P}^0$ with m randomly generated permutations Π_k^g (where $k = 1, \ldots, m$). This permutation-based representation is exactly the same as that in Sect. 4.7.2. Particularly, each permutation can be interpreted as a DAG through the use of the same decoding algorithm introduced in Sect. 4.7.2. In step 2, we evaluate the fitness values of each permutation against N randomly sam-

pled scenarios, see details in Sect. 5.4.3. The iterative steps (Step 4 to 7) will be repeated until the maximum number of generations, g_{max}, is reached. During each iteration, m permutations are produced from the same genetic operators, discussed in Sect. 4.4.3, to form the next generation $\mathcal{P}^{g+1}$. This newly created population is then evaluated by following the same process in Step 2. Consequently, the best solution with the highest fitness is returned by GA-MC.

5.4.3 Robustness Estimation based on Monte Carlo Sampling

ALGORITHM 24. Robustness estimation based on Monte Carlo sampling (Step 2 and 6 in ALGORITHM 23).

Input : population $\mathcal{P}^g$, the number of neighbor n_{nb}, sample size N, and service repository $\mathcal{SR}$

Output: evaluated $\mathcal{P}^g$

1: **foreach** Π *in* $\mathcal{P}^g$ **do**
2: Sample N scenarios based on pr_S of each S in $\mathcal{SR}$;
3: **foreach** *scenario* Q *in the* N *sampled scenarios* **do**
4: Produce another permutation Π' that encodes Π based on Q;
5: Generate a size n_{nb} of neighbors from $\Pi^\star$ (i.e., an encoded solution of Π') by local search operator;
6: Identify the best neighbor Π' with the highest fitness measured by Eq. (3.5);
7: Set the fitness of Π as an averaged fitness value of N best neighbours using Eq. (5.3);
8: **return** evaluated $\mathcal{P}^g$;

Our proposed fitness function in Eq. (5.3) approximates the robustness

of every candidate solution subject to N randomly sampled scenarios with fast local search for each scenario. This robustness estimation process is provided in ALGORITHM 24. Particularly, Step 4 and Step 5 play a crucial role. In Step 4, we produce another permutation Π' that encodes Π based on a sampled Q. This produced permutation allows some promising component services that belong to the composite service Π to be re-used by Π'. In Step 5, we exploit the neighbourhood of Π', starting from a new permutation $\Pi^{\star}$ encoded from Π' for performing local search. We will use **Example 6** to demonstrate the main steps in ALGORITHM 24.

Example 6. *Let us consider a composition task* $T = (\{a, b\}, \{e, f\})$ *and a service repository* $\mathcal{SR}$ *consisting of six atomic services.* $S_0 = (\{e, f\}, \{g\}, QoS_{S_0})$, $S_1 = (\{b\}, \{c, d\}, QoS_{S_1})$, $S_2 = (\{c\}, \{e\}, QoS_{S_2})$, $S_3 = (\{d\}, \{f\}, QoS_{S_3})$, $S_4 = (\{a\}, \{h\}, QoS_{S_4})$ *and* $S_5 = (\{c\}, \{e, f\}, QoS_{S_5})$. *The two special services* $Start = (\emptyset, \{a, b\}, \emptyset)$ *and* $End = (\{e, f\}, \emptyset, \emptyset)$ *are defined by a given composition task* T. *Fig. 5.2 illustrates a process of producing another permutation* Π' *that encodes a candidate permutation* Π *for a randomly sampled scenario. This permutation* Π' *is further encoded as permutation* $\Pi^{\star}$ *for performing local search.*

We firstly sample a scenario based on the service failure probability pr_S of each service S in a service repository $\mathcal{SR}$. Let $\{S_0, S_1, S_2, S_3, S_4\}$ be a sampled scenario based on pr_S of each S in $\mathcal{SR}$, so $\{0, 1, 2, 3, 4\}$ is a set of service indexes that corresponds to the sampled scenario in Fig. 5.2. In this example, we also take an arbitrary permutation $\Pi = [4, 1, 0, 2, 3, 5]$ as a candidate solution. Based on the sampled scenario, we produce another permutation Π' that encodes permutation Π for the scenario by only removing the service indexes of failed services (i.e., 5) from the permutation Π while other service indexes remain unchanged in the permutation Π. By doing these, the newly produced permutation Π' can keep some promising component services (e.g., 1) from the candidate Π in the corresponding DAG, i.e., $\mathcal{G}'$. This can be observed from two decoded DAGs, i.e., $\mathcal{G}$ and $\mathcal{G}'$, that have a service index 1 in common. In addition, service index 4, is removed since its outputs are not used in the composition.

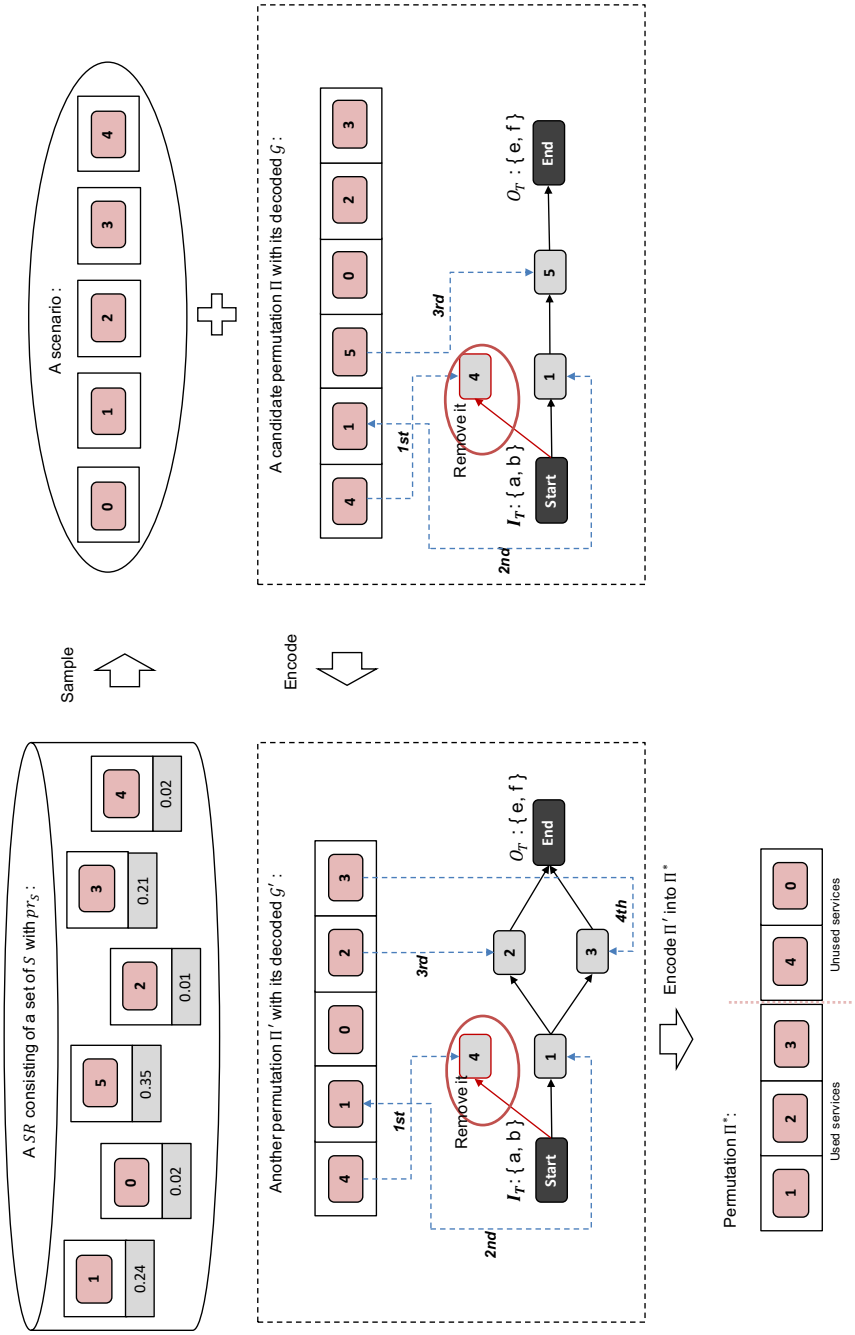


Fig. 5.2: A new permutation produced based on a sampled scenario.

Once the permutation Π' is produced, we begin with generating a starting permutation $\Pi^{\star}$ for a repairing process (i.e., local search). We encoded Π' into $\Pi^{\star}$ as $[1, 2, 3, |4, 0]$ ($|$ is just displayed for the courtesy of the reader, but not part of the representation). The detail of producing $\Pi^{\star}$ is exactly the same as the decoding part introduced in Sect. 3.5.2.

Consequently, our previously proposed stochastic local search with layer-based constrained one-point swap operator in Sect. 3.8.3 can now be performed on the new permutation $\Pi^{\star}$. We expect that this local search can help the permutation Π to maintain good comprehensive quality in the sampled scenario via effective neighbourhood exploitation.

5.5 GA-2Stage Algorithm to RWSC-SF

In this section, we first present our new robustness estimation method in Sect. 5.5.1. Subsequently, we present our GA-2Stage algorithm to the RWSC-SF problem in Sect. 5.5.2. Furthermore, we discuss the archive-based adaptive evolutionary control in Sect.5.5.3, Lastly, we demonstrate the details of our robustness estimation in Sect.5.5.4.

Monte Carlo based robustness estimation, introduced in Sect. 5.4.1, must determine its sampling size to ensure a good trade-off between the accuracy and computation time. However, when the size of the service repository increases, Monte Carlo sampling becomes computationally expensive for achieving an accurate robustness estimation. This is because a large sampling size is required to ensure that candidate composite services evolved by GA can be accurately ranked by the estimated robustness. Therefore, a new robustness estimation method with fewer sampled scenarios should be developed to achieve an ideal trade-off among accuracy and sampling cost. Particularly, we introduce a lower bound of the expected fitness as an estimated robustness. We expect that the improvement in lower bound leads to the improvement in true robustness with a high probability. This is achieved by carefully selecting different scenar-

ios, each of which only considers one service failure that is more likely to happen. Consequently, the robustness is estimated based on these selected scenarios with probabilities that measure their weights (see details in Sect. 5.5.1).

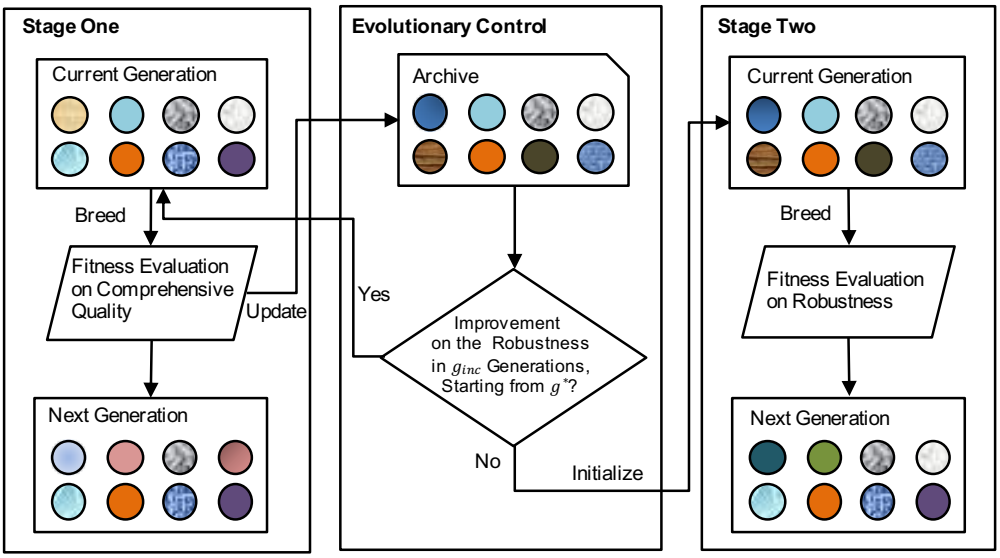


Fig. 5.3: Generation updates with an adaptive evolutionary control in GA-2Stage.

Moreover, to further reduce the computation time without hurting algorithm effectiveness, a GA-2Stage algorithm will be introduced with two consecutive evolutionary stages and two different evaluation methods. The robustness of evolved composite services in the first stage is estimated with a high level of noise based on individuals' comprehensive quality, without considering any service failure. However, the robustness of solutions in the second stage are more accurately estimated based on the lower bound robustness estimation method. More specifically, stage one tries to efficiently find good individuals that are more likely to have high robustness, i.e., composite services with a small number of component services. Such composite services are unlikely to be affected in the event of service failures due to their relatively small number of component services. They

will be stored in an archive and utilized to initialize the first population of stage two.

The generation updates with an adaptive evolutionary control over the two consecutive evolutionary stages in GA-2Stage are illustrated in Fig. 5.3. In stage one, an archive is updated by storing distinct and promising composite services evolved by the comprehensive quality. This archive is utilized to measure solution improvement in terms of their average robustness thereafter. Once the archive does not have any improvement on the robustness in g_{inc} consecutive generations, starting from generation $g^{\star}$, stage two will start by re-initializing the current generation with solutions in the archive.

5.5.1 Robustness Estimation

As shown in Eq. (5.3), the robustness is defined as the expected quality of a composite service across all possible scenarios. As discussed previously, we use a lower bound of expected fitness as an estimated robustness as follows:

$$\begin{aligned} r(\Pi) &= \sum_{Q \in \mathbb{Q}} f_{cq}(\mathscr{L}(\Pi, Q)) Pr(Q) \\ &\geqslant \sum_{Q^{\star} \in \mathbb{Q}^{\star}} f_{cq}(\mathscr{L}(\Pi, Q^{\star})) Pr(Q^{\star}) \\ &= \sum_{Q^{\star} \in \mathbb{Q}^{\star}} f_{cq}(\mathscr{L}(\Pi, Q^{\star})) pr_{S_i} \prod_{j \neq i} (1 - pr_{S_j}) \end{aligned} \tag{5.4}$$

where $\mathbb{Q}^{\star} \subseteq \mathbb{Q}$ are selected scenarios that only have one service failure, and any $Q^{\star} \in \mathbb{Q}^{\star}$ are not identical to each other. Therefore, the total number of scenarios in $\mathbb{Q}^{\star}$ equals to $|\mathcal{SR}|$. Let S_i be the failed service in every scenario, $Pr(Q^{\star})$ can be calculated based on a joint probability of services in $\mathcal{SR}$. Note that, when a service repository is very big, this joint probability might result in an arithmetic numerical overflow. To avoid this issue, we can calculate this joint probability in a logarithmic space. Thus, it becomes

a sum of the logarithms of each individual probability.

5.5.2 Outline of GA-2Stage

GA-2Stage is outlined in ALGORITHM 25. GA-2Stage takes six inputs: service composition task $T = (I_T, O_T)$, an ontology $\mathcal{O}$ that describes all the parameters of the web services, a service repository $\mathcal{SR}$, the generation counter g, the generation $g^\star$ at which stage two begins, and the number of neighbours n_{nb} to be exploited for repairing each solution in a scenario. In ALGORITHM 25, we start with setting the generation counter g to 0. Afterwards, we continue initializing an empty archive $\mathcal{A}$ that plays the role of evolutionary control, and population $\mathcal{P}^0$ with m randomly generated permutations Π_k^g (where $k = 1, \ldots, m$). The same permutation-based representation, discussed in Sect. 4.7.2, is utilized in this algorithm. In Step 3, we evaluate each permutation in the initialized population by decoding it into a functionally valid DAG. The DAG enables an easy evaluation on the comprehensive quality defined in Eq. (3.5). The purpose of utilizing this evaluation method has already been discussed at the beginning of Sect. 5.5. The iterative steps (Step 5 to 15) will be repeated until the maximum number of generations has been reached. During each iteration, we start with updating both the archive and the generation number $g^\star$, at which stage two begins, based on an evolutionary control using ALGORITHM 26 (see details in Sect. 5.5.3). In particular, $g^\star$ can be adaptively updated to control the lengths of stage one (when $g < g^\star$) and stage two (when $g \geq g^\star$). In stage one, we produce m new permutations to form the next generation $\mathcal{P}^{g+1}$ via genetic operators, introduced in Sect. 4.4.3. All the permutations in the newly formed population are then evaluated using Eq. (3.5). The initial population in stage two is constructed from the archive, and all the permutations in $\mathcal{P}^{g+1}$ are then evaluated using Eq. (5.4). Afterwards, the breed process of stage two is similar with that of stage one, but populated permutations are evaluated using Eq. (5.4). After

the iteration, the best solution Π^{opt} in $\mathcal{P}^g$ is returned as a baseline solution for the design stage.

ALGORITHM 25. GA-2Stage for the RWSC-SF problem.

Input : composition task T, Ontology $\mathcal{O}$, service repository $\mathcal{SR}$, the generation counter g, the generation $g^\star$, generation increment step g_{inc}, and the number of neighbors n_{nb}

Output: a baseline solution

1: Set generation counter $g \leftarrow 0$;
2: Initialize an empty archive $\mathcal{A}$ and a $\mathcal{P}^g$ with m random permutations, each represented as a Π_k^g (where $k = 1, \ldots, m$);
3: Evaluate each permutation in $\mathcal{P}^g$ using Eq. (3.5) based on its decoded DAG, $\mathcal{G}_k^g$;
4: **while** $g < g_{max}$ **do**
5: Set $g^\star$ based on an evolutionary control using ALGORITHM 26 ;
6: **if** $g < g^\star$ **then** `// stage one`
7: Populate $\mathcal{P}^{g+1}$ with m permutations from $\mathcal{P}^g$ through the use of genetic operators;
8: Evaluate each permutation in $\mathcal{P}^{g+1}$ using Eq. (3.5);
9: **if** $g = g^\star$ **then** `// stage two starts`
10: Populate $\mathcal{P}^{g+1}$ with m permutations from $\mathcal{A}$;
11: Evaluate each permutation in $\mathcal{P}^{g+1}$ using Eq. (5.4);
12: **if** $g > g^\star$ **then** `// stage two`
13: Populate $\mathcal{P}^{g+1}$ with m permutations from $\mathcal{P}^g$ through the use of genetic operators;
14: Evaluate each permutation in $\mathcal{P}^{g+1}$ using Eq. (5.4);
15: Set $g \leftarrow g + 1$;
16: Select the best solution Π^{opt} in $\mathcal{P}^g$ as a baseline;

5.5.3 Archive-based adaptive evolutionary control

The archive-based evolutionary control proposed in our GA-2Stage algorithm is used to adaptively update the generation number $g^\star$, at which stage two should begin. Specifically, $g^\star$ should be increased by g_{inc} generations based on the robustness changes in the archive, starting with a predefined generation number. Such an updating mechanism for $g^\star$ allows evolved solutions at the updated generation $g^\star$ achieve the highest possible robustness with the least computation resources due to the cheap evaluation method assigned to stage one.

ALGORITHM 26. Generating $g^\star$ based on an adaptive archive-based evolutionary control.

Input : archive $\mathcal{A}$ with maximal size m, population $\mathcal{P}^g$, generation counter g, initial generation number $g^\star$ and generation increment step g_{inc}
Output: generation number $g^\star$

1: **if** $g < g^\star$ **then**
2: Update $\mathcal{A}$ with $\mathcal{P}^g$;
3: **if** $g = g^\star$ **then**
4: Evaluate each permutation in $\mathcal{A}$ using Eq. (5.4);
5: Calculate average robustness $\overline{R}$ for permutations in $\mathcal{A}$;
6: Update $\mathcal{A}$ with $\mathcal{P}^g$;
7: Evaluate each permutation in $\mathcal{A}$ using Eq. (5.4);
8: Calculate average robustness $\overline{R}'$ for permutations in $\mathcal{A}$;
9: **if** $\overline{R}' > \overline{R}$ **then**
10: $g^\star \leftarrow g^\star + g_{inc}$;
11: **return** $g^\star$;

This evolutionary control is outlined in ALGORITHM 26. This algorithm takes three inputs: an archive of size m that stores good individuals, the

current population $\mathcal{P}^g$, the generation $g^\star$ at which stage two should start, and generation increment step g_{inc} for $g^\star$. When $g < g^\star$, this algorithm updates the archive by storing all distinct composite services from $\mathcal{P}^g$ based on the comprehensive quality in a descending order. Subsequently, when $g = g^\star$, we evaluate the robustness of each permutation in the archive using Eq. (5.4) and calculate the average robustness of all the permutations as $\overline{R}$. After calculating the average robustness $\overline{R}$, we calculate the average robustness again as $\overline{R}'$ after updating the archive. The archive is updated in the same way as we discussed above. Consequently, we increase $g^\star$ by g_{inc} if $\overline{R}' > \overline{R}$. Otherwise, $g^\star$ remains unchanged.

5.5.4 Robustness Estimation based on a Lower Bound

In ALGORITHM 27, we outline the process of calculating the robustness of each permutation Π in a population $\mathcal{P}^g$ through Eq. (5.4). We firstly produce $|\mathcal{SR}|$ scenarios, each of which only considers the failure of a different service. For each scenario, we identify the best-repaired solution (i.e., a neighbouring solution with the highest comprehensive quality) through exploiting n_{nb} neighbours (Step 4 to 6). Afterwards, we calculate $Pr(Q)$ as the weight of each scenario. Consequently, the comprehensive quality of all the repaired solutions for all the scenarios are summed up with different weights $Pr(Q)$ using Eq. (5.4).

Example 7. *Let us consider a composition task* $T = (\{a, b\}, \{e, f\})$ *and a service repository* $\mathcal{SR}$ *consisting of six atomic services.* $S_0 = (\{e, f\}, \{g\}, QoS_{S_0})$, $S_1 = (\{b\}, \{c, d\}, QoS_{S_1})$, $S_2 = (\{c\}, \{e\}, QoS_{S_2})$, $S_3 = (\{d\}, \{f\}, QoS_{S_3})$, $S_4 = (\{a\}, \{h\}, QoS_{S_4})$ *and* $S_5 = (\{c\}, \{e, f\}, QoS_{S_5})$. *The two special services* $Start = (\emptyset, \{a, b\}, \emptyset)$ *and* $End = (\{e, f\}, \emptyset, \emptyset)$ *are defined by a given composition task* T. *Fig. 5.4 illustrates a process of producing another permutation* Π' *that encodes a candidate permutation* Π *for the two sampled scenario. This permutation* Π' *is further encoded as permutation* $\Pi^\star$ *for performing local search.*

Based on the size of the given service repository $\mathcal{SR}$, we can produce

ALGORITHM 27. Robustness estimation based on a lower bound (Step 11 and 14 in ALGORITHM 25).

Input : population $\mathcal{P}^g$, the number of neighbor n_{nb} and service repository $\mathcal{SR}$
Output: evaluated $\mathcal{P}^g$

1: **for each** Π *in* $\mathcal{P}^g$ **do**
2: Sample $|\mathcal{SR}|$ scenarios based on the number of S in $\mathcal{SR}$;
3: **foreach** *scenario* Q *in the* $|\mathcal{SR}|$ *sampled scenarios* **do**
4: Produce another permutation Π' that encodes Π based on Q;
5: Generate a size n_{nb} of neighbors from $\Pi^\star$ (i.e., an encoded solution of Π') by local search operator;
6: Identify the best neighbor Π' with the highest fitness measured by Eq. (3.5);
7: Calculate $Pr(Q)$ as the weight for scenario Q;
8: Set the robustness of Π as a weighted-sum fitness value of $|\mathcal{SR}|$ Π' based on Eq. (5.4);
9: **return** evaluated $\mathcal{P}^g$;

six scenarios. Let $\{S_1, S_2, S_3, S_4, S_5\}$ be a produced scenario with S_0 becoming inaccessible. Therefore, $\{1, 2, 3, 4, 5\}$ is a set of service indexes corresponding to the produced scenario in Fig. 5.4. In a similar way, $\{0, 1, 2, 3, 4\}$ is a set of service indexes corresponding to another produced scenario with the failure of S_5. To demonstrate the repairing process, we also take an arbitrary permutation $\Pi = [4, 1, 0, 2, 3, 5]$ as an example. To encode the two scenarios for the permutation Π, we produce another permutation Π' for each scenario by only removing the service indexes of failed services from the permutation, keeping the order of other elements in the permutation. This encoding process for scenarios is exactly the same as that in Sect. 5.4.3. Therefore, two permutations, Π', with two

decoded DAGs, $\mathcal{G}'$, are created for encoding two scenarios Q_1 and Q_6 respectively. In the two decoded DAGs, we can see that the decoded DAG $\mathcal{G}'$ that corresponds to Q_1 remains the same as the original $\mathcal{G}$, as the failure of S_0 has no impact on the execution of the original $\mathcal{G}$. In such a case, a repairing process is not involved. In contrast, the decoded DAG $\mathcal{G}'$ that corresponds to Q_6 is not identical to the original $\mathcal{G}$, as the failure of S_5 prevents the successful execution of the original $\mathcal{G}$. Therefore, a repairing process will be involved by preparing a starting point, permutation $\Pi^{\star}$, see our previous discussion in Sect. 5.4.3.

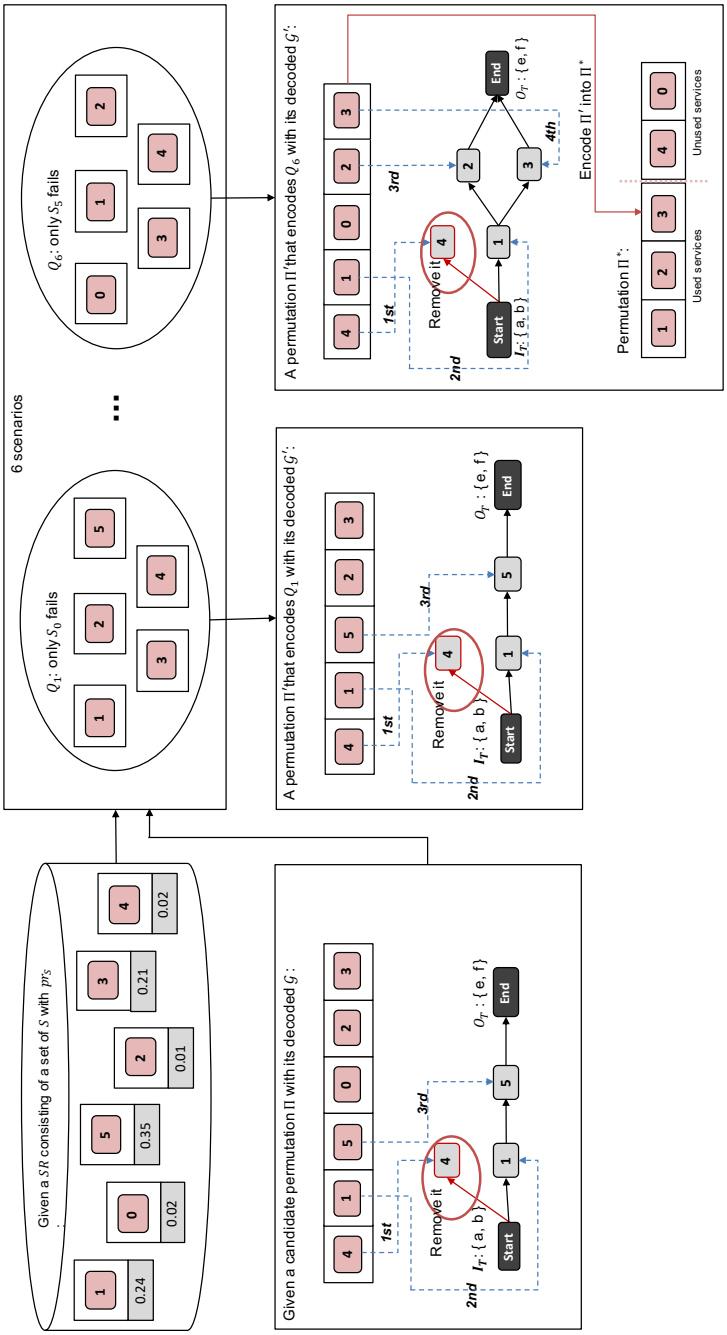


Fig. 5.4: A new permutation produced based on a sampled scenario.

5.6 Experimental Evaluation

In this section, we conduct two experiments to test GA-MC and GA-2Stage for solving the small-scale and large-scale RWSC-SF problem, respectively. In particularly, for the first experiment, we compare GA-MC with FL on a small benchmark, see details in Sect. 5.6.1. For the second experiment, we compare GA-2Stage with GA-RE, GA-MC and FL on both small and large benchmarks, see details in Sect. 5.6.2.

5.6.1 Comparing GA-MC against FL

Experimental Design

We use five composition tasks with corresponding service repositories for testing the small-scale RWSC-SF problem. These tasks (i.e., OWL-S TC1 to OWL-S TC5 utilized in [117, 152]) contain real-world web services and composition tasks originally collected from OWLS-TC [97]. Each service in the service repository is extended with real-world QoS attributes obtained from the QWS dataset [4]. Apart from that, each service is also associated with a separate service failure rate. The failure rate of a service is generated from the normal distribution $\mathcal{N}(\mu, \sigma^2)$ truncated in the interval $[0, 1]$ with mean μ and variance σ^2. According to the failure rates reported in [232] and by using $15\,000$ failure probabilities observed by 150 users on 100 web services, μ and σ are set to 0.0405 and 0.1732.

To perform the comparisons between GA-MC and FL, we follow the popular parameter settings in the literature [47, 95]: population size is set to 30, crossover and mutation rate are set to 0.95 and 0.05 respectively, tournament size is set to 2 and elitism is set to 2. We set the maximum generation to 100. For the robustness estimation in Eq. (5.3), a set of sample size N (i.e., 10, 30, 50, 70 and 90) is to be investigated in Sect. 5.6.1, and the number of local search steps (i.e., n_{nb}) is set to 10 that empirically produces a good compromise between computation cost and service qual-

ity. The weights settings in Eq. (3.5) follow our previous suggestion in Sect. 3.7. We have also conducted tests with other weights and parameters and generally observed the same behavior.

We run both GA-MC and FL 30 times with 30 different random seeds. We then test each baseline composite service obtained by every run of every algorithm over 200 simulated scenarios. Note that, a large number of sampled scenarios (e.g., 200) is taken into account for testing while a small number of sampled scenarios N is used at the design stage. This difference is important for the design stage in order to remain highly efficient, whereas we want to accurately measure the robustness of any composite service during the execution stage. Subsequently, we use an independent sample T-test with a significance level of 5% to verify the observed difference in the mean fitness values obtained on the baselines found by GA-MC and FL.

Parameters sensitivity

To evaluate the impact of N in Eq. (5.3) on the testing performance, we perform parameters sensitivity tests on OWL-S TC3 using different settings of N in GA-MC.

In Fig. 5.5, we present a box plot of the testing performance from testing baseline solutions found by GA-MC with varied settings of N (i.e., 10, 30, 70 and 90) across 30 independent algorithm runs. It is easy to observe that the performance boxes tend to reduce their sizes when N increases. This observation agrees with our expectation that a more accurate fitness evaluation with large N will enhance the reliability of our algorithm. Meanwhile, we can also observe that the medium values in these boxes are also positively correlated to N. This observation further confirms that more accurate fitness evaluations contribute to better algorithm performance. In the remaining experiments in Sect. 5.6.1, we set N to 50 according to Fig. 5.5, since 50 presents the ideal trade-off between the algorithm performance and sample cost.

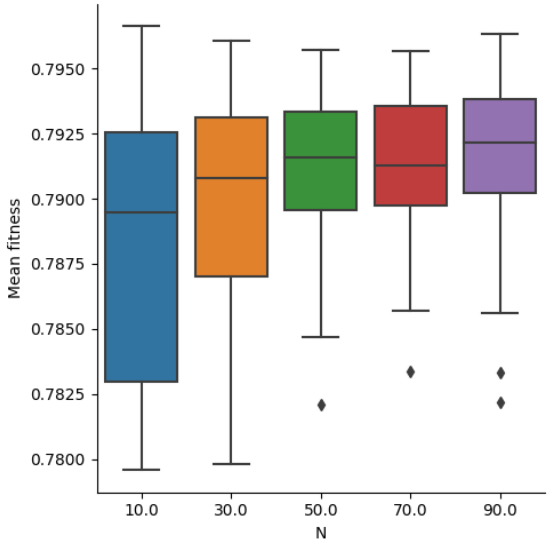


Fig. 5.5: Mean fitness values tested on near-optimal solutions found by GA-MC over a set of increasing N for OWLS-TC 03

Comparison of the effectiveness

Table. 5.1 shows the mean fitness values and standard deviations obtained with respect to the evolved baseline solutions. We verify the significant differences in the fitness values using an independent sample T-test, and the winner is highlighted in a green colour.

Table 5.1: Mean fitness values tested based on the baseline solutions for our approach in comparison to FL.
(Note: the higher the fitness the better)

Task	GA-MC	FL [47]
OWL-S TC1	0.922799 ± 0.000304	0.922791 ± 0.000311
OWL-S TC2	0.930779 ± 0.000998	0.929618 ± 0.005009
OWL-S TC3	0.864505 ± 0.001448	0.854218 ± 0.00779
OWL-S TC4	0.790862 ± 0.003172	0.779121 ± 0.012348
OWL-S TC5	0.82504 ± 0.005556	0.812852 ± 0.012388

At the execution stage, GA-MC can produce composite services that are clearly more robust to stochastic service failures as evidenced by the performance summarized in Table 5.1. Particularly, baseline solutions produced by GA-MC achieved significantly higher mean fitness values against 200 random scenarios for 3 out of 5 tasks. Therefore, composite services produced by GA-MC is more likely to maintain a good quality in the event of stochastic service failures. This finding matches well with the objective of the design stage for GA-MC.

Moreover, for the two tasks (i.e., OWL-TC1, OWL-TC2), GA-MC and FL are comparable to each other. Particularly, both GA-MC and FL can maintain very high quality over all tested scenarios with very small standard deviations. This is because the search space of feasible solutions in OWL-TC1, OWL-TC2 is small, and these two methods can always find high-quality solutions through local search in the event of service failures at the execution stage.

Comparison of the efficiency

Table 5.2 and 5.3 show two groups of execution time observed for the design stage and execution stage, respectively, using both GA-MC and FL [47]. We keep using an independent sample T-test to detect any noticeable differences in the experiment results in efficiency.

For the design stage, we note that FL consistently takes significant less execution time (in seconds) for all the tasks. This is because the fitness evaluation in FL through Eq. (3.5) is far more efficient than GA-MC through Eq. (5.3). In contrast, GA-MC consistently requires much more execution time. This observation indicates a sensible trade-off because the frequency of producing the baseline is far less frequent than that of repairing the baseline solutions by local search. Furthermore, although GA-MC consumes much longer execution time at the design stage, GA-MC gains much higher quality against the stochastic service failures at the execution stage.

Table 5.2: Mean execution time (in seconds) observed for our approach in comparison to FL at the design stage.
(Note: the shorter the time the better)

Task	GA-MC	FL [47]
OWL-S TC1	221.854233 ± 63.968435	2.279767 ± 0.594116
OWL-S TC2	51.851 ± 34.814491	1.502733 ± 0.163235
OWL-S TC3	27.075967 ± 14.63108	1.4005 ± 0.132212
OWL-S TC4	468.054967 ± 342.97007	13.785767 ± 21.966587
OWL-S TC5	$901.813933 \pm 598.884817$	19.577733 ± 71.642104

Table 5.3: Mean execution time (in milliseconds) per scenario by local search based on the baseline solutions found by our approach in comparison to FL.
(Note: the shorter the time the better)

Task	GA-MC	FL [47]
OWL-S TC1	0.155067 ± 0.06195	0.194944 ± 0.095481
OWL-S TC2	0.456811 ± 0.323291	1.173133 ± 1.618681
OWL-S TC3	0.788439 ± 0.574859	1.363739 ± 0.892455
OWL-S TC4	9.315556 ± 7.508798	10.824494 ± 5.943972
OWL-S TC5	12.694856 ± 10.350321	22.812806 ± 21.672252

For the execution stage, GA-MC requires significantly less execution time (in milliseconds) than FL for 3 out of 5 tasks per scenario. This observation indicates that baseline solutions produced by GA-MC are more likely to have services, required to build a robust DAG placed at the very front of the corresponding permutations. This can potentially accelerate the process of decoding from permutations to DAGs.

5.6.2 Comparing GA-2Stage with GA-RE, GA-MC and FL

Experimental Design

In addition to the OWL-S TC benchmark used in Sect. 5.6.1, two more benchmarks, WSC-08 [15] and WSC-09 [92], are used in this experiment for testing the performance of competing methods on large-scale RWSC-SF problem. Unlike OWL-S TC, WSC-08 and WSC-09 includes eight and five composition tasks respectively with web services that are simulated for the Web Service Challenges competition. Particularly, WSC-08 contains 8 composition tasks with an increasing size of service repository, i.e., 158, 558, 608, 1041, 1090, 2198, 4113, and 8119, and WSC-09 contains 5 composition tasks with an increasing size of service repository, i.e., 572, 4129, 8138, 8301, and 15211, respectively. Following our discussion in Sect. 5.6.1, each service in WSC-08 and WSC-09 are also extended with the QoS and failure rate in the same way.

To perform fair comparisons among GA-2Stage, GA-RE, GA-MC and FL, we follow the parameter setting suggested in Sect. 5.6.1: population size m is set to 30, tournament size is set to 2 and elitism is set to 2. The crossover and mutation rates are inspired by Koza's operator settings [95] and are set to 0.95 and 0.05, respectively. In Eq. (5.4), the number of scenarios equals the size of the service repository, and the number of scenarios that trigger local search equals to the number of component service utilized by the solution. Therefore, to ensure a fair comparison with GA-2Stage and GA-MC, the number of scenarios N in Eq. (5.3) for GA-MC does not follow the suggested size, 50, in Sect. 5.6.1. Instead, we ensure a doubled number of scenarios that employ local search in GA-MC for a more reliable estimation of robustness. To do that, we keep sampling scenarios randomly until twice the number of scenarios are triggered by local search. Other settings, such as n_{nb} for the neighbourhood size and weights in Eq. (3.5), follow our suggestion in Sect. 5.6.1. In addition, for the archive-based evolutionary control, the initial generation number $g^{\star}$

and increased generation number g_{inc} are set to 60 and 6 respectively. The parameters sensitivity of $g^{\star}$ and g_{inc} have been studied by considering different pair of parameters. This setting present the idea trade-off between the effectiveness and efficiency in GA-2Stage.

Following how the comparisons are conducted in Sect. 5.6.1, we run GA-2Stage, GA-RE, GA-MC and FL 30 times with 30 different random seeds. We then test each baseline composite service obtained by every run of every algorithm over 200 different simulated scenarios. Note that, scenarios considered at the design stage are different from those 200 simulated scenarios sampled for testing, such differences are important because we want to accurately measure the robustness of any composite service during the execution stage. Subsequently, we use an independent sample T-test with a significance level of 5% to verify the observed difference in the effectiveness and execution time with respect to GA-2Stage, GA-RE, GA-MC and FL. Lastly, we also compare the accuracy of lower bound robustness estimation in Eq. (5.4) to that of the Monte Carlo estimation in Eq. (5.3). Particularly, we demonstrate their accuracy in ranking candidate solutions evolved by GA using three most popular rank correlation methods, i.e., Pearson, Kendall's tau and Spearman's rho [21], with a significance level of 5%.

Comparison of the effectiveness

To study the effectiveness of GA-2Stage, GA-RE, FL and GA-MC in finding robust baseline solutions, Table 5.4 shows the mean fitness values and standard deviations obtained from testing on baseline solutions for GA2-Stage, GA-RE, FL and GA-MC over 30 runs, and each run is tested over 200 random scenarios of service failures at the execution stage. We verify the significant differences in the fitness values using an independent sample T-test, and the winner is highlighted in the table. In particular, we use pairwise comparisons among GA-2Stage, GA-RE, GA-MC and FL using an independent-sample T-test with a significance level of 5% to verify the observed differences in performance concerning fitness values. After-

wards, the top performances are identified, and its related value is highlighted in a green color in Table 5.4. The pairwise comparison results for fitness are summarized in Table 5.5, where *win/draw/loss* shows the scores of one method compared to all the others, and displays the frequency that this method outperforms, equals or is outperformed by the competing method. Note that all the P-values are lower than 0.001, and any "$-$" in the tables means results cannot be collected since the related testing instances has been running for more than 16 days.

Compared to FL and GA-MC, GA-2Stage and GA-RE outperform them as evidenced by the performance, summarised in Table 5.5. Particularly, baseline solutions produced by GA-2Stage and GA-RE both achieve consistently good performance for all the tasks in OWLS-TC, WSC-08, and WSC-09 as top performers regardless of the size of the service repository. In other words, composite services produced by GA-2Stage and GA-RE are more likely to maintain a good quality in the event of stochastic service failures. This finding matches well with our expectation that our robustness estimation methods in GA-2Stage and GA-RE is very effective in dealing with service requests over both small and large service repositories.

In addition, the effectiveness of GA-2Stage and GA-RE are very comparable to each other. This observation agrees with our expectation that GA-2Stage can maintain the effectiveness of GA-RE in finding robust composite services by two different evaluation methods over two consecutive evolutionary processes.

Moreover, for the two baseline methods FL and GA-MC, GA-MC outperforms FL in OWLS-TC benchmark while the same performance cannot be clearly observed from WSC-08 and WSC-09 benchmarks. This is because the robustness measure using Eq. (5.3) by GA-MC presents low variances for the OWLS-TC benchmarks. Therefore, for large-scale RWSC-SF problem, the robustness measure in GA-MC presents high variances and cannot be reliably used by GA to produce robust composite services.

Table 5.4: Mean fitness values tested based on the baseline solutions for GA-2Stage in comparison to GA-RE, FL and GA-MC. (Note: the higher the fitness the better)

Task	GA-2Stage	GA-RE	FL [47]	GA-MC
OWLS-TC 1	0.922788 ± 0.000179	0.922862 ± 0.00018	0.922791 ± 0.000311	0.910623 ± 0.033122
OWLS-TC 2	0.931586 ± 0.002107	0.932095 ± 0.000277	0.929618 ± 0.005009	0.915558 ± 0.025327
OWLS-TC 3	0.863518 ± 0.003623	0.863061 ± 0.00725	0.854218 ± 0.00779	0.862459 ± 0.003403
OWLS-TC 4	0.789396 ± 0.00857	0.791174 ± 0.004451	0.779121 ± 0.012348	0.789101 ± 0.005596
OWLS-TC 5	0.826731 ± 0.005054	0.826509 ± 0.005275	0.812852 ± 0.012388	0.824296 ± 0.013078
WSC08-1	0.397971 ± 0.002975	0.398896 ± 0.003538	0.395194 ± 0.002782	0.383523 ± 0.005433
WSC08-2	0.576417 ± 0.00378	0.576716 ± 0.00289	0.568166 ± 0.005362	0.572707 ± 0.006905
WSC08-3	0.025494 ± 0.00295	0.02501 ± 0.001415	0.025118 ± 0.001333	0.02509 ± 0.001424
WSC08-4	0.274788 ± 0.00295	0.275631 ± 0.002872	0.271112 ± 0.004133	0.268906 ± 0.003492
WSC08-5	0.275513 ± 0.002107	0.275814 ± 0.003147	0.275259 ± 0.002625	0.272394 ± 0.003412
WSC08-6	0.072178 ± 0.002047	0.072036 ± 0.001605	0.072017 ± 0.002114	0.071496 ± 0.002028
WSC08-7	0.216909 ± 0.003111	0.217957 ± 0.00322	0.216321 ± 0.003162	0.214177 ± 0.00333
WSC08-8	0.053596 ± 0.002047	0.054259 ± 0.001909	0.0536 ± 0.002007	0.053406 ± 0.001992
WSC09-1	0.514032 ± 0.005805	0.514389 ± 0.006263	0.501957 ± 0.005435	0.509053 ± 0.007493
WSC09-2	0.272445 ± 0.00309	0.272217 ± 0.002653	0.272905 ± 0.002886	0.269513 ± 0.003516
WSC09-3	0.387672 ± 0.003014	0.386712 ± 0.002356	0.384879 ± 0.002562	0.378036 ± 0.005318
WSC09-4	0.068139 ± 0.002223	0.067328 ± 0.001968	0.067515 ± 0.002214	-
WSC09-5	0.108845 ± 0.002602	0.109347 ± 0.002619	0.107809 ± 0.002011	0.108056 ± 0.002067

Table 5.5: Summary of statistical significance tests for mean fitness values, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09.

Dataset	Method	GA-2Stage	GA-RE	FL [47]	GA-MC
OWLS-TC (5 tasks)	GA-2Stage	-	0/5/0	0/1/4	0/3/2
	GA-RE	0/5/0	-	0/1/4	0/3/2
	FL [47]	4/1/0	4/1/0	-	2/3/0
	GA-MC	2/3/0	2/3/0	0/3/2	-
WSC08 (8 tasks)	GA-2Stage	-	0/8/0	0/5/3	0/3/5
	GA-RE	0/8/0	-	0/5/3	0/3/5
	FL	3/5/0	3/5/0	-	0/3/5
	GA-MC	5/3/0	5/3/0	5/3/0	-
WSC09 (5 tasks)	GA-2Stage	-	0/5/0	0/3/2	0/1/4
	GA-RE	0/5/0	-	0/2/3	0/1/4
	FL [47]	2/3/0	3/2/0	-	1/1/3
	GA-MC	4/1/0	4/1/0	3/1/1	-

Comparison of the efficiency

To study the efficiency of GA-2Stage, GA-RE, FL and GA-MC at both the design stage and execution stage, Table 5.6 and 5.7 show execution times observed for the design stage and execution stage, respectively, over 30 runs. We keep utilizing the pairwise comparisons with an independent sampled T-test to detect any noticeable differences in the algorithm running time, see Table 5.8 and 5.9.

At the design stage, see Table 5.6 and Tables 5.8, we can observe that FL consistently takes significantly less execution time (in seconds) comparing to all the other methods. This is because the fitness evaluation in FL through Eq. (3.5) is far more efficient than GA-Stage, GA-RE and GA-MC.

Furthermore, GA-MC consistently requires the longest execution time in the design stage, while GA-RE requires the second longest execution time in the design stage. This is because a single evaluation of one candidate solution involves N calculations of comprehensive quality using Eq. (5.3). This N is much larger than the number of scenarios required by

GA-RE in Eq. (5.4). As discussed previously, we increase N in Eq. (5.3) to allow more accurate robustness estimation in GA-MC. However, GA-MC still does not outperform GA-2Stage for finding high-robustness solutions on large benchmarks, such as WSC-08 and WSC-09.

In addition, GA-RE requires less significant execution time at the design stage, compared to GA-MC. This is because the efficiency of evolving robust composite services is further improved with the help of the lower bound robustness estimation in GA-RE.

Lastly, GA-2Stage further improves the efficiency of GA-RE by introducing a two-stage optimization process with the adaptive evolutionary control. This is because the majority of the generations in GA-2Stage employ a cheap evaluation on the comprehensive quality of each solution using Eq. (3.5), while the rest of the generations employ the fitness approximation using Eq. (5.4). Notably, GA-2Stage can still maintain high effectiveness in finding high-robustness baseline solutions.

At the execution stage, see Table 5.7, no results are highlighted in the tables since we do not observe any significant difference among any two competing methods. This indicates that baseline solutions produced by all the methods for every task need a similar amount of time to be repaired in the event of service failures. The amount of repair time is independent of the design stage. Moreover, the amount of repair time is much less than the execution time of the design stage, and the frequency of producing baseline solutions at the design stage is also far less than that of repairing the baseline solutions at the execution stage.

Table 5.6: Mean execution time (in s) observed for GA-2Stage in comparison to GA-RE, FL and GA-MC at the design stage. (Note: the shorter the time the better)

Task	GA-2Stage	GA-RE	FL [47]	GA-MC
OWLS-TC 1	101.009067 ± 23.07637	221.854233 ± 63.968435	2.279767 ± 0.594116	319.139533 ± 76.385136
OWLS-TC 2	35.9972 ± 31.302637	51.851 ± 34.814491	1.502733 ± 0.163235	115.441333 ± 71.247137
OWLS-TC 3	13.0314 ± 4.849419	27.075967 ± 14.63108	1.4005 ± 0.132212	55.393333 ± 24.899348
OWLS-TC 4	173.224267 ± 116.190079	468.054967 ± 342.97007	13.785767 ± 21.966587	786.6216 ± 365.605208
OWLS-TC 5	336.6059 ± 206.179729	901.813933 ± 598.884817	19.577733 ± 71.642104	1263.4348 ± 807.615199
WSC08-1	362.312133 ± 77.091671	1153.114633 ± 157.16768	14.593633 ± 7.036069	2401.1046 ± 290.709963
WSC08-2	136.833233 ± 32.808761	346.3632 ± 88.461812	9.5453 ± 5.685242	630.930267 ± 98.000929
WSC08-3	5445.803933 ± 2356.34388	20646.882333 ± 2831.701793	297.080867 ± 50.772516	8243.711267 ± 827.243178
WSC08-4	363.368233 ± 46.265638	946.272833 ± 116.302594	22.1297 ± 5.09158	1223.7892 ± 119.670871
WSC08-5	5411.626833 ± 1754.906719	21636.348867 ± 2504.446328	287.389433 ± 69.087335	33964.1487 ± 5147.121731
WSC08-6	58857.504067 ± 42330.514059	309276.646467 ± 36848.385724	3056.054767 ± 744.411865	304471.121333 ± 37767.070623
WSC08-7	8399.0462 ± 2735.343313	26800.4088 ± 4888.82274	477.589233 ± 166.107954	36787.8031 ± 5155.913198
WSC08-8	10500.235667 ± 4032.270104	39111.0008 ± 5217.311172	572.005833 ± 96.59887	22512.887867 ± 2243.886478
WSC09-1	327.1701 ± 80.081116	943.368733 ± 269.944223	15.832667 ± 12.600583	1633.791 ± 284.342794
WSC09-2	11490.332567 ± 3504.752592	40003.2077 ± 5467.541146	499.271067 ± 183.33795	61763.3132 ± 7293.294917
WSC09-3	6117.213533 ± 1423.425485	17718.882467 ± 3384.984318	401.276233 ± 238.865704	38380.1644 ± 7324.715594
WSC09-4	223757.395885 ± 192691.648686	1420123.714684 ± 292166.614721	11110.008733 ± 2371.111691	-
WSC09-5	50052.6394 ± 17629.244248	174415.461533 ± 34932.061346	1972.0222 ± 713.868674	160876.807733 ± 16575.127178

Table 5.7: Mean execution time (in ms) per scenario by local search based on the baseline solutions found by GA-2Stage in comparison to GA-RE, FL and GA-MC.
(Note: the shorter the time the better)

Task	GA-2Stage	GA-RE	FL [47]	GA-MC
OWLS-TC 1	0.185822 ± 0.075933	0.155067 ± 0.06195	0.194944 ± 0.095481	2.339717 ± 6.703382
OWLS-TC 2	0.95495 ± 1.129079	0.456811 ± 0.323291	1.173133 ± 1.618681	4.135078 ± 7.997827
OWLS-TC 3	1.160528 ± 0.836618	0.788439 ± 0.574859	1.363739 ± 0.892455	0.926944 ± 0.666366
OWLS-TC 4	8.860739 ± 5.273149	9.315556 ± 7.508798	10.824494 ± 5.943972	10.265489 ± 6.760639
OWLS-TC 5	15.748111 ± 12.911706	12.694856 ± 10.350321	22.812806 ± 21.672252	16.579461 ± 22.341204
WSC08-1	21.327167 ± 2.756077	21.454789 ± 2.873897	20.090467 ± 2.6058	25.159133 ± 3.281322
WSC08-2	12.451461 ± 3.029406	12.090028 ± 3.017539	12.761439 ± 2.55482	11.707733 ± 3.262212
WSC08-3	32.899689 ± 3.254111	33.112011 ± 4.172865	32.191772 ± 3.75622	33.462789 ± 4.481216
WSC08-4	16.180183 ± 2.256799	14.979589 ± 1.932228	15.879867 ± 2.083646	16.741572 ± 2.032044
WSC08-5	179.797372 ± 17.259394	175.223117 ± 20.767383	184.219939 ± 17.570621	184.892367 ± 24.642377
WSC08-6	911.157344 ± 121.979441	891.088928 ± 75.58605	929.152878 ± 85.605171	907.872372 ± 84.003491
WSC08-7	214.58365 ± 25.636885	207.045233 ± 21.754379	205.707067 ± 23.674507	221.702394 ± 32.026932
WSC08-8	88.189956 ± 10.84816	88.570883 ± 7.407168	88.2868 ± 8.049778	91.657628 ± 8.401898
WSC09-1	25.138389 ± 6.636436	25.804 ± 6.851249	24.943722 ± 5.962107	29.2981 ± 7.077208
WSC09-2	297.721911 ± 33.783551	315.269294 ± 31.808168	297.211728 ± 21.107784	312.352106 ± 38.903679
WSC09-3	302.659394 ± 57.691578	306.980983 ± 35.572703	297.8739 ± 42.378935	315.209356 ± 48.449898
WSC09-4	3411.716628 ± 447.572982	3560.59905 ± 449.65906	3528.859594 ± 620.60136	-
WSC09-5	603.66355 ± 117.838659	606.873822 ± 67.904197	581.632528 ± 64.545121	608.908839 ± 50.244834

Table 5.8: Summary of statistical significance tests for mean execution time of the design stage, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09.

Dataset	Method	GA-2Stage	GA-RE	FL [47]	GA-MC
OWLS-TC (5 tasks)	GA-2Stage	-	0/0/5	5/0/0	0/0/5
	GA-RE	5/0/0	-	5/0/0	0/0/5
	FL [47]	0/0/5	0/0/5	-	0/0/5
	GA-MC	5/0/0	5/0/0	5/0/0	-
WSC08 (8 tasks)	GA-2Stage	-	0/0/8	8/0/0	0/0/8
	GA-RE	8/0/0	-	8/0/0	0/0/8
	FL	0/0/8	0/0/8	-	0/0/8
	GA-MC	8/0/0	8/0/0	8/0/0	-
WSC09 (5 tasks)	GA-2Stage	-		5/0/0	0/0/5
	GA-RE	5/0/0	-	5/0/0	0/0/5
	FL [47]	0/0/5	0/0/5	-	0/0/5
	GA-MC	5/0/0	5/0/0	5/0/0	-

Table 5.9: Summary of statistical significance tests for mean execution time per scenario for local search at the execution stage, where each column shows the win/draw/loss score of one method against a competing one for all tasks of OWLS-TC, WSC08 and WSC09.

Dataset	Method	GA-2Stage	GA-RE	FL	GA-MC
OWLS-TC (5 tasks)	GA-2Stage	-	0/5/0	0/5/0	0/5/0
	GA-RE	0/5/0	-	0/5/0	0/5/0
	FL	0/5/0	0/5/0	-	0/5/0
	GA-MC	0/5/0	0/5/0	0/5/0	-
WSC08 (8 tasks)	GA-2Stage	-	0/8/0	0/8/0	0/8/0
	GA-RE	0/8/0	-	0/8/0	0/8/0
	FL	0/8/0	0/8/0	-	0/8/0
	GA-MC	0/8/0	0/8/0	0/8/0	-
WSC09 (5 tasks)	GA-2Stage	-	0/5/0	0/5/0	0/5/0
	GA-RE	0/5/0	-	0/5/0	0/5/0
	FL	0/5/0	0/5/0	-	0/5/0
	GA-MC	0/5/0	0/5/0	0/5/0	-

Comparison of the accuracy in robustness estimation

To study the accuracy of the lower bound robust estimation (called r_{LB}) in Eq. (5.4) and the Monte Carlo estimation (called r_{MC}) in Eq. (5.3), we compare each of them with the testing results, which serve as the "ground truth" of the robustness of any composite service in this study. Particularly, we are interested in how accurate different estimation methods are capable of ranking multiple composite services. For this purpose, we firstly record the fitness values of 30 composite services measured by the lower bound robust estimation, Monte Carlo estimation, and the "ground truth", for WSC09-1. Afterwards, we calculate the rank correlation between the r_{LB} and the "ground truth", and between the r_{MC} and the "ground truth", using Pearson, Kendall's tau and Spearman's rho.

Table 5.10: Results of three statistical correlation tests using Pearson, Kendall's tau, and Spearman's rho.

Method		r_{LB}	r_{MC}
Pearson	Correlation coefficient	0.611078	0.426510
	P-value	0.000334	0.018756
Kendall's tau	Correlation coefficient	0.452137	0.310345
	P-value	0.000468	0.016017
Spearman's rho	Correlation coefficient	0.615333	0.493215
	P-value	0.000296	0.005615

Table 5.10 shows the correlation coefficient values and P-values of Pearson, Kendall's tau and Spearman's rho over two pairs of ranks, i.e., r_{LB} and the "ground truth", and r_{MC} and the "ground truth". We can see that both two correlation tests reject the null hypothesis that the two ranks are uncorrelated because all the P-values are less than 0.05. In addition, we can observe that the correlation coefficient between r_{LB} and the "ground truth" is consistently higher than that between r_{MC} and the "ground truth". This indicates that lower bound robust estimation is more

accurate than the Monte Carlo robust estimation.

5.7 Summary

The overall goal of this chapter is to propose GA-based approaches to effectively and efficiently handle the RWSC-SF problem. We have achieved this goal by completing the following: (1) We propose to deal this problems via two stages, design stage and execution stage. At the design stage, robust composite services should be built to serve as the blueprint/baseline. These baseline composite services can cope with unexpected interruptions robustly via a repairing process (i.e., local search) while maintaining high QoS and QoSM at the time of execution. (2) A new GA-MC algorithm is developed to search for such robust composite services using a fitness function, which estimates the robustness of candidate composite services based on Monte Carlo sampling. (3) We studied the reliability of GA-MC in regard to the impact of sample size N, and conduct a study on the small-scale RWSC-SF problem. We compare the effectiveness of GA-MC with FL, which only focuses on searching high-quality solutions. (4) Different from the Monte Carlo sampling-based robustness estimation, we proposed a more accurate robustness estimation method based on a lower bound of the expected fitness. (5) We developed GA-2Stage to effectively and efficiently produce baseline solutions with high robustness. In stage one of GA-2Stage, without considering any service failure, a cheap robustness estimation with a high level of noise (i.e., comprehensive quality) is performed in GA to find good solutions that are likely to be robust. In stage two, these solutions can be further evolved to improve their robustness using the expensive but accurate robustness estimation based on the lower bound of the expected quality. (6) We conduct another study on the large-scale RWSC-SF problem and test the performances of GA-2Stage on this problem using additional large benchmark datasets, WSC-08, and WSC-09. In addition, GA-MC, GA-RE (i.e., a GA

method that utilize the lower bound robustness method throughout the generations), and FL are compared against GA-2Stage.

The development of new algorithms and their experimental study leads to several major findings: (1) For the Monte Carlo sampling-based estimation, we learned that the sampling size has a big impact on the accuracy of Monte Carlo sampling-based robustness estimation. The larger the sampling size is, the more accurate the fitness evaluations are and the higher robustness of the evolved composite services are. However, we also learned that Monte Carlo sampling becomes computationally expensive for approximating the robustness of solutions for large-scale RWSC-SF problem with large service repositories. Therefore, Monte Carlo sampling is not suitable for service composition problems with large repositories. (2) For the lower bound robust estimation, we learned that this estimation could successfully reduce the variance of the robustness estimation. We also found out that lower bound robust estimation is more accurate than the Monte Carlo sampling for ranking different composite services. On the other hand, the Monte Carlo sampling can be more efficient and accurate enough for the small RWSC-SF problem by utilizing a small but suitable sampling size N. (3) We learned that the success of our GA method in finding composite services relies on how accurate the fitness functions measure the robustness of composite services. Particularly, we found that GA-RE with the lower bound robust estimation is more effective and efficient than GA-MC Monte Carlo sampling-based estimation for finding solutions with high robustness. (4) Compared to GA-RE, GA-2Stage could achieve much higher efficiency at the design stage with a negligible impact on the effectiveness, with the help of both the adaptive evolutionary control over two consecutive evolutionary stages and two different involved robustness estimation methods.

Chapter 6

Conclusions

The overall goal of this thesis is to propose fully automated web service composition approaches that can produce functionally valid composite services with optimized QoSM and QoS. This goal has been achieved successfully with the development of innovative algorithms that combine AI planning and EC techniques to find such composite services in the context of single-objective, multi-objective and dynamic problems. Notably, to effectively and explicitly extract, and utilize the knowledge about promising solutions, we start by proposing three single-objective EDA-based approaches (including the use of local search) that sample promising solutions from different distribution models, supporting direct and indirect representations. Subsequently, we propose fully automated service composition approaches to handle two categories of MOCQP: WSC-MO and WSC-MQP. WSC-MO aims to simultaneously produce a set of approximated Pareto-optimal composite services in consideration of both QoSM and QoS. In particular, MNSGA2-EDA is proposed based on NSGA-II with a model-guided local search to effectively and efficiently find high-quality solutions for WSC-MO. WSC-MQP aims to concurrently produce a set of optimized composite services with distinctive preferences on QoSM. In particular, two multi-factorial evolutionary algorithms, PMFEA and PMFEA-EDA, have been proposed to effectively and efficiently

solve WSC-MQP based on implicit and explicit knowledge learning and sharing, respectively. Finally, a new RWSC-SF problem with separate design and execution stages has been proposed to handle stochastic service failures. To handle this dynamic problem, two EC-based approaches have been developed with the goal of building robust composite services, which can effectively handle service failures at the execution stage.

The remainder of this chapter is organized as follows: Sect. 6.1 outlines the objectives that are achieved in this thesis. Sect. 6.2 presents the main conclusions made in this work. Sect. 6.4 explores possible future research directions based on the contributions in this thesis.

6.1 Achieved Objectives

The achieved objectives of the thesis are listed below:

1. This thesis proposed two EDA-based approaches (i.e., EDA-NHM and EDA-EHM) for single-objective automated Web service composition. To effectively optimize the comprehensive quality, EDA-NHM and EDA-EHM rely on two different distribution models (i.e., NHM and EHM) and sampling techniques. Furthermore, memetic EDA-based approaches based on EDA-NHM was proposed to enable the use of several local search operators (see Chapter 3). For EDA-NHM, we introduced a novel permutation-based representation, allowing NHM to be learned from varied structures of candidate composite services. For EDA-NHM, EDA-EHM employs a graph-based representation that presents a set of service dependencies. Such dependencies can be efficiently queried with the help of our ontology-based querying technique. The novelty of EDA-NHM lies in employing EHM to capture the distributions of the service dependencies for building DAG-based composite services, and proposing the GEHBGS technique to efficiently sample promis-

ing and functionally valid DAG-based composite services. Experiments showed that EDA-NHM can outperform PSO, FL, PathSearch and our proposed EDA-EHM in finding composite services with much higher comprehensive quality. In contrast, EDA-EHN can achieve the highest efficiency among all the competing EC-based approaches, while maintaining reasonable effectiveness. Since the effectiveness is our focus, we further proposed memetic EDA-based approaches based on EDA-NHM by introducing an efficient and effective local search procedure. This local search procedure combines a uniform distribution schema with several local search operators. The schema is used to select a small number of suitable individuals to be considered for local search, and the local search operators are used to exploit the neighbouring solutions effectively. One proposed memetic EDA with the layer-based constrained one point swap (i.e., MEEDA-LOP) achieves significantly better effectiveness and efficiency, compared to MEFL, other our proposed memetic EDA-based approaches, and the baseline EDA-NHM. This indicates that the layer-based constrained one point swap is more effective than all the other local search operators utilized in the competing methods.

2. This thesis proposed EC-based approaches to solve two categories of MOCQP: WSC-MO and WSC-MQP (see Chapter 4). WSC-MO aims to identify a set of approximated Pareto-optimal solutions by simultaneously considering both QoSM and QoS. To effectively and efficiently handle this problem, we proposed a memetic service composition approach (i.e., MNSGA2-EDA) that employs EDA to make local improvements on the composite services found by NSGA-II. The novelty of this method lies in the innovative use of EDA for effective and efficient local improvements, rather than for global exploration. This local search is performed by sampling candidate composite services via multiple NHMs, which are constructed separately and concurrently in different regions of the Pareto front along

with some good candidate composite services. The selected Pareto solutions and the regions are identified through the use of a clustering technique. Our experiment showed that MNSGA2-EDA is much more effective and efficient in finding Pareto optimal solutions, compared to Hybrid and Hybrid-L. WSC-MQP focuses on finding a set of solutions by optimizing both QoSM and QoS, subject to different QoSM preferences from multiple user segments. In this thesis, we formulated the WSC-MQP problem as a multitasking problem and proposed two multi-factorial evolutionary algorithms, i.e., PMFEA and PMFEA-EDA, based on implicit and explicit knowledge learning and sharing (see Chapter 4). PMFEA is proposed based on the standard MFEA, but with a permutation-based representation and corresponding genetic operators. Two variations of PMFEA, i.e., PMFEA-NT and PMFEA-AT, have been proposed by performing additional evaluations based on neighbouring tasks and all tasks, respectively. Our experiment showed that all PMFEAs can be performed at the cost of only a fraction of time compared to one single-tasking EC-based method. Besides that, the proper use of the neighbourhood structure over multiple tasks in PMFEA-NT can enhance the effectiveness of PMFEA. Furthermore, PMFEA-EDA is proposed with single-tasking and multitasking NHMs, which are constructed to learn explicit knowledge of promising solutions for each task and every two adjacent tasks, respectively. Meanwhile, a new sampling mechanism, motivated by assortative mating, is proposed to balance the exploration and exploitation of the evolutionary search process for multiple tasks. Our experiment showed that the explicit learning mechanism in PMFEA-EDA can perform knowledge learning and sharing better, compared to PMFEA.

3. This thesis proposed two EC-based approaches (GA-MC and GA-2Stage) for the small-scale and large-scale RWSC-SF problems, respectively (see Chapter 5). To the best of our knowledge, we are

the first to formulate such a dynamic problem as a two-stage process for robust composite service design and execution. The design stage aims to build robust composite services that serve as the blueprint/baseline. These baseline composite services can cope with unexpected interruptions robustly via a repairing process (i.e., local search) for maintaining high quality at the execution stage. Two approaches, GA-MC and GA-2Stage, have been proposed for finding robust composite services at the design stage. In addition, a baseline GA-based method, called GA-RE, was proposed for the purpose of comparison. Particularly, the contribution of GA-MC and GA-RE lies in developing reliable fitness functions that can accurately estimate the robustness of evolved composite services throughout all the generations. Particularly, two robustness estimation methods have been proposed by using Monte Carlo sampling and a lower bound of the expected fitness, respectively. Our experiment showed that GA-RE can outperform GA-MC in finding composite services with high robustness, for both the small-scale and large-scale RWSC-SF problems, while GA-MC can perform very efficiently over service requests for the small-scale RWSC-SF problem. We observed that the Monte Carlo sampling-based method could present high variances for the robustness estimation when the size of the service repository becomes large. However, the lower bound robust estimation can successfully reduce such variances. Apart from GA-MC and GA-RE, GA-2Stage is proposed to achieve high algorithm efficiency with a negligible impact on the effectiveness, regardless of the service repository size. The contribution of GA-2Stage lies in proposing an adaptive evolutionary control mechanism to support two consecutive evolutionary stages by using two different fitness evaluation methods. Stage one tries to efficiently find good composite services that are likely to have high robustness based on a cheap evaluation method with a high level of noise, i.e., comprehensive quality, as-

suming no services fail. Particularly, this evaluation method can produce composite services with a small number of component services, which are less likely to be affected in the events of service failures. The solutions found in the first evolutionary stage are utilized to initialize the population in the second stage that uses the lower bound estimation. Our experiment results further confirmed that GA-2Stage is very comparable to GA-RE in terms of the effectiveness, and it also achieves much higher efficiency compared to GA-RE via the use of evolutionary control.

6.2 Main Conclusions

This section outlines the main conclusions of this thesis.

6.2.1 Explicit Distribution Models for Fully Automated Service Composition

This thesis proposed two novel ways to explicitly learn the knowledge of promising composite services, which are presented as two distribution models based on NHM and EHM. These two distribution models can be iteratively adjusted with the help of EDA for searching composite services with high comprehensive quality. To effectively learn a distribution model in the form of NHM from multiple composite services with varied workflow structures, we proposed a new permutation-based representation that is encoded from a DAG-based representation. Conversely, a DAG-based representation is decoded from randomly initialized or sampled permutations from the NHM. Distribution models based on EHM can naturally capture service dependencies of promising DAGs. In addition, with the help of EHM, we can build up a composition graph from the service dependencies. To prevent cycles in DAGs, general knowledge about service workflows has been utilized to guide the sampling process.

For example, only row indexes of non-zero entries in EHM are to be sampled, and layer information is used to verify sampled predecessors. EDA-EHM is found to be very efficient with competitive effectiveness in finding high-quality composite services, while the EDA-NHM is less efficient, but is found to yield the best quality composite services.

6.2.2 Neighbourhood Structure of Composite services

This thesis investigates appropriate structures of the neighbourhood for composite services to be explored by the newly developed local search operators. Particularly, our local search operators extend "swap" operators on permutation-based representations. In this thesis, we investigate four different problem-specific swap operators. Our local search operators include constrained one-point swap, constrained two-point swap, constrained one-block swap, and layer-based constrained one-point swap. Out of all of them, the layer-based constrained one-point swap is identified as the most effective operator to enhance the exploitation ability of EDA via local search. The layer-based constrained one-point swap operator is proposed by extending the constrained one-point swap with the layer information. The advantage of this swap is due to two reasons: two points (i.e., position indexes) to be swapped must be before | and after | respectively, resulting in a neighbouring permutation that has a high chance to use the services after | to build a different DAG. The layer information can effectively select more suitable services after |, increasing the chance for building a different DAG significantly. In other words, given the same computation budget on local search, the layer-based constrained one-point swap can potentially exploit more different DAGs, and is more likely to reach a better neighbouring solution.

6.2.3 Local Improvements on Pareto Solutions Using EDA

This thesis proposed MNSGA2-EDA with EDA-based local search, searching for Pareto optimal solutions with respect to QoSM and QoS. The contribution of this method is in the novel way of using EDA to perform a local search, rather than for global exploration. This local search is performed separately and concurrently in different regions of the Pareto front. Such regions are identified by a clustering technique to select multiple candidate Pareto solutions for local improvements and other good candidate solutions in each region for building NHMs. To make local improvements on the selected solutions, we propose an effective method to learn multiple NHMs. Particularly, we use the Euclidean distances in the objective space between the candidate Pareto solution and other members in this region to weight the influences of every chosen solution on NHM. This is because members far from the candidate Pareto solution should contribute less to the distribution model that we aim to learn. Our proposed MNSGA2-EDA can outperform Hybrid, Hybrid-L and the baseline NSGA-II in finding better Pareto composite services, without sacrificing the efficiency.

6.2.4 Using EDA to Achieve Effective and Efficient WSC-MQP

Solving one composition request at a time poses difficulty in meeting the efficiency target due to a large quantity of requests generated by a growing number of users [17]. In our thesis, multitasking service composition approaches have been proposed to deal with multiple composition requests concurrently in an efficient way. To solve multiple service requests with respect to several pre-determined user segments simultaneously, this thesis proposed PMFEA-EDA that explicitly learns and shares knowledge across composite services for several different service requests. The contribution of this method lies in iteratively building a set of single-tasking and multi-tasking NHMs for the purpose of effective knowledge learning and shar-

ing. Particularly, each single-tasking NHM can capture the knowledge of good solutions with respect to one task. At the same time, multitasking NHM can facilitate knowledge sharing across pairs of neighbouring tasks. Our experiment shows that PMFEA-EDA takes much less execution time than existing single-tasking service composition approaches that process each service request separately. Furthermore, PMFEA-EDA also produces composite services with much higher quality, compared to PMFEA that implicitly learns and shares knowledge across composite services based on assortative mating and the existing single-tasking service composition approaches, such as FL and our proposed EDA-NHM. The core observation is that explicitly learning and sharing knowledge is more effective than implicit learning and sharing through the use of MFEA.

6.2.5 The Application of Two-stage Robust Service Composition

This thesis proposed a new dynamic service composition problem with a key focus on stochastic service failures. This problem involves a two-stage robust service composition process, consisting of a design stage and an execution stage. The novelty of this problem lies in that the design stage constructs baseline composite services with high robustness (in terms of expected QoS and QoSM), which are likely to continue to work reliably or be efficiently re-optimized with negligible impact on the quality at the execution stage. To effectively and efficiently generate such baseline solutions, we employ EC techniques to evolve composite services with optimized robustness. One key concern is to create an accurate robustness estimation method. We have proposed two robustness estimation methods: one based on Monte Carlo sampling and the other based on a lower bound of the expected robustness. The lower bound estimation method is found to yield the highest accuracy in ranking candidate composite services regardless of the size of the service repository. Furthermore, com-

pared to GA-MC and GA-RE, our proposed GA-2Stage turns out be much more efficient at the design stage with a negligible impact on the quality at the execution stage. This is realized by using a new adaptive evolutionary control mechanism.

6.3 Practical Guidelines

In practise, no single algorithm can fit all customers' requirements. Before choosing a service composition algorithm, practitioners need to have a clear picture of their customers' needs. In Fig. 6.1, we use a decision tree to guide practitioners to choose one of our proposed algorithms based on the characteristics of users' requirements. These characteristics include: (1) quality preferences (i.e., weights) on each quality criteria that are involved into the fitness function; (2) whether service failures can be recovered in a timely manner; (3) algorithm performances in terms of effectiveness and efficiency; (4) scales of the service composition problem; (5) multiple constraints (e.g., multiple user segment preferences on QoSM).

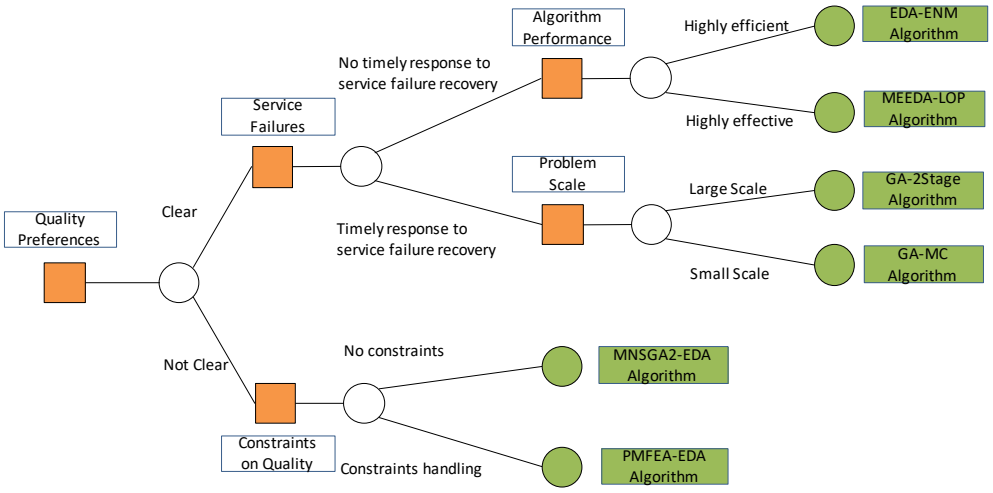


Fig. 6.1: A decision tree to guide practitioners for choosing algorithms

6.4 Future Work

This thesis reveals several additional avenues to be investigated further for automated Web service composition. In this section, we will discuss some potential future research directions.

6.4.1 Miscellaneous Distribution Models and Sampling Techniques

The success of EDA strongly relies on suitable distribution models that are used for learning the explicit knowledge of promising candidate composite services. The NHM and EHM utilized in EDA-NHM and EDA-EHM can effectively capture the distributions of component service positions or service dependencies, respectively. In addition to the two distribution models, we can investigate other suitable generative models for effectively learning the knowledge of composite services, such as variation auto-encoder. Such models may be more effective at extracting complex high-level knowledge from evolved composite services, potentially helping to enhance the performance of EDA. These models are out of the scope of this thesis because we focus on univariate models. Apart from the distribution models, we have not investigated the impacts of sampling templates. For example, a permutation template fixes the elements for a part of the positions on new permutations to sampled. The proper use of a template can guide the search effectively via sampling. Therefore, an effective strategy for selecting suitable templates can be crucial for effective sampling. This strategy is related to EDA, but it is not in the scope of our thesis.

6.4.2 Miscellaneous Decoding Strategy for Permutations

Permutation-based representations are used in our EDA-based approaches, such as EDA-NHM and PMFEA-EDA. Such permutation-based represen-

tations always require a decoding strategy to create DAG-based composite services. This decoding must be performed on all the permutation-based solutions in order to evaluate their QoS and QoSM. Therefore, the decoding strategy has a high impact on the overall execution time of EC-based algorithms. In our thesis, a forward-decoding strategy is often used to build up a DAG, starting from the $Start$ node, based on a permutation. Particularly, a DAG is gradually built up by adding service indexes in the permutation from left to right, following a sequential order. Note that this decoding process can be more computationally expensive when the length of ordered service indexes is very large. Therefore, future work in this area could investigate alternative decoding strategies. For example, instead of following sequentially ordered services for decoding, one possible improvement could be to cache all the service dependencies and design a decoding strategy guided by the service dependencies.

6.4.3 Many-Objective Optimisation

This thesis studies multi-objective semantic web service composition with the aim to optimize two objectives, i.e., QoSM and QoS. However, each objective still combines more than one quality criteria in QoSM or QoS. These quality criteria might be conflicting with each other. For example, cost and response time in QoS. Future work could investigate many-objective optimization techniques and optimize each individual criterion in QoSM and QoS rather than a combined criterion. Simultaneously optimizing many (more than three) quality criteria is a challenging task, requiring further studies on existing many-objective optimization techniques to cope with this goal. For example, when more objectives are considered, selecting appropriate individuals for the next generation that can help the population toward the Pareto optimal set is very difficult [115]. Despite some recent success in tackling this issue in EC-based algorithms, we cannot simply use these algorithms because proper modifications are needed to

cope better with our problem.

6.4.4 Robustness estimation

This thesis proposes two different methods to approximate the robustness of composite services. One is based on Monte Carlo sampling, and the other one is based on a lower bound of the expected fitness. The lower bound estimation has been empirically shown to be more accurate than the Monte Carlo technique when ranking composite services according to their robustness during the evolutionary process. However, the lower bound estimation introduces bias on the robustness estimation because the lower bound is computed based on a manually-created rule for selecting scenarios. In the future, instead of manually selecting scenarios, we can automatically learn a rule to guide scenario selection. For example, GP has been widely used to generate scheduling policies in job shop scheduling problems. Motivated by the successful use of GP in this field, we could use GP to evolve problem specific rules. We expect such rules can be effective for selecting suitable scenarios, contributing to more accurate and efficient estimation of the robustness.

Bibliography

[1] ACID, S., AND DE CAMPOS, L. M. Searching for bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research* (2003), 445–490.

[2] AGARWAL, S., JUNGHANS, M., FABRE, O., TOMA, I., AND LORRE, J.-P. D5. 3.1 first service discovery prototype. *Deliverable D5 3* (2009).

[3] AGARWAL, S., LAMPARTER, S., AND STUDER, R. Making web services tradable: A policy-based approach for specifying preferences on web service properties. *Web Semantics: Science, Services and Agents on the World Wide Web 7*, 1 (2009), 11–20.

[4] AL-MASRI, E., AND MAHMOUD, Q. H. Qos-based discovery and ranking of web services. In *International Conference on Computer Comm. Networks* (2007), IEEE, pp. 529–534.

[5] AL-MASRI, E., AND MAHMOUD, Q. H. Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 795–804.

[6] AMIN, A., COLMAN, A., AND GRUNSKE, L. An approach to forecasting qos attributes of web services based on arima and garch models. In *IEEE International Conference on Web Services* (2012), IEEE, pp. 74–81.

[7] AMIRI, M. A., AND SERAJZADEH, H. Effective web service composition using particle swarm optimization algorithm. In *6th International Symposium on Telecommunications (IST)* (2012), IEEE, pp. 1190–1194.

[8] ANDREWS, T., CURBERA, F., DHOLAKIA, H., GOLAND, Y., KLEIN, J., LEYMANN, F., LIU, K., ROLLER, D., SMITH, D., THATTE, S., ET AL. Business process execution language for web services, 2003.

[9] ARDAGNA, D., AND PERNICI, B. Adaptive service composition in flexible processes. *IEEE Trans. Software Engineering 33*, 6 (2007), 369–384.

[10] ARTHUR, D., AND VASSILVITSKII, S. K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 1027–1035.

[11] ARUNACHALAM, N., AND AMUTHAN, A. Integrated probability multi-search and solution acceptance rule-based artificial bee colony optimization scheme for web service composition. *Natural Computing* (2019), 1–16.

[12] AVERSANO, L., DI PENTA, M., AND TANEJA, K. A genetic programming approach to support the design of service compositions. *International Journal of Computer Systems Science & Engineering 21*, 4 (2006), 247–254.

[13] BACK, T., HAMMEL, U., AND SCHWEFEL, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation 1*, 1 (1997), 3–17.

[14] BALUJA, S. Population-based incremental learning. a method for integrating genetic search based function optimization and compet-

itive learning. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

[15] BANSAL, A., BLAKE, M. B., KONA, S., BLEUL, S., WEISE, T., AND JAEGER, M. C. Wsc-08: continuing the web services challenge. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on* (2008), IEEE, pp. 351–354.

[16] BANSAL, S., BANSAL, A., GUPTA, G., AND BLAKE, M. B. Generalized semantic web service composition. *Service Oriented Computing and Applications 10*, 2 (2016), 111–133.

[17] BAO, L., QI, Y., SHEN, M., BU, X., YU, J., LI, Q., AND CHEN, P. An evolutionary multitasking algorithm for cloud computing service composition. In *World Congress on Services* (2018), Springer, pp. 130–144.

[18] BARESI, L., AND GUINEA, S. Self-supervising bpel processes. *IEEE Transactions on Software Engineering 37*, 2 (2011), 247–263.

[19] BERETA, M. Baldwin effect and lamarckian evolution in a memetic algorithm for euclidean steiner tree problem. *Memetic Computing 11*, 1 (2019), 35–52.

[20] BLUM, A. L., AND FURST, M. L. Fast planning through planning graph analysis. *Artificial intelligence 90*, 1 (1997), 281–300.

[21] BOLBOACA, S.-D., AND JÄNTSCHI, L. Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences 5*, 9 (2006), 179–200.

[22] BOOTH, D., HAAS, H., MCCABE, F., NEWCOMER, E., CHAMPION, M., FERRIS, C., AND ORCHARD, D. Web services architecture. w3c working note. *W3C Working Notes* (2004).

[23] BOUDRIES, F., SADOUKI, S., AND TARI, A. A bio-inspired algorithm for dynamic reconfiguration with end-to-end constraints in web services composition. *Service Oriented Computing and Applications 13*, 3 (2019), 251–260.

[24] BOUSSALIA, S. R., AND CHAOUI, A. Optimizing qos-based web services composition by using quantum inspired cuckoo search algorithm. In *International Conference on Mobile Web and Information Systems* (2014), Springer, pp. 41–55.

[25] BOUSTIL, A., MAAMRI, R., AND SAHNOUN, Z. A semantic selection approach for composite web services using owl-dl and rules. *Service Oriented Computing and Applications 8*, 3 (2014), 221–238.

[26] CANFORA, G., DI PENTA, M., ESPOSITO, R., AND VILLANI, M. L. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (2005), ACM, pp. 1069–1075.

[27] CANFORA, G., DI PENTA, M., ESPOSITO, R., AND VILLANI, M. L. Qos-aware replanning of composite web services. In *IEEE International Conference on Web Services (ICWS'05)* (2005), IEEE, pp. 121–129.

[28] CAO, L., LI, M., AND CAO, J. Using genetic algorithm to implement cost-driven web service selection. *Multiagent and Grid Systems 3*, 1 (2007), 9–17.

[29] CEBERIO, J., IRUROZKI, E., MENDIBURU, A., AND LOZANO, J. A. A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progr. Artificial Intellig. 1*, 1 (2012), 103–117.

[30] CHATTOPADHYAY, S., AND BANERJEE, A. Qscas: Qos aware web service composition algorithms with stochastic parameters. In *2016*

IEEE International Conference on Web Services (ICWS) (2016), IEEE, pp. 388–395.

[31] CHATTOPADHYAY, S., BANERJEE, A., AND BANERJEE, N. A scalable and approximate mechanism for web service composition. In *2015 IEEE International Conference on Web Services* (2015), IEEE, pp. 9–16.

[32] CHATTOPADHYAY, S., BANERJEE, A., AND BANERJEE, N. A fast and scalable mechanism for web service composition. *ACM Transactions on the Web (TWEB) 11*, 4 (2017), 26.

[33] CHEN, M., AND YAN, Y. Qos-aware service composition over graphplan through graph reachability. In *Services Computing (SCC), 2014 IEEE International Conference on* (2014), IEEE, pp. 544–551.

[34] CHEN, X., ONG, Y.-S., LIM, M.-H., AND TAN, K. C. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation 15*, 5 (2011), 591–607.

[35] CHEN, Y., HUANG, J., AND LIN, C. Partial selection: An efficient approach for qos-aware web service composition. In *IEEE International Conference on Web Services* (2014), IEEE, pp. 1–8.

[36] CHIFU, V. R., POP, C. B., SALOMIE, I., SUIA, D. S., AND NICULICI, A. N. Optimizing the semantic web service composition process using cuckoo search. In *Intelligent distributed computing V*. Springer, 2011, pp. 93–102.

[37] CHIFU, V. R., SALOMIE, I., POP, C. B., NICULICI, A. N., AND SUIA, D. S. Exploring the selection of the optimal web service composition through ant colony optimization. *Computing and Informatics 33*, 5 (2015), 1047–1064.

[38] CURBERA, F., DUFTLER, M., KHALAF, R., NAGY, W., MUKHI, N., AND WEERAWARANA, S. Unraveling the web services web: an in-

troduction to soap, wsdl, and uddi. *IEEE Internet computing 6*, 2 (2002), 86–93.

[39] CURBERA, F., NAGY, W., AND WEERAWARANA, S. Web services: Why and how. In *Workshop on Object-Oriented Web Services-OOPSLA* (2001).

[40] DA SILVA, A. S., MA, H., MEI, Y., AND ZHANG, M. A hybrid memetic approach for fully automated multi-objective web service composition. In *2018 IEEE International Conference on Web Services* (2018), IEEE, pp. 26–33.

[41] DA SILVA, A. S., MA, H., AND ZHANG, M. Graphevol: a graph evolution technique for web service composition. In *DEXA* (2015), Springer, pp. 134–142.

[42] DA SILVA, A. S., MA, H., AND ZHANG, M. Genetic programming for QoS-aware web service composition and selection. *Soft Computing* (2016), 1–17.

[43] DA SILVA, A. S., MA, H., ZHANG, M., AND HARTMANN, S. Handling branched web service composition with a qos-aware graph-based method. In *International Conference on Electronic Commerce and Web Technologies* (2016), Springer, pp. 154–169.

[44] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. A memetic algorithm-based indirect approach to web service composition. In *Evolutionary Computation (CEC), IEEE Congress on* (2016).

[45] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. Particle swarm optimisation with sequence-like indirect representation for web service composition. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (2016), Springer, pp. 202–218.

[46] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. Fragment-based genetic programming for fully automated multi-objective web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), ACM, pp. 353–360.

[47] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. Evolutionary computation for automatic web service composition: an indirect representation approach. *J. Heuristics* (2018), 425–456.

[48] DANIEL, A. M., AND MENASC, T. Qos issues in web services. *IEEE Internet Computing 6*, 6 (2002), 72–75.

[49] DAVIS, L. Applying adaptive algorithms to epistatic domains. In *IJCAI* (1985), vol. 85, pp. 162–164.

[50] DE BONET, J. S., ISBELL JR, C. L., AND VIOLA, P. A. Mimic: Finding optima by estimating probability densities. In *Advances in neural information processing systems* (1997), pp. 424–430.

[51] DE CAMPOS, A., POZO, A. T., VERGILIO, S. R., AND SAVEGNAGO, T. Many-objective evolutionary algorithms in the composition of web services. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on* (2010), IEEE, pp. 152–157.

[52] DE JONG, K. Evolutionary computation: a unified approach. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (2016), pp. 185–199.

[53] DEB, K., AND JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation 18*, 4 (2013), 577–601.

[54] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[55] DU, W., ZHONG, W., TANG, Y., DU, W., AND JIN, Y. High-dimensional robust multi-objective optimization for order scheduling: A decision variable classification approach. *IEEE Transactions on Industrial Informatics 15*, 1 (2018), 293–304.

[56] ERL, T. *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice hall, 2004.

[57] FAN, S.-L., YANG, Y.-B., AND WANG, X.-X. Efficient web service composition via knapsack-variant algorithm. In *International Conference on Services Computing* (2018), Springer, pp. 51–66.

[58] FANJIANG, Y.-Y., AND SYU, Y. Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach. *Information and Software Technology 56*, 3 (2014), 352–373.

[59] FENG, L., ONG, Y.-S., TAN, A.-H., AND TSANG, I. W. Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems. *Memetic Computing 7*, 3 (2015), 159–180.

[60] FENG, Y., NGAN, L. D., AND KANAGASABAI, R. Dynamic service composition with service-dependent QoS attributes. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 10–17.

[61] FENSEL, D., FACCA, F. M., SIMPERL, E., AND TOMA, I. *Semantic web services*. Springer Science & Business Media, 2011.

[62] GABREL, V., MANOUVRIER, M., MEGDICHE, I., AND MURAT, C. A new 0–1 linear program for qos and transactional-aware web service composition. In *2012 IEEE Symposium on Computers and Communications (ISCC)* (2012), IEEE, pp. 000845–000850.

[63] GAO, A., YANG, D., TANG, S., AND ZHANG, M. Web service composition using integer programming-based models. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on* (2005), IEEE, pp. 603–606.

[64] GAO, C., CAI, M., AND CHEN, H. Qos-aware service composition based on tree-coded genetic algorithm. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)* (2007), vol. 1, IEEE, pp. 361–367.

[65] GEYIK, S. C., SZYMANSKI, B. K., AND ZERFOS, P. Robust dynamic service composition in sensor networks. *IEEE Trans. Services Computing* (2013), 560–572.

[66] GHOBAEI-ARANI, M., RAHMANIAN, A. A., ASLANPOUR, M. S., AND DASHTI, S. E. CSA-WSC: cuckoo search algorithm for web service composition in cloud environments. *Soft Computing 22*, 24 (2018), 8353–8378.

[67] GHOBAEI-ARANI, M., AND SOURI, A. LP-WSC: a linear programming approach for web service composition in geographically distributed cloud environments. *The Journal of Supercomputing 75*, 5 (2019), 2603–2628.

[68] GILCHRIST, W. *Statistical modelling with quantile functions*. Chapman and Hall/CRC, 2000.

[69] GRÄNING, L., JIN, Y., AND SENDHOFF, B. Individual-based management of meta-models for evolutionary optimization with application to three-dimensional blade optimization. In *Evolutionary computation in dynamic and uncertain environments*. Springer, 2007, pp. 225–250.

[70] GROSAN, C., AND ABRAHAM, A. Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In *Hybrid evolutionary algorithms*. Springer, 2007, pp. 1–17.

[71] GUPTA, A., ONG, Y.-S., AND FENG, L. Multifactorial evolution: toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation 20*, 3 (2016), 343–357.

[72] GUPTA, I. K., KUMAR, J., AND RAI, P. Optimization to quality-of-service-driven web service composition using modified genetic algorithm. In *Computer, Communication and Control (IC4), 2015 International Conference on* (2015), IEEE, pp. 1–6.

[73] HART, W. E., KRASNOGOR, N., AND SMITH, J. E. Memetic evolutionary algorithms. In *Recent advances in memetic algorithms*. Springer, 2005, pp. 3–27.

[74] HAUSCHILD, M., AND PELIKAN, M. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation 1*, 3 (2011), 111–128.

[75] HENNIG, P., AND BALKE, W.-T. Highly scalable web service composition using binary tree-based parallelization. In *2010 IEEE International Conference on Web Services* (2010), IEEE, pp. 123–130.

[76] HOSSAIN, M. S., MONIRUZZAMAN, M., MUHAMMAD, G., GHONEIM, A., AND ALAMRI, A. Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. *IEEE Transactions on Services Computing 9*, 5 (2016), 806–817.

[77] HUANG, Z., JIANG, W., HU, S., AND LIU, Z. Effective pruning algorithm for QoS-aware service composition. In *2009 IEEE Conference on Commerce and Enterprise Computing* (2009), IEEE, pp. 519–522.

[78] HUY, N. Q., SOON, O. Y., HIOT, L. M., AND KRASNOGOR, N. Adaptive cellular memetic algorithms. *Evolutionary Computation* (2009), 231–256.

[79] HWANG, C.-L., AND YOON, K. Lecture notes in economics and mathematical systems. *Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey* (1981).

[80] HWANG, S.-Y., HSU, C.-C., AND LEE, C.-H. Service selection for web services with probabilistic qos. *IEEE transactions on services computing 8*, 3 (2014), 467–480.

[81] HWANG, S.-Y., LIM, E.-P., LEE, C.-H., AND CHEN, C.-H. Dynamic web service selection for reliable web service composition. *IEEE Transactions on Services Computing 1*, 2 (2008), 104–116.

[82] JATOTH, C., AND GANGADHARAN, G. Qos-aware web service composition using quantum inspired particle swarm optimization. In *International Conference on Intelligent Decision Technologies* (2017), Springer, pp. 255–265.

[83] JIAN, X., ZHU, Q., AND XIA, Y. An interval-based fuzzy ranking approach for qos uncertainty-aware service composition. *Optik-International Journal for Light and Electron Optics 127*, 4 (2016), 2102–2110.

[84] JIANG, S., ONG, Y.-S., ZHANG, J., AND FENG, L. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Trans. Cybernetics 44*, 12 (2014), 2391–2404.

[85] JIANG, W., ZHANG, C., HUANG, Z., CHEN, M., HU, S., AND LIU, Z. Qsynth: A tool for qos-aware automatic service composition. In *2010 IEEE International Conference on Web Services* (2010), IEEE, pp. 42–49.

[86] JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation 1*, 2 (2011), 61–70.

[87] JIN, Y., BRANKE, J., ET AL. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evolutionary Computation 9*, 3 (2005), 303–317.

[88] JIN, Y., HÜSKEN, M., AND SENDHOFF, B. Quality measures for approximate models in evolutionary computation. In *GECCO* (2003), pp. 170–173.

[89] KALEPU, S., KRISHNASWAMY, S., AND LOKE, S. W. Verity: a qos metric for selecting web services and providers. In *Fourth International Conference on Web Information Systems Engineering Workshops, 2003. Proceedings.* (2003), IEEE, pp. 131–139.

[90] KLEIN, A., ISHIKAWA, F., AND HONIDEN, S. Efficient heuristic approach with improved time complexity for qos-aware service composition. In *2011 IEEE International Conference on Web Services* (2011), IEEE, pp. 436–443.

[91] KNOWLES, J., AND CORNE, D. A comparative assessment of memetic, evolutionary, and constructive algorithms for the multiobjective d-mst problem. In *GECCO-2001 workshop program* (2001), pp. 162–167.

[92] KONA, S., BANSAL, A., BLAKE, M. B., BLEUL, S., AND WEISE, T. Wsc-2009: a quality of service-oriented web services challenge. In *2009 IEEE Conference on Commerce and Enterprise Computing* (2009), IEEE, pp. 487–490.

[93] KONING, M., SUN, C.-A., SINNEMA, M., AND AVGERIOU, P. Vxbpel: Supporting variability for web services in bpel. *Information and Software Technology 51*, 2 (2009), 258–269.

[94] KOPECKÝ, J., VITVAR, T., BOURNEZ, C., AND FARRELL, J. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing 11*, 6 (2007).

[95] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[96] KUROPKA, D., TRÖGER, P., STAAB, S., AND WESKE, M. *Semantic service provisioning*. Springer, 2008.

[97] KÜSTER, U., KÖNIG-RIES, B., AND KRUG, A. Opossum-an online portal to collect and share SWS descriptions. In *Semantic Computing, 2008 IEEE International Conference on* (2008), IEEE, pp. 480–481.

[98] LACOMME, P., PRINS, C., AND RAMDANE-CHERIF, W. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research 131*, 1-4 (2004), 159–185.

[99] LAUSEN, H., AND FARRELL, J. Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C* (2007), 749–758.

[100] LAUSEN, H., POLLERES, A., AND ROMAN, D. W3c member submission-web service modeling ontology (wsmo). *W3C. Available at¡ URL: http://www. w3. org/Submission/WSMO* (2005).

[101] LÉCUÉ, F. Optimizing QoS-aware semantic web service composition. In *International Semantic Web Conference* (2009), Springer, pp. 375–391.

[102] LÉCUÉ, F., AND DELTEIL, A. Making the difference in semantic web service composition. In *Proceedings of the National Conference on Artificial Intelligence* (2007), vol. 22, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1383.

[103] LÉCUÉ, F., DELTEIL, A., AND LÉGER, A. Optimizing causal link based web service composition. In *ECAI* (2008), pp. 45–49.

[104] LÉCUÉ, F., AND LÉGER, A. A formal model for semantic web service composition. In *International Semantic Web Conference* (2006), Springer, pp. 385–398.

[105] LEUNG*, S. C., AND WU, Y. A robust optimization model for stochastic aggregate production planning. *Production planning & control 15*, 5 (2004), 502–514.

[106] LI, G., LIAO, L., SONG, D., AND ZHENG, Z. A fault-tolerant framework for qos-aware web service composition via case-based reasoning. *International Journal of Web and Grid Services 10*, 1 (2014), 80–99.

[107] LI, J., YAN, Y., AND LEMIRE, D. Full solution indexing for top-k web service composition. *IEEE Transactions on Services Computing* (2016).

[108] LI, M., HUA, Z., ZHAO, J., ZOU, Y., AND XIE, B. Arima model-based web services trustworthiness evaluation and prediction. In *International Conference on Service-Oriented Computing* (2012), Springer, pp. 648–655.

[109] LI, X., ZHAO, Q., AND DAI, Y. A semantic web service composition method based on an enhanced planning graph. In *2010 International Conference on E-Business and E-Government* (2010), IEEE, pp. 2288–2291.

[110] LIAO, J., LIU, Y., WANG, J., WANG, J., AND QI, Q. Lightweight approach for multi-objective web service composition. *IET Software 10*, 4 (2016), 116–124.

[111] LIN, K.-J., ZHANG, J., ZHAI, Y., AND XU, B. The design and implementation of service process reconfiguration with end-to-end qos constraints in soa. *Service Oriented Computing and Applications 4*, 3 (2010), 157–168.

[112] LIU, J., LI, J., LIU, K., AND WEI, W. A hybrid genetic and particle swarm algorithm for service composition. In *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on* (2007), IEEE, pp. 564–567.

[113] LIU, S., LIU, Y., JING, N., TANG, G., AND TANG, Y. A dynamic web service selection strategy with QoS global optimization based on multi-objective genetic algorithm. In *International Conference on Grid and Cooperative Computing* (2005), Springer, pp. 84–89.

[114] LONG, J., AND GUI, W. An environment-aware particle swarm optimization algorithm for services composition. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on* (2009), IEEE, pp. 1–4.

[115] LÓPEZ JAIMES, A., AND COELLO COELLO, C. A. *Many-Objective Problems: Challenges and Methods*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 1033–1046.

[116] MA, H., SCHEWE, K.-D., THALHEIM, B., AND WANG, Q. A formal model for the interoperability of service clouds. *Service Oriented Computing and Applications 6*, 3 (2012), 189–205.

[117] MA, H., WANG, A., AND ZHANG, M. A hybrid approach using genetic programming and greedy search for QoS-aware web service composition. *Trans. Large-Scale Data Knowledge-Centered Syst. 18* (2015), 180–205.

[118] MARKOU, G., AND REFANIDIS, I. Non-deterministic planning methods for automated web service composition. *Artificial Intelligence Research 5*, 1 (2015), 14.

[119] MARTIN, D., BURSTEIN, M., HOBBS, J., LASSILA, O., MCDERMOTT, D., MCILRAITH, S., NARAYANAN, S., PAOLUCCI, M., PARSIA, B.,

PAYNE, T., ET AL. Owl-s: Semantic markup for web services. *W3C member submission 22* (2004), 2007–04.

[120] MCILRAITH, S. A., SON, T. C., AND ZENG, H. Semantic web services. *IEEE intelligent systems 16*, 2 (2001), 46–53.

[121] MENASCÉ, D. A. QoS issues in web services. *IEEE internet computing 6*, 6 (2002), 72–75.

[122] MIER, P. R., PEDRINACI, C., LAMA, M., AND MUCIENTES, M. An integrated semantic web service discovery and composition framework.

[123] MOGHADDAM, M., AND DAVIS, J. G. Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations*. Springer, 2014, pp. 321–346.

[124] MOHANTY, R., RAVI, V., AND PATRA, M. R. Web-services classification using intelligent techniques. *Expert Systems with Applications 37*, 7 (2010), 5484–5490.

[125] MOSTAFA, A., AND ZHANG, M. Multi-objective service composition in uncertain environments. *IEEE Trans. Services Computing* (2015).

[126] MUCIENTES, M., LAMA, M., AND COUTO, M. I. A genetic programming-based algorithm for composing web services. In *2009 Ninth International Conference on Intelligent Systems Design and Applications* (2009), IEEE, pp. 379–384.

[127] NASERI, A., AND NAVIMIPOUR, N. J. A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing 10*, 5 (2019), 1851–1864.

[128] O'LEARY, D. Review: Ontologies: A silver bullet for knowledge management and electronic commerce. *The Computer Journal 48*, 4 (2005), 498–498.

[129] OVERDICK, H. The resource-oriented architecture. In *Services, 2007 IEEE Congress on* (2007), IEEE, pp. 340–347.

[130] PAGANELLI, F., AMBRA, T., AND PARLANTI, D. A qos-aware service composition approach based on semantic annotations and integer programming. *International Journal of Web Information Systems 8*, 3 (2012), 296–321.

[131] PAIXÃO, T., BADKOBEH, G., BARTON, N., ÇÖRÜŞ, D., DANG, D.-C., FRIEDRICH, T., LEHRE, P. K., SUDHOLT, D., SUTTON, A. M., AND TRUBENOVÁ, B. Toward a unifying framework for evolutionary processes. *Journal of Theoretical Biology 383* (2015), 28–43.

[132] PAOLUCCI, M., KAWAMURA, T., PAYNE, T. R., AND SYCARA, K. Semantic matching of web services capabilities. In *International Semantic Web Conference* (2002), Springer, pp. 333–347.

[133] PAPAZOGLOU, M. P. Service-oriented computing: Concepts, characteristics and directions. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on* (2003), IEEE, pp. 3–12.

[134] PAPAZOGLOU, M. T. P., dustdar, s., leymann, f.. service-oriented computing. research roadmap, 2006.

[135] PAPOULIS, A. Probability, random variables and stochastic processes ', mc. graw-hill book company, new york, usa.

[136] PAREJO, J. A., FERNANDEZ, P., AND CORTÉS, A. R. QoS-aware services composition using tabu search and hybrid genetic algorithms.

Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos 2, 1 (2008), 55–66.

[137] PEER, J. Web service composition as AI planning-a survey. *University of St. Gallen* (2005).

[138] PELIKAN, M., GOLDBERG, D. E., AND TSUTSUI, S. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. *Information Sciences 156*, 3-4 (2003), 147–171.

[139] PELIKAN, M., HAUSCHILD, M. W., AND LOBO, F. G. *Estimation of Distribution Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 899–928.

[140] PENG, S., WANG, H., AND YU, Q. Estimation of distribution with restricted boltzmann machine for adaptive service composition. In *IEEE ICWS* (2017), pp. 114–121.

[141] PETRIE, C. J. *Web Service Composition*. Springer, 2016.

[142] PICHANAHAREE, K., AND SENIVONGSE, T. Qos-based service provision schemes and plan durability in service composition. In *IFIP International Conference on Distributed Applications and Interoperable Systems* (2008), Springer, pp. 58–71.

[143] POP, C. B., CHIFU, V. R., SALOMIE, I., AND DINSOREANU, M. Immune-inspired method for selecting the optimal solution in web service composition. In *International Workshop on Resource Discovery* (2009), Springer, pp. 1–17.

[144] POP, F.-C., PALLEZ, D., CREMENE, M., TETTAMANZI, A., SUCIU, M., AND VAIDA, M. Qos-based service optimization using differential evolution. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), pp. 1891–1898.

[145] QI, L., TANG, Y., DOU, W., AND CHEN, J. Combining local optimization and enumeration for QoS-aware web service composition. In *IEEE ICWS* (2010), pp. 34–41.

[146] RAO, J., DIMITROV, D., HOFMANN, P., AND SADEH, N. A mixed initiative approach to semantic web service discovery and composition: Sap's guided procedures framework. In *2006 IEEE International Conference on Web Services (ICWS'06)* (2006), IEEE, pp. 401–410.

[147] RAO, J., AND SU, X. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition* (2004), Springer, pp. 43–54.

[148] RAO, J., AND SU, X. Semantic web services and web process composition, volume 3387 of lncs, chapter a survey of automated web service composition methods, 2005.

[149] RENDERS, J.-M., AND FLASSE, S. P. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 2 (1996), 243–258.

[150] REZAIE, H., NEMATBAKSH, N., AND MARDUKHI, F. A multi-objective particle swarm optimization for web service composition. In *International Conference on Networked Digital Technologies* (2010), Springer, pp. 112–122.

[151] RODRIGUEZ-MIER, P., MUCIENTES, M., AND LAMA, M. Automatic web service composition with a heuristic-based search algorithm. In *2011 IEEE International Conference on Web Services* (2011), IEEE, pp. 81–88.

[152] RODRIGUEZ-MIER, P., MUCIENTES, M., LAMA, M., AND COUTO, M. I. Composition of web services through genetic programming. *Evolutionary Intelligence 3*, 3-4 (2010), 171–186.

[153] RUBINSTEIN, R. Y., AND KROESE, D. P. *Simulation and the Monte Carlo method*, vol. 10. Wiley, 2016.

[154] RUSSELL, S. S, p. norvig,". *Artificial Intelligence: A Modern Approach." Upper Saddle River, NJ: Prentice Hall/Pearson Education* (2003).

[155] RUSSELL, S. J., AND NORVIG, P. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[156] SADEGHIRAM, S., MA, H., AND CHEN, G. Cluster-guided genetic algorithm for distributed data-intensive web service composition. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), IEEE, pp. 1–7.

[157] SALAS, J., PEREZ-SORROSAL, F., PATIÑO-MARTÍNEZ, M., AND JIMÉNEZ-PERIS, R. Ws-replication: a framework for highly available web services. In *Proceedings of the 15th international conference on World Wide Web* (2006), ACM, pp. 357–366.

[158] SAWCZUK DA SILVA, A. Evolutionary computation for multifaceted web service composition.

[159] SHAN, Y., MCKAY, R. I., ESSAM, D., AND ABBASS, H. A. A survey of probabilistic model building genetic programming. In *Scalable optimization via probabilistic modeling*. Springer, 2006, pp. 121–160.

[160] SHET, K., ACHARYA, U. D., ET AL. A new similarity measure for taxonomy based on edge counting. *arXiv preprint arXiv:1211.4709* (2012).

[161] SIRIN, E., PARSIA, B., WU, D., HENDLER, J., AND NAU, D. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web 1*, 4 (2004), 377–396.

[162] SOHRABI, S., PROKOSHYNA, N., AND MCILRAITH, S. A. Web service composition via the customization of golog programs with user

preferences. In *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 319–334.

[163] SRINIVAS, M., AND PATNAIK, L. M. Genetic algorithms: A survey. *computer 27*, 6 (1994), 17–26.

[164] STUART, R., PETER, N., ET AL. Artificial intelligence: a modern approach, 2003.

[165] SUN, X., CHEN, J., XIA, Y., HE, Q., WANG, Y., LUO, X., ZHANG, R., HAN, W., AND WU, Q. A fluctuation-aware approach for predictive web service composition. In *2018 IEEE International Conference on Services Computing (SCC)* (2018), IEEE, pp. 121–128.

[166] SUN, X., WANG, S., XIA, Y., AND ZHENG, W. Predictive-trend-aware composition of web services with time-varying quality-of-service. *IEEE Access 8* (2020), 1910–1921.

[167] TANG, M., AND AI, L. A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on* (2010), IEEE, pp. 1–8.

[168] TSUTSUI, S. A comparative study of sampling methods in node histogram models with probabilistic model-building genetic algorithms. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on* (2006), vol. 4, IEEE, pp. 3132–3137.

[169] TSUTSUI, S., PELIKAN, M., AND GOLDBERG, D. E. Node histogram vs. edge histogram: A comparison of pmbgas in permutation domains. *MEDAL Report*, 2006009 (2006).

[170] VENKATACHALAM, K., KARTHIKEYAN, N., AND KANNIMUTHU, S. Comprehensive survey on semantic web service discovery and composition. *Advances in Natural and Applied Sciences 10*, 5 (2016), 32–41.

[171] VITVAR, T., KOPECKÝ, J., VISKOVA, J., AND FENSEL, D. Wsmo-lite annotations for web services. In *European Semantic Web Conference* (2008), Springer, pp. 674–689.

[172] WADA, H., SUZUKI, J., YAMANO, Y., AND OBA, K. E^3: A multiobjective optimization framework for sla-aware service composition. *IEEE Transactions on Services Computing 5*, 3 (2011), 358–372.

[173] WAGNER, F., ISHIKAWA, F., AND HONIDEN, S. Robust service compositions with functional and location diversity. *IEEE Trans. Services Computing 9*, 2 (2016), 277–290.

[174] WANG, A., MA, H., AND ZHANG, M. Genetic programming with greedy search for web service composition. In *DEXA* (2013), Springer, pp. 9–17.

[175] WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Comprehensive quality-aware automated semantic web service composition. In *AI 2017: Advances in Artificial Intelligence* (2017), Springer, pp. 195–207.

[176] WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Towards robust web service composition with stochastic service failures based on a genetic algorithm. In *AI 2019: Advances in Artificial Intelligence* (2019), Springer, pp. 445–459.

[177] WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Memetic EDA-based approaches to comprehensive quality-aware automated semantic web service composition. *IEEE Transactions on Services Computing* (Submitted).

[178] WANG, C., MA, H., CHEN, A., HARTMANN, S., AND BRANKE, J. Robustness estimation and optimisation for semantic web service composition with stochastic service failures. *IEEE Transactions on Emerging Topics in Computational Intelligence* (Major revision).

[179] WANG, C., MA, H., CHEN, A., HARTMANN, S., AND ONG, Y.-S. Using an Estimation of Distribution Algorithm to achieve multitasking semantic web service composition. *ACM Transactions on Evolutionary Learning and Optimization* (Major revision).

[180] WANG, C., MA, H., AND CHEN, G. EDA-based approach to comprehensive quality-aware automated semantic web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2018), GECCO '18, ACM, pp. 147–148.

[181] WANG, C., MA, H., AND CHEN, G. Using EDA-based local search to improve the performance of NSGA-II for multiobjective semantic web service composition. In *Database and Expert Systems Applications* (2019), Springer, pp. 434–451.

[182] WANG, C., MA, H., CHEN, G., AND HARTMANN, S. GP-based approach to comprehensive quality-aware automated semantic web service composition. In *Simulated Evolution and Learning* (2017), Springer, pp. 170–183.

[183] WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Knowledge-driven automated web service composition — an EDA-based approach. In *Web Information Systems Engineering – WISE 2018* (2018), Springer, pp. 135–150.

[184] WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Towards fully automated semantic web service composition based on estimation of distribution algorithm. In *AI 2018: Advances in Artificial Intelligence* (2018), Springer, pp. 458–471.

[185] WANG, C., MA, H., CHEN, G., AND HARTMANN, S. Evolutionary multitasking for semantic web service composition. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), pp. 2490–2497.

[186] WANG, C., MA, H., CHEN, G., AND HARTMANN, S. A memetic NSGA-II with EDA-based local search for fully automated multi-objective web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2019), GECCO '19, ACM, pp. 421–422.

[187] WANG, D., HUANG, H., AND XIE, C. A novel adaptive web service selection algorithm based on ant colony optimization for dynamic web service composition. In *Algorithms and Architectures for Parallel Processing*. Springer, 2014, pp. 391–399.

[188] WANG, H., MA, P., AND ZHOU, X. A quantitative and qualitative approach for nfp-aware web service composition. In *2012 IEEE Ninth International Conference on Services Computing* (2012), IEEE, pp. 202–209.

[189] WANG, J., TANG, K., LOZANO, J. A., AND YAO, X. Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems. *IEEE Trans. Evolutionary Comput. 20*, 1 (2016), 96–109.

[190] WANG, L., SHEN, J., AND LUO, J. Impacts of pheromone modification strategies in ant colony for data-intensive service provision. In *IEEE International Conference on Web Services* (2014), IEEE, pp. 177–184.

[191] WANG, L., SHEN, J., AND YONG, J. A survey on bio-inspired algorithms for web service composition. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* (2012), IEEE, pp. 569–574.

[192] WANG, P., DING, Z., JIANG, C., AND ZHOU, M. Automated web service composition supporting conditional branch structures. *Enterprise Information Systems 8*, 1 (2014), 121–146.

[193] WANG, P., DING, Z., JIANG, C., ZHOU, M., AND ZHENG, Y. Automatic web service composition based on uncertainty execution effects. *IEEE Transactions on Services Computing 9*, 4 (2016), 551–565.

[194] WANG, S., SUN, Q., ZOU, H., AND YANG, F. Particle swarm optimization with skyline operator for fast cloud-based web service composition. *Mobile Networks and Applications 18*, 1 (2013), 116–121.

[195] WANG, S.-Y., AND WANG, L. An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. *IEEE Trans. Systems 46*, 1 (2016), 139–149.

[196] WANG, W., SUN, Q., ZHAO, X., AND YANG, F. An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems 3*, sup01 (2010), 18–30.

[197] WEN, S., TANG, C., LI, Q., CHIU, D. K., LIU, A., AND HAN, X. Probabilistic top-k dominating services composition with uncertain qos. *Service Oriented Computing and Applications 8*, 1 (2014), 91–103.

[198] WU, H., DENG, S., LI, W., FU, M., YIN, J., AND ZOMAYA, A. Y. Service selection for composition in mobile edge computing systems. In *2018 IEEE International Conference on Web Services (ICWS)* (2018), IEEE, pp. 355–358.

[199] XIA, H., CHEN, Y., LI, Z., GAO, H., AND CHEN, Y. Web service selection algorithm based on particle swarm optimization. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing* (2009), IEEE, pp. 467–472.

[200] XIA, Y.-M., AND YANG, Y.-B. Web service composition integrating qos optimization and redundancy removal. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 203–210.

[201] XIANG, F., HU, Y., YU, Y., AND WU, H. Qos and energy consumption aware service composition and optimal-selection based on pareto group leader algorithm in cloud manufacturing system. *Central European Journal of Operations Research 22*, 4 (2014), 663–685.

[202] XIAO, L., CHANG, C. K., YANG, H.-I., LU, K.-S., AND JIANG, H.-Y. Automated web service composition using genetic programming. In *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops* (2012), IEEE, pp. 7–12.

[203] XIONG, J., XING, L.-N., AND CHEN, Y.-W. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *International J. Production Economics 141*, 1 (2013), 112–126.

[204] XU, B., LUO, S., YAN, Y., AND SUN, K. Towards efficiency of qos-driven semantic web service composition for large-scale service-oriented systems. *Service Oriented Computing and Applications 6*, 1 (2012), 1–13.

[205] XU, C., LIANG, P., WANG, T., WANG, Q., AND SHEU, P. C. Semantic web services annotation and composition based on ER model. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on* (2010), IEEE, pp. 413–420.

[206] XU, X., LIU, Z., WANG, Z., SHENG, Q. Z., YU, J., AND WANG, X. S-abc: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition. *Future Generation Computer Systems 68* (2017), 304–319.

[207] XU, X., RONG, H., PEREIRA, E., AND TROVATI, M. Predatory search-based chaos turbo particle swarm optimisation (ps-ctpso): A new particle swarm optimisation algorithm for web service combi-

nation problems. *Future Generation Computer Systems 89* (2018), 375–386.

[208] YAN, G., JUN, N., BIN, Z., LEI, Y., QIANG, G., AND YU, D. Immune algorithm for selecting optimum services in web services composition. *Wuhan University Journal of Natural Sciences 11*, 1 (2006), 221–225.

[209] YAN, Y., AND CHEN, M. Anytime qos-aware service composition over the graphplan. *Service Oriented Computing and Applications 9*, 1 (2015), 1–19.

[210] YANG, S., ONG, Y.-S., AND JIN, Y. *Evolutionary computation in dynamic and uncertain environments*, vol. 51. Springer Science & Business Media, 2007.

[211] YANG, X.-S., AND DEB, S. Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)* (2009), IEEE, pp. 210–214.

[212] YAO, J., CHEN, S., NEPAL, S., LEVY, D., AND ZIC, J. Truststore: Making amazon s3 trustworthy with services composition. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (2010), IEEE Computer Society, pp. 600–605.

[213] YAO, Y., AND CHEN, H. Qos-aware service composition using nsga-ii 1. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (2009), ACM, pp. 358–363.

[214] YIN, H., ZHANG, C., ZHANG, B., GUO, Y., AND LIU, T. A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem. *Mathematical Problems in Engineering 2014* (2014).

[215] YIN, Y., ZHANG, B., AND ZHANG, X. Qos-driven transactional web service reselection for reliable execution. In *Information Science and Management Engineering (ISME), 2010 International Conference of* (2010), vol. 2, IEEE, pp. 79–82.

[216] YOO, J. J.-W., KUMARA, S., LEE, D., AND OH, S.-C. A web service composition framework using integer programming with non-functional objectives and constraints. In *2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services* (2008), IEEE, pp. 347–350.

[217] YU, Q., AND BOUGUETTAYA, A. Efficient service skyline computation for composite service selection. *Knowledge and Data Engineering, IEEE Transactions on 25*, 4 (2013), 776–789.

[218] YU, Q., CHEN, L., AND LI, B. Ant colony optimization applied to web service compositions in cloud computing. *Computers & Electrical Engineering 41* (2015), 18–27.

[219] YU, Q., LIU, X., BOUGUETTAYA, A., AND MEDJAHED, B. Deploying and managing web services: issues, solutions, and directions. *The VLDB Journal—The International Journal on Very Large Data Bases 17*, 3 (2008), 537–572.

[220] YU, T., ZHANG, Y., AND LIN, K.-J. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB) 1*, 1 (2007), 6.

[221] YU, Y., MA, H., AND ZHANG, M. An adaptive genetic programming approach to QoS-aware web services composition. In *IEEE CEC* (2013), pp. 1740–1747.

[222] YU, Y., MA, H., AND ZHANG, M. A genetic programming approach to distributed qos-aware web service composition. In *2014 IEEE*

Congress on Evolutionary Computation (CEC) (2014), IEEE, pp. 1840–1846.

[223] YU, Y., MA, H., AND ZHANG, M. A hybrid gp-tabu approach to qos-aware data intensive web service composition. In *Asia-Pacific Conference on Simulated Evolution and Learning* (2014), Springer, pp. 106–118.

[224] YU, Y., MA, H., AND ZHANG, M. F-mogp: A novel many-objective evolutionary approach to qos-aware data intensive web service composition. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (2015), IEEE, pp. 2843–2850.

[225] YUAN, Y., ONG, Y.-S., GUPTA, A., TAN, P. S., AND XU, H. Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP. In *TENCON 2016*, IEEE, pp. 3157–3164.

[226] ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., AND SHENG, Q. Z. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 411–421.

[227] ZESHAN, F., AND MOHAMAD, R. Semantic web service composition approaches: overview and limitations. *International Journal on New Computer Architectures and Their Applications (IJNCAA) 1*, 3 (2011), 640–651.

[228] ZHANG, C., NING, J., WU, J., AND ZHANG, B. A multi-objective optimization method for service composition problem with sharing property. *Swarm and Evolutionary Computation 49* (2019), 266–276.

[229] ZHANG, Q., AND LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation 11*, 6 (2007), 712–731.

[230] ZHANG, W., CHANG, C. K., FENG, T., AND JIANG, H.-Y. Qos-based dynamic web service composition with ant colony optimization. In *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual* (2010), IEEE, pp. 493–502.

[231] ZHAO, X., SONG, B., HUANG, P., WEN, Z., WENG, J., AND FAN, Y. An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing 12*, 8 (2012), 2208–2216.

[232] ZHENG, Z., ZHANG, Y., AND LYU, M. R. Investigating qos of real-world web services. *IEEE Trans. Services Computing 7*, 1 (2014), 32–39.

[233] ZHOU, L., FENG, L., ZHONG, J., ONG, Y.-S., ZHU, Z., AND SHA, E. Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), IEEE, pp. 1–8.