ADVANCED EVOLUTIONARY COMPUTATION FOR DYNAMIC AND MULTI-TASK OPTIMIZATION VIA EFFICIENT KNOWLEDGE TRANSFER

By Kejing Du

Thesis submitted in total fulfillment of the requirement for the degree of
Doctor of Philosophy

Victoria University, Australia
Institute for Sustainable Industries and Livable Cities

May 2024

ABSTRACT

Advanced Evolutionary Computation For Dynamic And Multi-Task Optimization Via Efficient Knowledge Transfer

Kejing Du, Ph.D.

Victoria University 2024

Evolutionary computation (EC) is a kind of population-based search method, drawing inspiration from natural selection and gene inheritance. Although EC has shown advantages over traditionally mathmetic-based optimization methods, it often neglects a crucial aspect: knowledge gained from past and other problem-solving experiences. This thesis explores how EC can improve by learning from past experiences or experiences across different tasks. Inspired by knowledge transfer (KT) observed in human evolution through cultural genes, this thesis investigates the potential of EC to acquire and apply knowledge from past and other problem-solving experiences and focuses on dynamic optimization problems (DOP) and multi-task optimization problems (MTOP). Because these types of problems offer ideal opportunities for KT. DOP involves dynamic changes over time, while MTOP optimizes multiple tasks simultaneously, both scenarios benefiting from problem-solving experiences.

This thesis emphasizes the importance of KT, proposing novel EC algorithms for efficiently solving DOP and MTOP. In DOP, a challenge lies in effectively utilizing historical information to accelerate algorithm convergence. This requires solutions for selecting and updating historical data and ensuring its validity amidst environmental changes. Meanwhile, MTOP presents difficulties in balancing optimization objectives across multiple tasks and designing appropriate differential evolution strategies to accommodate different task properties and constraints. Real-world problems are often more complex than theoretical research, involving numerous variables and factors that leading to dynamic charactertics and multi-task charactertics. For example, in bike-

sharing systems, fluctuations in the numbers of available bicycles and stations make the path planning a DOP. Another example is feature selection in deep learning for high-dimensional data. Due to the curse of dimensionality, considering all features is challenging. Thus, integration of KT and EC is essential for addressing practical problems.

Main contents and contributions of this thesis are detailed as follows.

1. To address DOP efficiently, a historical information-based differential evolution (HIDE) algorithm is proposed in Chapter 3. HIDE uses previous knowledge for faster convergence to new optimal regions and employs an archive-based strategy to retain the best-performing individuals from previous environment, facilitating effective KT.

2. To tackle MTOP, a multi-criteria EC algorithm is proposed in Chapter 4. MTOP is conceptualized as a multi-criteria optimization problem (MCOP), where KT occurs across a consolidated population.

3. Chapter 5 addresses dynamic user route planning problem (URPP) in bike-sharing systems. The challenge of fluctuating station inventory turns URPP into DOP. To utilize experiential knowledge and guide the algorithm's search process, knowledge learning and random pruning-based memetic algorithm (KLRP-MA) is introduced, enhancing KT integration and effectively tackling URPP dynamics.

4. In Chapter 6, a practical application of integrating MTOP with the bi-directional feature fixation (BDFF) method in multi-tasking bi-directional particle swarm optimization (MBDPSO) is discussed. This integration allows for effective KT between tasks, improving capabilities in high-dimensional feature selection for deep learning.

In summary, this thesis systematically investigates DOP and MTOP and their practical applications, proposing advanced EC algorithms to enhance the efficiency of KT.

Keywords: evolutionary computation, knowledge transfer, dynamic optimization, multi-task optimization, differential evolution, particle swarm optimization, memetic algorithm, route planning, feature selection

DOCTOR OF PHILOSOPHY DECLARATION

I, Kejing Du, declare that the Ph.D. thesis entitled Advanced Evolutionary Computation For Dynamic And Multi-Task Optimization Via Efficient Knowledge Transfer is no more than 80, 000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work. I have conducted my research in alignment with the Australian Code for the Responsible Conduct of Research and Victoria University's Higher Degree by Research Policy and Procedures.

Signature  Date 02/05/2024

ACKNOWLEDGEMENT

Foremost, I express my deepest appreciation to Professor Hua Wang, my primary supervisor, and Professor Zhi-Hui Zhan, my co-supervisor, for their unwavering professional guidance, continuous encouragement, and constructive advice spanning over three years. They generously invested their time in cultivating my skills in academic research and critical thinking. Without the invaluable help and support of these mentors, my thesis would not have seen the light of success. I am equally appreciative of Prof. Yanchun Zhang, Prof. Jinli Cao, and Dr. Lili Sun for their advice and support. Special thanks are also due to Prof. Sam Kwong and Prof. Weineng Chen for writing recommendation letters for my Ph.D. application.

I extend my gratitude to the esteemed panel members who contributed to my towards submission milestone reviews—Prof Stephen Gray, Dr. Siuly Siuly, and Dr. Kate Wang. Additionally, I appreciate the valuable insights provided by Dr. Khandakar Ahmed, Dr. Kate Wang, and Dr. Yongfeng Ge during my Mid-Candidature review, as well as Dr. Elmira Jamei, Dr. Siuly Siuly, and Dr. Sudha Subramani, who served on the panel for my milestone of Confirmation of Candidature. Thank you all for generously dedicating your time to engage with my reports at each milestone, offering invaluable suggestions, and providing guidance.

I extend my heartfelt gratitude to two invaluable Research Fellows in my laboratory, Dr. Jiao Yin and Dr. Yongfeng Ge. I also want to express my appreciation to my colleagues—Samesad Jahan, Taslima Khanam, Phavithra, Tawhid, Dr. Ashik Mostafa Alvi, Mingshan You, Wei Hong, Puti Xu, Xiyu Qiao, and David Ning. A special mention goes to Dr. Jiao Yin and Ph.D. Student Mingshan You for your continuous support in both my personal and scientific endeavors. I extend my warmest wishes to their daughter, Anna, for a life filled with health and happiness. I would like to convey my sincere appreciation to Xiyu Qiao, Wei Hong, and Samesad Jahan, who have been a constant presence with me in the lab every day. Their kindness and efforts have touched me deeply. A warm thank you to visiting scholar Xingping Zhang; it was truly

a pleasure collaborating with you. Thanks to Prof. Yuan Miao for his encouragement, and we used to meet in the tea room every day at lunchtime. I would like to thank Trish Dwyer, Jo Xuereb who worked in VU for helping with administrative affairs.

I express my gratitude to the China Scholarship Council for generously supporting my living expenses. I convey my thanks to the Chinese Consulate General in Melbourne for their dedicated services and arrangements for international students. I am indebted to Victoria University for the waiver of my tuition fees and for providing essential research equipment, a conducive laboratory environment, and access to sports venues.

I extend my heartfelt gratitude to Prof. Zhi-Hui Zhan, Dr. Jian-Yu Li, and Dr. Sheng-Hao Wu, Master Jiaquan Yang, Master Min Gao, who are currently in China. Due to the challenges posed by Covid-19, my arrival at Victoria University was delayed. Fortunately, I seized the opportunity to initiate research work at the South China University of Technology, collaborating with exceptional scientists in their lab, resulting in significant scientific research outcomes. I eagerly anticipate future opportunities for collaboration.

Finally, I convey my heartfelt appreciation to my family in Guangzhou. My mother remained in Guangzhou to care for my two young children. I extend my deepest appreciation to my understanding and supportive husband, who has been a pillar of encouragement throughout my academic journey. I also want to thank my father, grandpa, and grandma for their unwavering support. Special thanks go to my neighbors, Xiaohong Yu and Prof. Bing Hu, who frequently assist with picking up and dropping off my children from our apartment to school. I am grateful to my children's teachers and the parents of their classmates, who have occasionally lent a helping hand in looking after my children during my absence. A profound sense of gratitude is reserved for my two sons, Wenyu Zhan, and Wenbo Zhan. Despite being primary students, they demonstrated remarkable dedication to their studies and a commendable ability to live independently during my doctoral studies.

PUBLICATIONS

1. **Ke-Jing Du**, Jian-Yu Li, Hua Wang, and Jun Zhang, “Multi-objective multi-criteria evolutionary algorithm for multi-objective multi-task optimization,” *Complex & Intelligent Systems*, vol. 9, pp. 1211-1228, 2023. [IF=5.8, JCR Q1]
2. **Ke-Jing Du**, Jian-Yu Li, Hua Wang, and Jun Zhang, “A knowledge learning and random pruning-based memetic algorithm for user route planning in bike-sharing system,” *Memetic Computing*, vol. 15, no. 2, pp. 259-279, Jun. 2023. [IF=4.7, JCR Q1]
3. **Ke-Jing Du**, Jia-Quan Yang, Limin Wang, Xuming Han, Hua Wang, and Zhi-Hui Zhan, “Multi-objective demand responsive transit scheduling in smart city: a multiple populations ant colony system approach,” in *Proceedings of 16th International Conference on Advanced Computational Intelligence (ICACI 2024)*, Zhangjiajie, China, May 2024, pp. 197-205.
4. Jian-Yu Li, **Ke-Jing Du**, Zhi-Hui Zhan, Hua Wang, and Jun Zhang, “Distributed differential evolution with adaptive resource allocation,” *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 2791-2804, May. 2023. [IF=11.8, JCR Q1]
5. Jia-Quan Yang, Qi-Te Yang, **Ke-Jing Du**, Chun-Hua Chen, Hua Wang, Sang-Woon Jeon, Jun Zhang, and Zhi-Hui Zhan, “Bi-Directional feature fixation-based particle swarm optimization for large-scale feature selection,” *IEEE Transactions on Big Data*, vol. 9, no. 3, pp. 1004-1017, May./Jun. 2023. [IF=7.2, JCR Q1]
6. Chuan Wang, Bing Sun, **Ke-Jing Du**, Jian-Yu Li, Zhi-Hui Zhan, Sang-Woon Jeon, Hua Wang, and Jun Zhang, “A novel evolutionary algorithm with column and sub-block local search for sudoku puzzles,” *IEEE Transactions on Games*, vol. 16, no. 1, pp. 162-172, Mar. 2024. [IF=2.3, JCR Q2]
7. Yong Zhang, **Ke-Jing Du (Corresponding Author)**, Yi Jiang, Li-Min Wang, Hua Wang, and Zhi-Hui Zhan, “Adaptive aggregative multitask competitive particle swarm optimization with bi-directional asymmetric flip strategy for high-dimensional feature selection,” in *Proceedings of the ACM Genetic and*

Evolutionary Computation Conference (GECCO 2024), Melbourne, Australia, Jul. 2024. (Accepted)

8. Min Gao, **Ke-Jing Du (Co-First Author)**, Pei-Yao Zhu, Jian-Yu Li, Hua Wang, and Zhi-Hui Zhan, "A robust two-part modeling strategy for knowledge graph enhanced recommender systems," in *Proceedings of 15th International Conference on Advanced Computational Intelligence (ICACI 2023)*, Seoul, Korea, May. 2023, pp. 1-7.
9. Jia-Quan Yang, **Ke-Jing Du (Co-First Author)**, Chun-Hua Chen, Hua Wang, Jun Zhang, and Zhi-Hui Zhan, "Evolutionary multitasking bi-directional particle swarm optimization for high-dimensional feature selection," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2023)*, Jul. 2023, pp. 1-8.
10. Pei-Yao Zhu, Sheng-Hao Wu, **Ke-Jing Du**, Hua Wang, Jun Zhang, and Zhi-Hui Zhan, "Diversity-driven multi-population particle swarm optimization for dynamic optimization problem," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 2023)*, Jul. 2023, pp. 107–110.
11. Jun Hong, Lin Shi, **Ke-Jing Du**, Chun-Hua Chen, Hua Wang, Jun Zhang, Zhi-hui Zhan, "A Multi-population genetic algorithm for multiobjective recommendation system," in *Proceedings of IEEE Symposium Series on Computational Intelligence (SSCI 2023)*, Mexico City, Mexico, Dec. 2023, pp. 998-1003.
12. Sheng-Hao Wu, **Ke-Jing Du**, Zhi-Hui Zhan, Hua Wang, and Jun Zhang, "Historical information-based differential evolution for dynamic optimization problem," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2021)*, Jun. 2021 pp. 119-126.
13. Jian-Yu Li, **Ke-Jing Du**, Zhi-Hui Zhan, Hua Wang, and Jun Zhang, "Multi-criteria differential evolution: treating multitask optimization as multi-criteria optimization," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 2021)*, Jul. 2021, pp. 183–184.

LIST OF ABBREVIATION

Abbreviation	Full form
ACO	ant colony optimization
AHIR	archive-based historical information reuse
AI	Artificial Intelligence
APL	adaptive parameter learning
BDFF	bi-directional feature fixation
CC	cooperative coevolution
DE	differential evolution
DMP	distributed multiple populations
DOP	dynamic optimization problems
DyKLRP-MA	dynamic knowledge learning and random pruning-based memetic algorithm
EC	evolutionary computation
EDA	estimation of distribution algorithms
EMTO	evolutionary multi-task optimization
EP	evolutionary programming
ES	evolution strategies
ETO	evolutionary transfer optimization
FA	fire-fly algorithm
GA	genetic algorithms
GP	genetic programming
HIDE	historical information-based differential evolution
IA	immune algorithm
KLC	knowledge learning-based crossover
KLRP-MA	knowledge learning and random pruning-based memetic algorithm

KT	knowledge transfer
MA	memetic algorithm
MBDPSO	multi-task bi-directional particle swarm optimization
MCDE	multi-criteria differential evolution
MCOP	multi-criteria optimization problem
MFEA	multi-objective evolutionary algorithm
MO-MCEA	multi-objective multi-criteria evolutionary algorithm
MO-MCOP	multi-objective multi-criteria optimization problem
MO-MTOP	multi-objective MTOP
MPB	moving peaks benchmark
PSO	particle swarm optimization
RPM	random pruning-based mutation
RSI	region-based subpopulation initialization
TPLS	two-phase local search
URPP	user route planning problem
VRS	variable relocation strategy

TABLE OF CONTENTS

ABSTRACT **2**
DOCTOR OF PHILOSOPHY DECLARATION **4**
ACKNOWLEDGEMENT **5**
PUBLICATIONS **7**
LIST OF ABBREVIATION **9**
TABLE OF CONTENTS **11**
LIST OF TABLES **13**
LIST OF FIGURES **16**
CHAPTER 1 ..... **16**
1.1 Research Background 17
1.2 Research Motivations 18
1.2.1 Research Motivation of DOP 18
1.2.2 Research Motivation of MTOP 19
1.3 Research Content 20
1.3.1 Research Content of DOP 21
1.3.2 Research Content of MTOP 22
1.4 Research Contributions 24
1.5 Thesis Structure 25
CHAPTER 2 ..... **28**
2.1 Evolutionary Computation 28
2.1.1 Differential Evolution Algorithm 30
2.1.2 Particle Swarm Optimization 32
2.1.3 Memetic Algorithm 33
2.2 Knowledge Transfer 34
2.2.1 Introduction of Knowledge Transfer 34
2.2.2 KT in EC 35
2.3 Dynamic Optimization Problems (DOP) 37
2.3.1 Introduction of DOP 37
2.3.2 KT in DOP 41
2.4 Multi-task Optimization Problems (MTOP) 46
2.4.1 Introduction of MTOP 46
2.4.2 KT in MTOP 48
CHAPTER 3 ..... **52**
3.1 Introduction 52

3.2 Related Work 54
3.2.1 Dynamic Optimization Problem 54
3.2.2 Differential Evolution 56
3.2.3 Multi-population Methods 56
3.3 Framework of HIDE Method 59
3.3.1 RSI Strategy (region-based subpopulation initialization) 59
3.3.2 DE Optimization Process 61
3.3.3 Subpopulation Merge 61
3.3.4 Archive-based historical information reuse (AHIR) 63
3.3.5 The Whole HIDE Algorithm 65
3.4 Experiment 66
3.4.1 Experiment Setting 66
3.4.2 Performance Measure 69
3.4.3 Results and Discussions 69
3.5 Effect of the AHIR Strategy 71
3.6 Conclusion 71
CHAPTER 4 73
4.1 Introduction 73
4.2 Related Work 75
4.3 Treating Multi-Task Optimization as Multi-Criteria Optimization 77
4.3.1 Introduction 77
4.3.2 Method 79
4.3.3 Experiment 79
4.3.4 Conclusion 80
4.4 MCOP for Multi-Objective Multi-Task Optimization 81
4.4.1 Introduction 81
4.4.2 Method of MO-MCEA 83
4.4.3 Experiment 89
4.4.4 Conclusion and Future Work 90
4.5 Conclusion 92
CHAPTER 5 93
5.1 Introduction 93
5.2 URPP Model 97
5.3 KLRP-MA Approach 99
5.3.1 Encoding Scheme 99
5.3.2 Population Initialization 100

5.3.3 Knowledge Learning-based Crossover 101
5.3.4 Random Pruning-based Mutation 104
5.3.5 Two-phase Local Search 104
5.3.6 The Completed Algorithm 104
5.4 DyKLRP-MA Approach 105
5.5 Experimental Results 108
5.5.1 Experimental Design 108
5.5.2 Experimental Results under Static Situations 109
5.5.3 Experimental Results under Dynamic Situations 116
5.5.4 Experiments on Real-world Bike-sharing Instances 118
5.5.5 Component Analysis 119
5.5.6 Influence of Parameter 121
5.6 Conclusion 124
CHAPTER 6 125
6.1 Introduction 125
6.2 Related Work 130
6.2.1 Feature Selection 130
6.2.2 Bidirectional Feature Fixing Framework 130
6.2.3 Evolutionary Multi-Task Optimization 131
6.3 Framework of MBDPSO 132
6.3.1 Two tasks of feature selection 132
6.3.2 Multi-Task Knowledge Transfer 134
6.3.3 Complete MBDPSO 135
6.4 Experiment Results and Discussion 137
6.4.1 Datasets 137
6.4.2 Experimental Setup 137
6.4.3 Compare the results and discussion 138
6.4.4 Composition Analysis of MBDPSO 141
6.4.5 Impact of Parameter K 142
6.5 Conclusion 143
CHAPTER 7 144
7.1 Conclusions 144
7.2 Future Work 144
BIBLIOGRAPHY 146

LIST OF TABLES

Table 3.1 Parameter Setting for MPB 68

Table 3.2 Experimental Results on the MPB with P=5 70

Table 3.3 Experimental Results on the MPB with P=10 70

Table 3.4 Experimental Results on the MPB with P=20 70

Table 3.5 Experimental Results of the Effect of the AHIR Strategy on the MPB Test Suite 71

Table 4.1 Number of papers of MTOP in the last ten years 74

Table 4.2 Comparison with State-of-the-Art Algorithms 80

Table 4.3 Comparisons between the proposed MO-MCEA and state-of-the-art algorithms 91

Table 5.1 The experimental results between KLRP-MA and competitor methods on the medium-distance and small-scale test instances 110

Table 5.2 The experimental results between KLRP-MA and competitor methods on the medium-distance and medium-scale test instances 110

Table 5.3 The experimental results between KLRP-MA and competitor methods on the medium-distance and large-scale test instances 111

Table 5.4 The experimental results between KLRP-MA and competitor methods on the long-distance and small-scale test instances 112

Table 5.5 The experimental results between KLRP-MA and competitor methods on the long-distance and medium-scale test instances 112

Table 5.6 The experimental results between KLRP-MA and competitor methods on the long-distance and large-scale test instances 113

Table 5.7 The running time of KLRP-MA and the enumeration method on the medium-distance test instances (Unit: Second) 114

Table 5.8 The running time of KLRP-MA and the enumeration method on the long-distance test instances (Unit: Second) 115

Table 5.9 The experimental results of DyKLRP-MA on the long-distance test instances 117

Table 5.10 The experimental results of DyKLRP-MA, ma-r, and ma-a on the real-world test instances 119

Table 5.11 The experimental results among KLRP-MA variants on the long-distance and small-scale

test instances. 120
Table 5.12 The experimental results among KLRP-MA variants on the long-distance and medium-scale test instances. 120
Table 5.13 The experimental results among KLRP-MA variants on the long-distance and large-scale test instances. 121
Table 5.14 The optimization results and running time of different KLRP-MA variants with different maximum dimensions on the long-distance and medium-scale test instances. 124
Table 5.15 The optimization results and running time of different KLRP-MA variants with different generation probabilities on the long-distance and medium-scale test instances. 122
Table 6.1 Basic information of 10 data sets 138
Table 6.2. Classification accuracy of MBDPSO and other comparative algorithms on 10 datasets (bold numbers indicate the best results). 139
Table 6.3 Average number of selected features by algorithms on 10 datasets. 140
Table 6.4 Comparison of running time 140
Table 6.5. Experimental results of component analysis. 142
Table 6.6. Experimental results of different variants of MBDPSO with various *K* values 142

LIST OF FIGURES

Figure 1.1 Research Structure 21

Figure 2.1 The flowchart of DE 31

Figure 2.2 The flowchart of EC 36

Figure 4.1 Number of papers of MTOP in the last ten years 75

Figure 4.2 Framework of MO-MCEA 86

Figure 5.5 Illustration of S_c and S_n for the current situation and new situation. 106

Figure 5.6 Visualization of bike location data in real-world test instances. 118

Figure 5.7 The convergence curve of different KLRP-MA variants on the L-M-1 test instance. 123

CHAPTER 1

INTRODUCTION

1.1 Research Background

Evolutionary computation (EC) is an important branch of Artificial intelligence (AI) [1]. AI is one of the transformative technologies that have profoundly transformed our work and life, now and in the future. With the emergence of emerging technologies such as large-scale AI models, autonomous driving, and smart cities, it is widely recognized that AI has started to reshape almost all jobs, industries, and lives. As a developed country, AI has been widely deployed in various industries in Australia. The Australian government is a world leader in investment and research in AI and attaches great importance to the combination of government, scientific research institutions, and industry development. In various industries, optimization algorithms are widely utilized, including medical and health [2]-[3], privacy preservation [4]-[12], cybersecurity [13]-[15], internet of things [16], smart cities [17], natural language processing [18]-[19], and many other fields. These findings are essential for maintaining Australia's world leadership in AI and shaping new frontiers as they evolve.

The advancement of AI has faced significant challenges, encountering bottlenecks in its development. In various industries, managing large enterprises may involve dealing with millions of variables, representing a high-dimensional optimization challenge. Many of these variables exhibit nonlinearity and non-convexity, presenting formidable mathematical challenges. As market dynamics evolve, enterprises are required to respond rapidly, intensifying the complexity of dynamic optimization problems (DOP). The modern enterprise market's supply chain is complex, spanning the globe with suppliers and factories. Beyond pursuing profits, enterprises must address a range of social responsibilities, including energy security, ecological considerations, and safeguarding employee rights. Enterprises frequently find themselves needing to reconcile multiple optimization tasks simultaneously, which are

multi-task optimization problems (MTOP).

Human civilization has developed rapidly in the past 300 years. Therefore, many scholars began to explore approaches to solve the bottleneck problem of EC from the perspective of human evolution. The idea of Knowledge Transfer (KT) has generated the interest of leading scholars in the EC domain [21]-[23]. The difference between the human brain and computer problem solving is that the human brain can use past historical experience, while the computer solves optimization problems from zero every time [21], which is inefficient for difficult problems. In real-world optimization problems, DOP and MTOP will face a large number of similar problems [22], and the problems are either repeated or have domain-specific similarities. Therefore, this thesis aims to investigate a research direction that EC has not fully explored to date, evolutionary transfer optimization (ETO), exploring the combination of EC and KT to solve DOP and MTOP.

1.2 Research Motivations

1.2.1 Research Motivation of DOP

In practical scenarios, optimization problems often exhibit dynamic attributes, where the optimal solution may change over time as a result of changing environmental conditions [24]. DOP present unique challenges compared to static optimization problems, as they require algorithms to adapt to changing environments, varying problem dimensions, and shifting search spaces [25]. Traditional optimization techniques may be difficult to cope with these dynamic changes effectively.

EC algorithms offer promising solutions for addressing DOP due to their inherent qualities such as global search capability, adaptability, and parallelism [24]. Moreover, EC algorithms can use memory mechanisms to retain past experiences, enabling them to capture the relationship between previous and current environments in DOP [25]. KT techniques play a vital role in improving the performance of EC algorithms by facilitating the transfer of valuable information and insights gained from past optimization processes to guide the search in dynamic environments [27].

Despite the potential of EC algorithms in addressing DOP, there are significant

challenges to overcome. One key challenge is enabling populations to escape from previous optimal solutions when environmental changes occur, and rapidly identify new optimal solutions in the altered environment. Integrating KT with DOP within EC algorithms has emerged as a prominent research direction to address these challenges effectively.

Based on the practical applications of DOP, existing research often relies on certain assumptions to support algorithmic development. For instance, it is commonly assumed that while DOPs are overall dynamic, there are specific time intervals where the problem nature remains relatively static, with minimal changes. Additionally, neighboring static time intervals often exhibit some degree of similarity in problem features, allowing for the partial reuse of information from previous environments to expedite optimization in subsequent intervals.

By exploring and addressing these challenges, our research strives to enhance the comprehension and efficiency of EC algorithms in handling DOP, ultimately contributing to the development of more robust and efficient optimization techniques for real-world applications.

1.2.2 Research Motivation of MTOP

Despite the advancements in EC in recent years, it still faces two critical challenges: heavy computational burdens and limited generalization abilities [28]. To overcome these obstacles, researchers have turned to MTOP as a promising strategy [29]. MTOP involves addressing multiple distinct optimization problems concurrently within the same framework. These problems may interact at the task level, allowing for the sharing of knowledge and resources among them. This approach is analogous to how individuals apply learning techniques across different subjects, leveraging shared insights and methodologies.

In contrast to traditional KT methods, which typically involve one-way transfers of knowledge, MTOP adopts a bidirectional KT approach [30]. This approach fosters mutual reinforcement between different tasks, enhancing overall optimization performance. Representative algorithms in this domain, such as the Multi-Objective

Evolutionary Algorithm (MFEA) [30], have demonstrated the efficacy of bidirectional KT in MTOP settings. Subsequent research has further refined and extended these approaches.

By leveraging bidirectional KT, MTOP efficiently harnesses the parallel optimization capabilities of algorithms while integrating cross-disciplinary knowledge to enhance overall performance. Existing studies have primarily implemented KT through genetic operations, such as selection and crossover, shared among different tasks [31]-[32]. While these approaches effectively transfer knowledge between tasks, there is still room for more innovative strategies to address the complex challenges inherent in MTOP.

In summary, our research aims to explore novel approaches for addressing the challenges of MTOP within EC algorithms. While existing studies have delved into both one-way and bidirectional KT methods, our focus lies in advancing the current edge through innovative strategies that efficiently leverage bidirectional KT for enhanced optimization performance.

1.3 Research Content

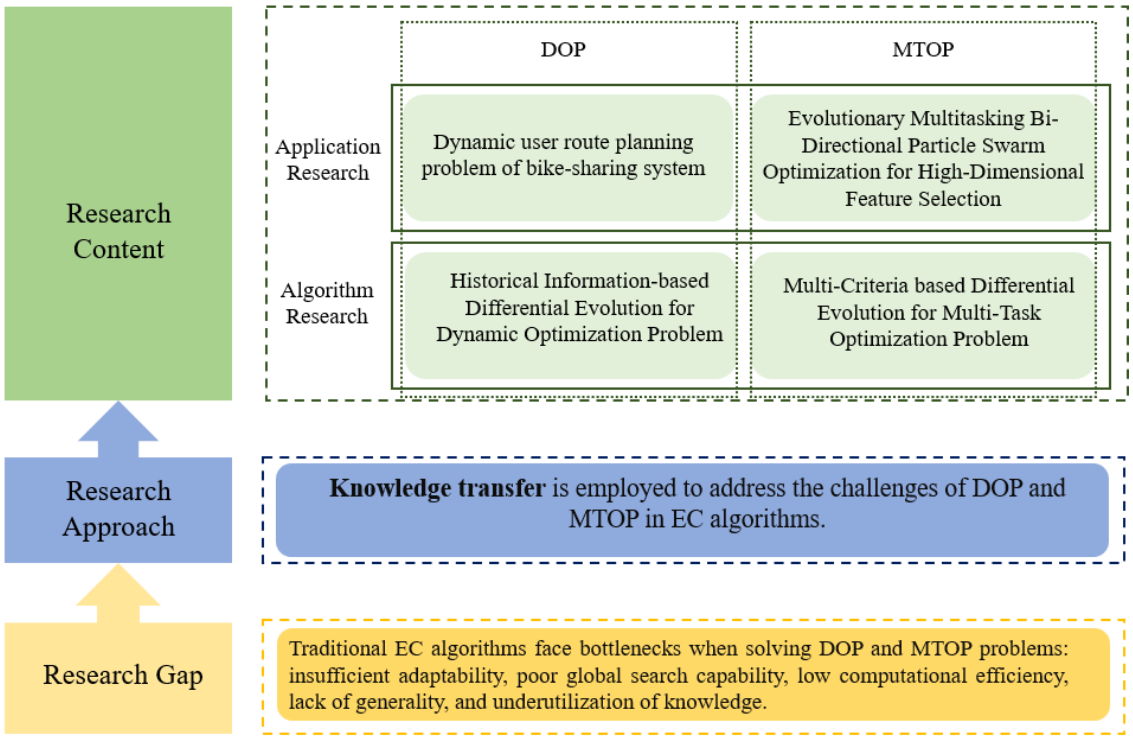


Figure 1.1 Research Structure

As illustrated in Figure 1, this thesis is dedicated to addressing the challenges encountered in DOP and MTOP using KT principles. The research content is divided into algorithmic research and application research. In algorithmic research, it includes studies on algorithms for solving DOP and algorithms for addressing MTOP. Correspondingly, in application research, there are chapters dedicated to the application research of DOP and MTOP, respectively.

1.3.1 Research Content of DOP

To enhance the performance of EC algorithms in DOP, early review papers categorized research methods into five types when changes occur: introducing variety during alterations, preserving diversity during exploration, techniques based on memory, methods based on prediction, and approaches involving multiple populations [24]. In a recent review, Zhan et al. consolidated these into three functions: reducing problem complexity, enhancing algorithm variety, and speeding up convergence [25]. Among these, multi-population methods are commonly employed, falling under the category of increasing algorithm diversity. Each sub-population can be considered as decomposing the problem, focusing on solving a specific portion. This decomposition aids in addressing the complexity of the problem space, enhancing scalability, and enabling the resolution of large-scale problems. Memory-based and prediction-based methods aim to expedite convergence speed. As per the two assumptions mentioned earlier, continuous dynamic environments often exhibit strong correlations, letting historical solutions for reuse has great potential to expedite convergence in novel environments. Consequently, our research on DOP primarily addresses two key issues: multi-population methods and the reuse of historical optimal solutions, aiming to effectively tackle DOP. The specific research problems are outlined below.

1) Regarding multi-population methods, the issue involves the complexity and multimodality of the problem search space in DOP. How to generate multiple subpopulations in a balanced manner in a new environment to effectively address DOP with multiple peaks?

2) For methods utilizing historical optimal solutions, the problem focuses on effectively coping with changes in dynamic environments and leveraging historical information to enhance search efficiency. How to fully use peak information found from the preceding context to direct exploration in a new environment?

To address these challenges, a region-based subpopulation initialization (RSI) strategy is first proposed to produce numerous subpopulations in a balanced way in the new environment. Through the initialization of multiple subpopulations across diverse regions within the search space, diversity is enhanced, thereby facilitating the resolution of DOP characterized by multiple peaks. Second, to effectively harness previously discovered peaks within the environment, an archive-based historical information reuse (AHIR) strategy is proposed. This strategy involves managing and reusing historical information to navigate the search process in novel environments.

In practical applications, shared bicycle path optimization is chosen as an applied study of DOP. In this real-world application, the investigated problems are as follows:

1) In a shared bicycle system, users have limited free riding time, but mid-to-long-distance rides may exceed the free time threshold. Reasonable transfers can address this issue. How to design bike routes with transfers?

2) Assuming bike station inventory changes over time, how to adjust user riding routes?

To address these issues, a dynamic knowledge learning and random pruning-based memetic algorithm (DyKLRP-MA) is further proposed to adjust reasonable riding routes for users in dynamic scenarios.

1.3.2 Research Content of MTOP

In tackling this problem, a Multi-Criteria Optimization Problem (MCOP) was proposed to solve MTOP. This approach enables the knowledge inheritance from all the tasks within a single population, thus enhancing the effectiveness of MTOP's solution. Similar to students in a class studying multiple subjects, a testing method is adopted to improve academic performance, randomly selecting subjects for testing, and students learn based on the test results to find optimal solutions. The main focus of this study is

to address the following three issues:

1) In the context of MTOP, existing methods often treat MTOP as separate tasks, posing challenges in designing effective KT strategies between tasks/populations. To what extent does this issue impact the effectiveness of solving MTOP?

2) How can multiple relevant evaluation criteria aid in individual selection and evolution?

3) How can the utilization of multiple relevant evaluation criteria contribute to the process of individual selection and evolution?

In MCOP, each task's fitness evaluation function serves as a criterion, offering multiple pertinent criteria to aid in the selection and evolution of individuals across various stages. Moreover, a criterion selection strategy based on probabilities and a method for adaptive parameter learning is proposed to optimize the choice of fitness functions as criteria during different evolutionary phases. This enables the algorithm to effectively utilize suitable criteria from different tasks at different evolution stages, guiding individual selection and population evolution toward discovering Pareto-efficient solutions for all tasks. Through the integration of these methods, a comprehensive MCOP framework tailored for addressing MTOP is presented.

In the context of applying MTOP to high-dimensional feature selection problems, EC is used to mitigate the inefficiency associated with selecting features from datasets containing a large number of attributes. The goal is to enhance model performance or reduce computational costs by identifying the most relevant or important features. While the Bi-Directional Feature Fixation (BDFF) method for Particle Swarm Optimization (PSO) has shown promise in high-dimensional feature selection, it may suffer from directional biases and prolonged convergence times when searching for small feature subsets. This study primarily addresses two key issues:

1) How to address the inefficiency in high-dimensional feature selection?

2) What are the limitations of the Bi-Directional Feature Fixation (BDFF) framework?

Prior knowledge of attribute selection is introduced into the Bi-Directional Feature

Fixation (BDFF) method while preserving its global search capability. Subsequently, by combining BDFF with the MTOP technique, Multi-Task Bi-Directional Particle Swarm Optimization (MBDPSO) is proposed, effectively transferring knowledge between two tasks.

1.4 Research Contributions

This thesis contributes significantly to theoretical research in three distinct aspects:

The primary theoretical contribution focuses on the integration of KT concepts into EC for optimization problems by leveraging historical information to DOP and adopting multi-criteria strategies to MTOP. This approach, inspired by the efficiency of human KT, equips algorithms to adeptly address new tasks and future optimization challenges. The theoretical foundation laid in this aspect enhances the overall efficacy of EC.

The introduction of the HIDE algorithm tackles a critical challenge in DOP by efficiently leveraging past information and knowledge to quickly identify and converge to new optimal regions. HIDE innovatively employs a strategy for reusing historical information stored in an archive, retaining the best individuals from previous environments. Additionally, it utilizes a strategy for initializing subpopulations based on regions to fully exploit peaks identified in the preceding environment. This balanced approach in generating multiple subpopulations aids in effectively localizing and tracking movements to reach the peak.

In the realm of MTOP, the theoretical contribution is embodied in the MCOP algorithm. Existing methods in the multi-task optimization community often treat tasks in MTOP as distinct problems, overlooking their interconnected nature. Addressing this gap, MCDE advocates treating the entire MTOP as a MCOP. This shift in perspective aims to enhance the efficiency of MTOP solutions by considering tasks as constituent elements within the broader optimization framework.

In summary, the thesis advances theoretical research by introducing innovative concepts such as KT in EC, the HIDE, and the paradigm shift in approaching MTOP through the MCOP. These contributions deepen our understanding and provide valuable

frameworks for addressing complex optimization challenges.

This thesis makes dual contributions in the realm of application research:

In the DOP application, the primary contribution lies in proposing a robust solution to the dynamic nature of URPP within shared bicycle systems. The innovation here is the introduction of a Memetic Algorithm based on KLRP-MA. The key contribution is twofold: first, the algorithm efficiently adapts to dynamic changes in bicycle station availability, ensuring swift re-optimization of planned routes. Second, the incorporation of a KT mechanism from the best-performing individual accelerates convergence, significantly improving the algorithm's efficiency. By addressing dynamic URPP through KLRP-MA, the thesis provides a tangible and effective solution for real-world applications.

In MTOP application, the major contribution is the integration of MTOP with the BDFF framework within a MBDPSO algorithm. This integration significantly enhances the algorithm's effectiveness in selecting features within high-dimensional spaces for pattern recognition across multiple tasks. The real-world impact of this contribution is evident in the improved performance of the algorithm, demonstrating its efficacy in solving complex MTOP. This integration not only advances the theoretical understanding of MTOP but also provides a practical tool for researchers and practitioners working in the field of pattern recognition.

In summary, the thesis contributes by presenting effective solutions to dynamic route planning challenges through KLRP-MA and by enhancing the capabilities of multi-task optimization through the integration of MTOP with the BDFF framework. These contributions address real-world challenges and provide valuable insights and tools for researchers and practitioners in related fields.

1.5 Thesis Structure

The thesis, titled “ Evolutionary Computation-based Dynamic and Multi-Task Optimization and the Application,” is structured as depicted in Figure 1.1. The research encompasses two main aspects: algorithm research and application research. Algorithmic research delves into two distinct challenges: DOP in Chapter 3 and MTOP

in Chapter 4. In parallel, application research explores the DOP related to shared bicycles in Chapter 5 and MTOP in Feature Selections in Chapter 6.

This thesis comprises seven chapters in total.

Chapter 1 is the Introduction, which includes Section 1.1 providing an overview of the research context, Section 1.2 exploring the research motivation and listing the research questions, Section 1.3 outlining the research content, Section 1.4 highlighting the research contributions, and Section 1.5 detailing the structure of the study and the content of each chapter.

In Chapter 2, a comprehensive review of existing literature is conducted, including an overview of EC in Section 2.1 and KT in Section 2.2. Furthermore, it discusses DOP in Section 2.3 and MTOP in Section 2.4, along with KT ideas in these two complex problems.

Chapter 3 proposed HIDE method for DOP. It starts with Section 3.1 of the introduction and Section 3.2 of related work. Section 3.3 elaborates on the Framework of HIDE. Section 3.4 details the Experiment, Section 3.5 explores the impact of the AHIR Strategy, while Section 3.6 concludes the chapter.

Chapter 4 delves into Multi-Objective Multi-Task Optimization (MO-MTO). Section 4.1 is an introduction and Section 4.2 discusses related work. Section 4.3 introduces the concept of treating multitask optimization as multi-criteria optimization. Section 4.4 adds the challenge of Multi-Objective to Multi-Task Optimization, comprising Introduction, Method of MO-MCEA, Experiment, and Conclusion and Future Work. Finally, Section 4.5 concludes the chapter.

Chapter 5 covers the URPP Model and the KLRP-MA Approach. It begins with an Introduction in Section 5.1, providing an overview of the research focus. Section 5.2 delves into the URPP Model, while Section 5.3 introduces the KLRP-MA Approach and Section 5.4 introduces DyKLRP-MA approach. Section 5.5 is for the experiment and Section 5.6 draw a conclusion.

Chapter 6 presents the MBDPSO framework, starting with an Introduction in Section 6.1 and related work in Section 6.2. Section 6.3 outlines the Framework of

MBDPSO, including Two tasks of feature selection, Multi-Task KT, and the Complete MBDPSO algorithm. Experimental Results is in Section 6.4. Finally, the chapter concludes in Section 6.5, summarizing the findings and contributions of the MBDPSO framework.

Chapter 7 summarizes the research undertaken in this thesis and anticipates feasible research directions and subsequent work in the future.

CHAPTER 2

LITERATURE REVIEW

In Chapter 2, the definitions of the key concepts associated with Evolutionary Computation (EC), Knowledge Transfer (KT), Dynamic Optimization Problem (DOP), and Multi-Task Optimization Problem (MTOP) are thoroughly explored. DOP and MTOP are categorized as types of problems, while EC is presented as the method employed to address these problems. Additionally, KT is discussed as a means of knowledge assistance throughout the exploration. The chapter includes a comprehensive review of prior research, establishes a theoretical framework, traces the evolution of research in the field, and identifies research gaps in the context of these concepts.

2.1 Evolutionary Computation

Over the centuries, a myriad of mathematical approaches, ranging from Linear Programming [33], Quadratic Programming [34], and Convex Optimization [35], have been harnessed to address optimization problems. Nevertheless, these methodologies frequently impose substantial constraints on the objective function, often mandating differentiability once or twice. Furthermore, the dependence on mathematical techniques in optimization endeavors often leads to the identification of local optimal solutions, especially in the face of multi-modal problems. Therefore, since the 1960s, an increasing number of scholars have delved into the realm of EC as a viable alternative method [36]. In the 1990s, EC has emerged as a promising global optimization technique for many optimization problems [37]. EC is typically classified into two main branches: Evolutionary Algorithms (EA) and Swarm Intelligence (EI).

EA simulate the biological evolutionary process and principles of natural selection to optimize problems [38]. EA has achieved success and popularity due to its algorithmic characteristics of being assumption-free, flexible, robust, and capable of global optimization [39]. EA comprises Evolutionary Programming (EP), Genetic

Programming (GP), Genetic Algorithms (GA), and Evolution Strategies (ES). New techniques that emerged in the 1990s, such as Estimation of Distribution Algorithms (EDA) and Differential Evolution (DE) are also regarded as EA.

Swarm intelligence (SI) is a computational model that simulates collective behaviors observed in nature, drawing inspiration from the collective wisdom exhibited by social organisms such as ants, bees, and bird flocks [40]. In SI, individuals collaborate through interactions and information exchange to achieve a common goal. This process initiates from a chaotic state and gradually unfolds by exploring valuable heuristic information. It systematically reveals patterns, regularities, and knowledge within the problem, ultimately leading to a solution. SI evolves through dynamic processes characterized by randomness, nonlinearity, traversal, self-organization, adaptability, diversity, stability, and high parallelism. This implies that through such a process, solutions to problems can be discovered in a diverse and highly parallel manner, adapting dynamically and incorporating elements of randomness. Its representative methods include Fire-fly algorithm (FA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), immune algorithm (IA), etc.

The memetic algorithm (MA), a heuristic optimization technique, integrates the broad exploration features of evolutionary algorithms with the precise adjustments of local search methods [41]. The name “memetic” is derived from “meme”, signifying the propagation of beneficial information through learning and adaptation within a population [41]. The memetic algorithm typically comprises an evolutionary phase, where individuals evolve through operations such as crossover and mutation, and a local search phase, where individuals undergo more refined adjustments. The integration of these approaches seeks to achieve equilibrium between global and local exploration, thereby improving the efficiency in discovering high-quality solutions. The integration of memetic algorithms with KT has been applied in data-driven domains [42] and vehicle routing optimization [43] [29]. This synergy leverages learned information and adaptive techniques to enhance the algorithm's efficacy across diverse scenarios.

2.1.1 Differential Evolution Algorithm

Differential Evolution (DE) belongs to the realm of EC, serving as an algorithm that leverages individual differences among groups to steer the evolutionary process. Storn and Price introduced the DE algorithm in 1995, originally designed to address Chebyshev polynomial problems [44]. Subsequently, it was discovered that DE proves to be a potent technique for tackling complex optimization problems. DE often exhibits superior global search capabilities and faster convergence speeds.

Assume that there are NP individuals in the solution space (that is, the population size is NP), and each individual is a vector with *D* dimensions. The initial population **is** generated randomly:

$$\overrightarrow{X_i^g} = [x_{i,1}^g, x_{i,2}^g, \cdots, x_{i,D}^g] \tag{2.1}$$

where *g* is the evolutionary algebra and *D* is the dimensionality. And *i* represents the individual number. Each individual is a solution. During initialization, each dimension takes random values in the exploration domain.

DE uses two different vectors in the population to interfere with an existing vector and perform differential operations to achieve mutation.

$$\overrightarrow{V_i^g} = \overrightarrow{X_{r1}^{g-1}} + F(\overrightarrow{X_{r2}^{g-1}} - \overrightarrow{X_{r3}^{g-1}}) \tag{2.2}$$

Among them, *r1*, *r2* and *r3* are different individuals. The “scaling factor,” also referred to as the “differentiation” vector, is denoted as *F*.

During the evolution process, in order to ensure the validity of the solution, it is necessary to determine whether each component of the mutant individual satisfies the boundary conditions. If the boundary conditions are not met, the mutant individuals are regenerated randomly.

For each individual and the offspring mutation vector generated by it are crossed, specifically, for each component, the offspring mutation vector (otherwise it is the original vector) is selected with a certain probability to generate a test individual.

$$\overrightarrow{U_i^g} = [u_{i,1}^g, u_{i,2}^g, \cdots, u_{i,D}^g] \tag{2.3}$$

$$u_{i,d}^{g} = \begin{cases} v_{i,d}^{g}, & \text{if } r_d \le CR \text{ or } d = rn(i) \\ x_{i,d}^{g-1}, & \text{if } r_d > CR \text{ and } d \ne rn(i) \end{cases} \quad (2.4)$$

CR represents the "crossover probability" and *rn*(*i*) is a randomly selected integer from the range [1,*D*], with *rn*(*i*) being a random decimal within the interval [0, 1]. The use of *rn*(*i*) ensures that this crossover strategy can ensure that Ui has at least one component contributed by the corresponding component of Vi obtained by the mutation operator.

The flowchart of DE is as follows [45].

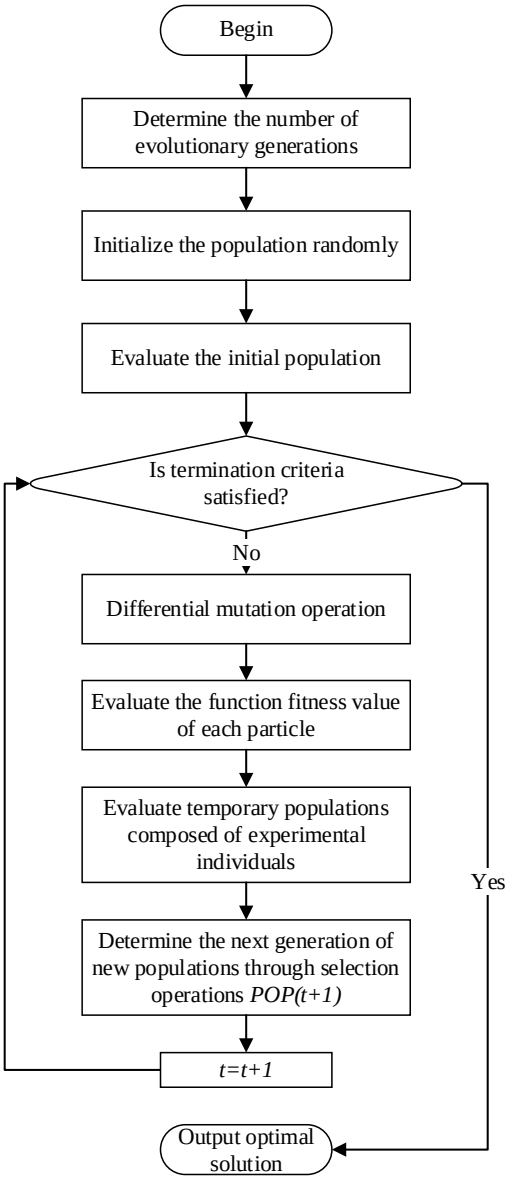


Figure 2.1 The flowchart of DE

DE algorithm finds applications across various domains, encompassing

optimization problem solving, machine learning, signal processing, image processing, engineering optimization, and economic modeling [45]. In the realm of optimization problem solving, the DE algorithm finds widespread application, addressing various optimization challenges such as function optimization and parameter tuning. Its remarkable global search capabilities and rapid convergence render it highly effective in navigating complex multidimensional spaces [46]. In the domain of machine learning, the DE is employed for tasks like feature selection and hyperparameter tuning. It is also used in training deep neural networks to search for suitable weights and parameters [47]. In signal processing, DE is applied to tasks such as filter design and signal denoising, showcasing advantages in handling complex signal scenarios [48]. The algorithm is harnessed for image processing tasks like segmentation and enhancement, allowing optimization of the image processing pipeline through parameter adjustments [49]. In engineering, DE is extensively used for process optimization and system design, aiding engineers in finding optimal solutions within complex systems [50].

2.1.2 Particle Swarm Optimization

In 1995, first introduced by Eberhart and Kennedy, PSO is a global search algorithm which belongs to SI [51]. The fundamental principle of the PSO algorithm involves individuals, referred to as particles, which can be analogized to birds or small fish. The evolution of each particle is guided by learning from both the historical best solution of itself and the collective global best solution of the group. This process mirrors a bird learning from its past best position and adapting based on the optimal position of the leading bird in the flock. In a way analogous to human decision-making, where individuals consider both personal experience and the experiences of others, PSO leverages two types of crucial information in its optimization process [51]-[52].

Unlike the genetic algorithm, PSO does not have the process of selection, crossover, and mutation operators, but solely progresses towards the global optimal solution by employing the speed update formula and position update formula continuously [51]. Hence, the operation of PSO is straightforward. The formula presented is as follows [51].

$$v_i^d = \omega \times v_i^d + c_1 \times rand_1^d \times (pBest_i^d - x_i^d) + c_2 \times rand_2^d \times (gBest^d - x_i^d) \quad (2.5)$$

$$x_i^d = x_i^d + v_i^d \quad (2.6)$$

In the formula, ω is the inertia weight. It is commonly initialized to 0.9 and subsequently reduces to 0.4 as the evolution progresses. c_1 and c_2 is the acceleration coefficient, which generally takes the value 2.0. $rand_1^d$ and $rand_2^d$ are a pair of random numbers within the range [0,1].

The research on PSO includes theoretical research, algorithm parameter research, topology research, hybrid algorithm research, and algorithm application research [53]. The PSO algorithm has a lot of applications, and the existing research can be divided into two categories, optimization and design applications, and scheduling and planning applications, and have achieved results in many industries. Regarding the application of PSO in engineering and system design, it includes optimization of neural networks [54]-[55], wing optimization design, and power system stabilizers. As for the application of PSO in scheduling and planning, it encompasses the traveling salesman problem [56], flow shop scheduling [57], and business planning [58].

2.1.3 Memetic Algorithm

When facing large-scale and complex optimization problems, traditional EC algorithms such as GA, PSO, and ACO often suffer from slow convergence speeds and difficulty in finding high-precision optimal solutions. Introducing local search methods can improve the solutions discovered by EC algorithms, enhancing both solution efficiency and accuracy. MA is a novel optimization technique that combines population-based EC algorithms with local search techniques [59]. The term “memetic” originates from the concept of “meme” mentioned by Oxford University scholar Dawkins in his book “The Selfish Gene” [60] published in 1976. In cultural evolution, similar to biological evolution, beneficial cultural genes, or memes, can be inherited and developed.

The framework of MA was proposed by Krasnogor and Smith, and it comprises nine elements [61].

$$\text{MA}=(P^0, \delta^0, off springSize, popSize, l, F, G, U, L) \quad (2.6)$$

P^0 represents the initial population, δ^0 denotes the initial parameter settings of the algorithm, $off springSize$ indicates the number of offspring generated through the production function G, $popSize$ represents the population size, l denotes the length of the encoding, F represents the fitness function, G denotes the generation function, U denotes the update function, and L is a set of local search strategies. It can be observed that compared to traditional EC algorithms, MA only adds an operation for local search.

Currently, significant progress has been made in the theoretical research of MA, including cooperative evolution-based MA [62]. In practical applications, MA has been employed in image processing [63], the traveling salesman problem [64] [65], business analytics, and data science [66]. Additionally, MA has been effectively utilized for real-world optimization problems like protein structure prediction [67], cellular mobile networks [69], data privacy [71], and cancer chemotherapy design [70].

2.2 Knowledge Transfer

2.2.1 Introduction of Knowledge Transfer

In the field of EC, the process of gaining experience from previously solved problems and applying relevant knowledge to new tasks or situations is commonly referred to as KT. In practical applications, problems rarely exist in isolation. Ignoring the search experience gained from related problems in previous optimization processes may lead to redundant searches on similar problems, resulting in unnecessary computational costs. Therefore, KT becomes crucial. For DOP, the experience and knowledge accumulated during the solution of problems within a specific time period are transferable and can be effectively applied to solve problems in other time periods. Similarly, for MTOP, the experience and knowledge gained while addressing one task may be transferable to addressing other tasks. However, existing EC algorithm solvers often start the search process from scratch, without considering the similarity between new and previous problems. Considering these factors, the ability of KT to generalize learned knowledge to other problems is highly significant for complex optimization problems. The integration of EC with KT holds great promise.

The concept of KT, also known as transfer learning (TL), has garnered earlier attention in the field of machine learning [72] [67] [73]. TL leverages knowledge obtained from domains with abundant high-quality training data to enhance learning models in target domains lacking sufficient training data [72], thereby avoiding costly data labeling efforts. Early transfer learning **is** primarily applied to tasks in machine learning domains such as speech recognition [74], computer vision [75], [80], natural language processing [76], indoor localization, face recognition, and training of deep learning models [77]. These tasks belong to traditional machine learning domains, including classification, regression, reinforcement learning, and deep learning [78] [79]. Research in machine learning has shown that KT can use knowledge learned from one problem and apply it to another, reducing the workload required to model from scratch. These findings provide insights for the application of KT in EC.

2.2.2 KT in EC

The combination of KT and EC is theoretically grounded and has garnered increasing attention from renowned scholars in recent years. Integrating EC solvers with KT across diverse domains aims to enhance algorithm performance, accelerate convergence speed, and improve optimization efficiency [27]. The flowchart of EC is depicted in Figure 2.2. Initially, a population is created, where each individual represents a potential solution. Next, Evolutionary operations are conducted iteratively until a satisfactory solution is found or termination conditions are met [81]. Throughout this process, it is observed that the population in evolutionary search contains crucial information for problem-solving. Useful features can be learned from the iteration of the population, and if applicable, KT across problems can guide the search to enhance optimization performance [27].

From the perspective of algorithm design, one application of KT in EC is to improve algorithm initialization through transfer learning. Information about model parameters or fitness functions learned from previous tasks can be used to initialize optimization algorithms, thereby enhancing their performance on new tasks. In EC algorithms like genetic programming, population initialization significantly influences

optimization outcomes. KT can be employed to design more informative initial populations, facilitating better exploration of the search space. In existing research, Ardeh et al. used genetic programming to transfer knowledge acquired from addressed old problems to tackle new problems [82]. Guo et al. adopted the theory of a knowledge pool for constructing a hybrid transfer strategy for generating new initial populations [83].

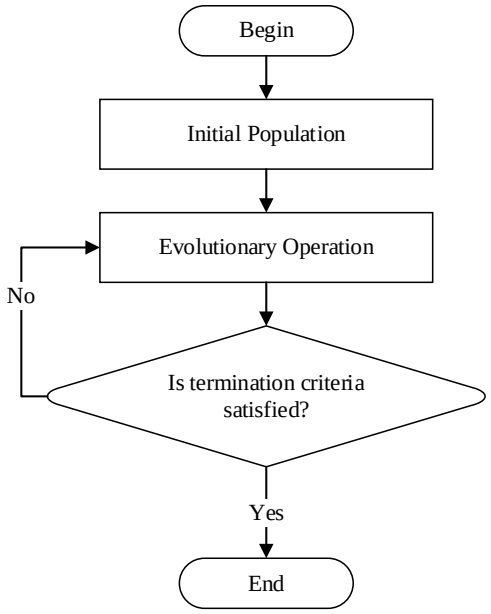


Figure 2.2 The flowchart of EC

In terms of integrating KT and EC algorithms, existing ETO methods can be classified into two types according to the search space of the problems being addressed: homogeneous ETO and heterogeneous ETO [27]. The former focuses on KT among problems with common search spaces, while the latter considers KT among problems with different search spaces, such as those with varying dimensions, decision variables, and objective functions [27].

According to the types of problems to be solved, KT and EC combined algorithms can be categorized into five classes: DOP, MTOP, multi-objective and many-objective optimization, expensive optimization, and algorithm applications [27]. This paper primarily focuses on DOP and MTOP, which belong to the first two categories among these five problems. DOP and MTOP problems are relatively complex and have been recent research hotspots. These two types of problems require algorithms to perform

well in the face of dynamic problem changes or when simultaneously handling multiple tasks. Solving such problems necessitates more advanced optimization algorithms and KT mechanisms. In practical terms, DOP involve changes in optimization problems over time, while MTOP involve optimizing multiple tasks simultaneously. Both of these problem types are relatively common in practical applications, hence researching these areas is practically significant for solving real-world problems.

Regarding KT in EC algorithms, research focuses on two questions at the execution level: when to perform KT and how to perform KT [22]. Firstly, determining when to perform KT is a crucial issue. The timing of KT during the execution of optimization algorithms can affect the performance and effectiveness of the algorithms. Determining when to perform KT may involve factors such as the dynamic characteristics, the quality of the current search state, and changes in the environment. Therefore, focusing on when to perform KT can help identify the most effective timing for KT. Secondly, how to perform KT involves specific operations and mechanisms, including knowledge representation, methods of KT, and selection of individuals to receive knowledge. Researching this issue requires considering the specific implementation of the algorithm and the nature of the problem. Focusing on how to perform KT can help design more effective KT strategies.

2.3 Dynamic Optimization Problems (DOP)

2.3.1 Introduction of DOP

Compared to static problems like finding function extrema, numerous optimization problems encountered in real-world scenarios exhibit dynamic characteristics and are subject to uncertainty [84]. These types of problems are commonly known as DOP. In DOP, objective functions, constraints, Pareto fronts, etc., may change over time. DOP is more challenging than static optimization because the same problem must be repeatedly optimized over time [85]. Compared with static problems such as solving function extreme values, many real-world optimization problems in uncertain environments are dynamically changing [84], and such problems are called DOP. In DOP, the objective function, constraints, Pareto front, etc. may all change over time.

DOP is more difficult than static optimization because the same problem has to be optimized repeatedly over time [85].

Here is an example of a DOP. Picture a scenario in supply chain management where a company manufactures goods and distributes them to retailers. The primary objective of this system is to minimize overall transportation expenses. Initially (at time t=0), the routes and transportation costs are established. However, due to factors like fluctuating fuel prices and variable traffic conditions, these costs may change over time. At each time step, the system has the flexibility to adjust transportation routes to accommodate these evolving cost scenarios. The objective of the DOP is to choose the most cost-efficient route at each time interval to minimize the total transportation expenses. This necessitates making real-time decisions in response to changing circumstances, ensuring adaptability to environmental shifts while consistently aiming to minimize costs.

DOP is an optimization problem in which the fitness function, constraints, and environment parameters may all change over time. The objective function of DOP is expressed as:

$$\max f(X,e) = f(x_1, x_2, ..., x_D, e) \tag{2.6}$$

In this function, f represents the objective function, X denotes a decision vector with D dimensions, e signifies the evolving environment, the range of each dimension x_j is defined within the interval.

The challenge in DOP lies in the need to find the optimal solution (or an approximate one) within acceptable time and cost constraints. This requirement translates to the algorithm being able to quickly converge to the global optimum after each environmental change, without getting trapped by previous optimal solutions and falling into local optima [85].

EC is commonly employed for static problems, typically aiming to rapidly converge the population to the global optimum. However, this poses challenges in dynamic environments, where the peaks of the objective function constantly fluctuate, necessitating continuous tracking of the optimal values. When using traditional

evolutionary algorithms, once convergence is achieved, the diversity of the population diminishes, making it difficult to track new optimization targets. Conversely, excessive diversity may also hinder algorithm performance. Moreover, during reiteration, it is crucial to use KT to track new optimization targets. If the objective functions before and after the change are similar, it is advisable to retain some historical information. However, preserving too much information may lead to premature convergence. For DOP, scholars like Zhan classified existing EC research into three categories based on functionality: Decreasing the complexity of the problem, enhancing algorithmic variety, and speeding up convergence are essential objectives [25].

When simplifying problem difficulty, two approaches are commonly employed: decomposing dimensions into teams and segmenting the search area into sections. The concept of decomposing dimensions into teams originates from Cooperative Coevolution (CC), initially employed for solving evolutionary optimization problems of large scale [86]. For DOP, CC can partition the dimensionality of the search area, evolving various dimensions to locate and track the optimal solution [87]. Specifically, in cooperative coevolution, the solutions to a problem are divided into different parts or groups, with each part being independently optimized by a separate subpopulation. This cooperative evolution approach allows the algorithm to handle high-dimensional, complex problems more efficiently, as different parts can be optimized independently, thus enhancing the search efficiency. Additionally, some studies have employed competitive-cooperative CC algorithms, where each species subpopulation competes to represent specific subcomponents of the multi-objective problem, and the eventual winners cooperate to evolve better solutions [88]. Segmenting the search space into pieces involves the use of cellular automata methods. Cellular Automaton is a mathematical model composed of a set of identical automata (referred to as “cells”) that evolve at discrete time steps according to predefined rules. Each cell can be in one of a finite number of states, and the rules determine how the cell's state evolves based on its state and the states of its neighboring cells. Based on this model, Hashemi et al. proposed Cellular PSO [89]. Noroozi et al. introduced CellularDE, utilizing the same

cellular automaton framework to address DOP [90], while Sharifi and Noroozi proposed a two-stage Cellular PSO [92]. In summary, these methodologies offer effective strategies for addressing the complexities of optimization problems, particularly in dynamic environments. They provide insights into enhancing search efficiency and facilitating the exploration of high-dimensional solution spaces.

Various methods have been introduced to enhance algorithm diversity, including the utilization of multi-populations, the creation of composite solutions, and the development of innovative solution update strategies [25]. Among these methods, research on multi-population approaches is the most extensive. In the evolutionary process, each subpopulation is responsible for an independent task, akin to students in a class forming groups to complete different assignments. Multi-population models can be divided into two categories: homogeneous and heterogeneous models. In homogeneous models, each population has the same task, while in heterogeneous models, multiple populations are situated at different levels or have different tasks. To continue with the classroom analogy, a homogeneous model is where every group of students has the same task, though they may adopt different approaches. In contrast, a heterogeneous model is where different groups of students may have different specializations or skills, and each group is responsible for solving an independent problem. In homogeneous algorithms, many existing studies employ clustering algorithms to create subpopulations, including hierarchical clustering [93], a combination of random immigration strategies and hierarchical clustering [94], competitive clustering [95], K-means clustering [96], adaptive multi-population approaches [97], distributed multi-populations [98], and clustering-based clone selection algorithms [99]. For heterogeneous models, the methods employed in existing research are more diverse. Branke et al. proposed the SOS algorithm, and Li et al. introduced FPSO, both of which include a parent population and multiple subpopulations, with the parent population conducting global search and the subpopulations performing local search [100][101]. In recent years, a heterogeneous model called distributed multiple populations (DMP) has been proposed. DMP employs

six strategies designed at three levels (i.e., population level, subpopulation level, and individual level) to address different types of DOP. Diversity preservation at the individual and population levels accelerates the entire population's response to new landscapes, while elite self-learning of individuals at the subpopulation level promotes the development of promising areas [102][103].

To expedite convergence speed, methods involving the direct reuse of historical solutions and the prediction of promising solutions can be applied [25]. Regarding the reuse of historical solutions, the difference lies in the methods of archiving historical information, which include direct reuse [104], fine-grained archiving and coarse-grained archiving [105], and direct incorporation into the new initialized population [106]. However, historical optimal solutions may struggle to adjust to alternations in the new environment, making methods derived from predicting promising solutions in the new environment more advantageous. Existing research includes variable relocation strategy (VRS) [107], adaptive PSO with VRS [108], and orthogonal learning particle swarm optimization with VRS [109], combining population prediction strategies based on prediction centers and estimated manifolds [110], neural network-based change prediction methods [111], and neural network information transfer [112].

“The only absolute motion is the motion of change; the only constant is change itself. [113]” DOP find widespread applications in various fields owing to the time-varying characteristics of real-world optimization problems. DOP research extensively conducted in domains such as logistics and transportation [114], power systems [115], financial markets [116], manufacturing [117], and unmanned systems including autonomous vehicles [118], ships [119], and drones [120]. With the development of economy and technology, the emergence of DOP research in more domains such as greenhouse control in agriculture is observed [121], electric vehicles [122], healthcare [124], environmental monitoring [125], and telecommunication networks [126].

2.3.2 KT in DOP

In real-world problems, DOP typically exhibit two key characteristics. On one hand, while DOP is inherently dynamic, the nature of the problem remains relatively static

within a specific period without significant changes. During this time frame, DOP can be treated as a static problem. On the other hand, adjacent static time periods should exhibit some degree of similarity in the characteristics and features of the problem. This similarity allows for partial reuse of information from previous environments when addressing DOP across these time periods, without the need to start optimization from scratch [106]. Therefore, existing research on DOP aims to closely track changes in time by approximating Pareto optimal solutions as closely as possible when the environment changes [127]. This involves reusing historical information, which, as mentioned in the previous section, can be categorized into direct reuse of historical information and prediction of promising solutions based on historical solutions. The primary purpose of reusing historical information is to expedite convergence speed or reduce runtime by leveraging past experiences. Utilizing past experiences can help us solve new problems in dynamic environments more efficiently, which is the primary focus of most current research efforts. Additionally, leveraging KT can reduce problem complexity and increase algorithm diversity.

Firstly, KT can contribute to reducing the difficulty of DOP by decomposing dimensions into teams and segmenting the search area into pieces. Currently, there is limited research from this perspective. For the first approach, decomposing the dimensions of the problem into groups involves grouping relevant decision variables together to form subproblems. This can be based on dependencies between variables, functional properties, or other correlations. If effective optimization strategies or search directions for specific variable combinations were learned from previous problem instances, this knowledge can be transferred and applied to similar variable combinations in the current problem. The research of Rakitianskaia did not explicitly mention KT, but it introduced the concept of context vectors, which can be viewed as a form of information transfer [87]. Liu et al. used probability distribution functions to adjust the relationships between variables and groups, making the understanding of variable dependencies more flexible [105]. For the second approach, segmenting the search space into pieces involves dividing the entire search space into non-overlapping

regions, each of which can be considered a local subproblem. This segmentation can be derived from the attributes of the problem, constraints, or other distinguishing properties. In this case, KT involves applying knowledge gained from previously solved problems, such as effective regions in the search space or information about related subproblems, to similar regions in the current problem. KT facilitates improved algorithm understanding of the problem space and enhances search efficiency by sharing information between similar regions. Noroozi's research did not explicitly mention KT either, but it involved utilizing local information from different regions [90]. Through these two approaches, KT can help algorithms better understand the structure and characteristics of the problem, reduce the complexity of the exploration domain, and thus decrease the difficulty of decision optimization problems. This decomposition and segmentation strategy aids in improving the local search effectiveness of algorithms, allowing them to focus more on smaller problem domains and thereby increasing the efficiency of problem-solving. Due to the limited research from this perspective, it represents a potential research direction for future exploration. Additionally, Jin et al. discussed three types of knowledge transfer methods in data-driven evolutionary optimization, including semi-supervised learning, parameter sharing and domain adaptation, and transfer optimization [91]. These methods help in reducing the difficulty of data-driven optimization by effectively leveraging knowledge from various sources.

Secondly, KT can enhance algorithm diversity in DOP through various methods. Firstly, the most common approach is the utilization of multi-population methods, which involves transferring knowledge between different subpopulations. Each subpopulation may focus on distinct regions of the search space or tackle different types of problems, and by sharing knowledge among them, the overall diversity of the algorithm can be increased [106]. The second method involves introducing heterogeneous knowledge sources. Leveraging knowledge from different problem instances, domains, or algorithms can enhance algorithm heterogeneity. Introducing heterogeneous knowledge may include rules or heuristic information from other

optimization algorithms or problem domains, thereby enriching the algorithm's search strategies. For instance, Zhou et al. attempted to learn structured knowledge obtained from early time slots and apply it to dynamic vehicle routing problems [128]. Wu et al., within a multi-population framework, introduced a certain degree of heterogeneity based on the different properties of each subpopulation to generate multiple subpopulations balanced in new environments [106]. Yan et al. generated new subpopulations by learning from the final populations of adjacent environments and extracting patterns of dynamic environmental changes from high-quality solutions in historical environments [129]. The third approach involves introducing randomness and perturbation. Introducing randomness and perturbation is a classical method for increasing algorithm diversity. By introducing a moderate amount of randomness or perturbation during the search process, algorithms can avoid local optima and explore the problem domain more comprehensively. Sun et al. used a random perturbation approach to solve missile trajectory optimization problems [130]. Additionally, introducing randomness when selecting individuals or subpopulations for reproduction or search can ensure that the algorithm does not overly focus on a specific region of the search domain, thus enhancing algorithm diversity. Lastly, dynamic parameter adjustment involves dynamically adjusting algorithm parameters using KT. Parameter settings learned from previous problem instances may not be applicable to new problems, so dynamically adjusting parameters through KT can better adapt the algorithm to the characteristics of the current problem. Zhan et al. employed adaptive parameter control in the APSO algorithm, meaning that the parameters of algorithm can be adjusted according to optimization progress or environmental changes [108].

Finally, KT accelerates the convergence rate and reduces the execution time of DOP. In existing research, enhancing algorithm variety and speeding up convergence rate often appear in the same study. For instance, Jiang et al. introduced the concept of the Knee point, which cleverly integrates a small number of high-quality individuals and imbalanced transfer learning techniques [131], thereby increasing algorithm diversity while also speeding up convergence. According to the two assumptions of DOP,

continuous dynamic environments often exhibit significant correlations with one another, and reusing historical solutions holds significant potential for expediting convergence in novel environments. For instance, Jiang et al. reused past experiences to generate an effective initial population pool [132]. There are two approaches to historical solution reuse: directly utilizing historical solutions and predicting the optimal solution's location, both of which are mentioned in early survey articles on DOP. In practical applications, the more commonly used method may rely on the specific nature of the problem, algorithm design, and researchers' preferences. Sometimes, directly reusing historical solutions is straightforward, while predicting the optimal solution's location may require more complex models and algorithms. For direct reuse of historical optimal solutions, in Cao et al.'s study, the best solution in each generation is stored, and when environmental changes are detected, historical solutions are retrieved to collaborate with newly generated solutions to adapt to the new environment [104]. Methods based on historical information for predicting the optimal solution have received more research attention, such as variable relocation strategies [107]-[108]. In specific operations, Rang et al. did not use a linear prediction model but employed Long Short-Term Memory networks for prediction [127]. Wu et al. proposed the Archive-based Historical Information Reuse (AHIR) strategy [106]. Guo et al. introduced a subspace alignment method for KT [83]. Hatzakis et al. inferred the estimated value of the next position using a prediction model created from the sequence of previous optimal solution positions [133]. Jiang et al. used manifold transfer learning for prediction [136]. Additionally, Liu and Wang proposed an enhanced population prediction strategy for dynamic multi-objective optimization algorithms utilizing transfer learning, aiming to effectively track optimal solutions in dynamic environments by integrating historical information [134]. It is worth noting that if the prediction model is inaccurate, it may have a negative impact on the optimization process. Ruan et al. pointed out that prediction models based on incorrect assumptions may lead to inaccurate predictions of the optimal solution [87]. Ma et al. proposed a higher order knowledge transfer strategy for dynamic community detection, aiming to retain and

transfer advantages from previous snapshots to subsequent ones [135].

In DOP, another key challenge lies in determining when and how to conduct transfer learning effectively [137]. Similar issues are also present in MTOP [22]. Transfer learning proves to be effective in addressing fixed POS problems and scenarios with minor environmental changes [137]. However, some problems are not suitable for transfer learning. Therefore, when dealing with issues where transfer learning fails, it is advisable to avoid its usage.

2.4 Multi-task Optimization Problems (MTOP)

2.4.1 Introduction of MTOP

A significant distinction between humans and machine learning lies in humans' capability for multitasking. During the learning process, humans can use knowledge acquired in one task to aid in learning another task [21]. For instance, for a high school student, improvement in mathematics performance may also contribute to learning physics. Many outstanding students excel in all subjects precisely because they can apply learning experiences from one subject to others. In complex optimization problems, EC has been used to address various optimization challenges, but it faces two key obstacles: heavy computational burden and poor generalization ability [21]. Inspired by human learning, a better strategy to address these challenges is MTOP. Compared to single-task learning, multi-task learning offers several advantages: multiple tasks share one model, reducing memory usage; enhanced convergence speed and reduced learning difficulty; performance improvement in associated tasks through shared information and KT [22]. Additionally, MTOP can also address scenarios with insufficient data sources [21]. In the real world, insufficient training data for individual tasks is common, making mutual learning between different tasks meaningful and valuable [138].

Primarily focusing on the integration of EC and MTOP because EC exhibits implicit parallelism, allowing for the simultaneous optimization of multiple tasks when solving MTOP. Evolutionary computing algorithms can perform selection, crossover, and mutation operations concurrently when handling multiple tasks. These operations

do not interfere with each other when dealing with different tasks, enabling parallel execution. By leveraging the parallelism between tasks, the optimization process can be accelerated, enhancing the efficiency of the algorithm. This parallelism is implicit, as multiple tasks can be processed simultaneously without the explicit use of parallel computing techniques. Therefore, the combination of MTOP and EC has become a research hotspot in recent years, with review papers by prominent scholars providing insights into research progress [27] [141].

Existing MTOP can be classified into single population and multiple population strategies.

The most representative algorithm for single-population MTOP is the multifactorial evolutionary algorithm (MFEA) [30]. MFEA is a single-population MTOP algorithm that uses a single population to simultaneously optimize multiple problems. The core idea of MFEA is to apply a single population to solve multiple related tasks and promote information sharing and individual adaptability across tasks through controlling mating intensity and implementing skill inheritance. This makes MFEA an effective method for addressing MTOP.

In multi-population MTOP algorithms, each population corresponds to an optimization task. Throughout the evolution process, populations can engage in two distinct operations: self-evolution and inter-task evolution. Self-evolution means that individuals in the population only use parents from the same population to generate offspring. In other words, individuals undergo genetic operations only within the current population. Inter-task evolution refers to the population using parents from the same population and randomly selected parents from other populations to generate offspring for the task. In inter-task evolution, operators and solutions generated by parents can represent the shared information between different populations. This means that individuals can leverage information from other tasks to generate offspring. The determination of whether to conduct self-evolution or inter-task evolution is based on a random parameter called the random mating probability (*rmp*), which is a control parameter utilized to determine the probability of each population selecting self-

evolution or inter-task evolution. This multi-population framework aims to allow individuals to exchange information between different tasks and enhance the efficacy of multi-task optimization through inter-task evolution. Compared to single-population MTOP, there are more research achievements in multi-population optimization. Chen et al. adopted an adaptive archive mechanism to determine which task similar to the current task can provide the most useful assistance [142]. Huang et al. used surrogate-assisted strategies to minimize the quantity of fitness evaluations [143]. They also developed a surrogate-assisted MA model using DE as the global search component and the Gaussian process as the surrogate model [144]. Wei et al. addressed the challenge of multi-class classification problems in multi-task optimization using gene expression programming GEP [145]. In traditional GEP methods, an M-class classification tasks is regarded as M independent binary classification tasks without considering the correlation between classes. This approach may lead to output conflicts because different binary classifiers may give inconsistent class labels. Therefore, traditional GEP methods may perform poorly in handling multi-class classification problems. By introducing the evolutionary multi-task optimization paradigm, this method allows interaction and KT between different binary classifiers to address the problem of output conflicts. Additionally, some scholars have combined multi-task optimization with multi-objective optimization [32] and dynamic optimization [146] as composite optimization problems.

2.4.2 KT in MTOP

The inspiration behind MTOP comes from the human ability to simultaneously perform multiple tasks and the mature concept of multi-task learning in predictive analytics. By applying this ability to optimization problems, MTOP can consider multiple tasks simultaneously in a single optimization process, thereby improving the efficiency and performance of the search. Therefore, a key concept in MTOP is inter-task KT. During the evolutionary optimization process, useful knowledge transferred across tasks can lead to the automatic resolution of related problems. This means that solving one task may positively impact the optimization process of other tasks,

enhancing the search capability. Early applications of KT-inspired solutions to MTOP problems were unidirectional [28], [29]. Subsequently, Gupta et al. proposed MFEA, which was inspired by the biocultural model of multifactorial genetics, explaining how genetic and cultural factors interact to pass on complex developmental traits to offspring [30]. The cultural factor mentioned in the model refers to KT.

Research on MTOP has proliferated in the years following MFEA, with studies emerging one after another [147]-[159]. These studies, originating from the perspective of KT, delve into how evolutionary algorithms can be used to address multiple independent tasks. Some papers propose new evolutionary algorithm paradigms, such as generalized multi-task optimization and multifactorial genetics, which enhance optimization by transferring knowledge across tasks [147],[151],[152]. Others focus on improving and applying evolutionary multi-task algorithms, with some methods employing online parameter estimation for KT [159]. Furthermore, some research explores the application of cross-domain optimization and resource allocation strategies in KT [148],[146],[153]. Collectively, these studies provide important theoretical and practical foundations for the development and application of evolutionary multi-task optimization.

However, as research into Evolutionary Multi-Task Optimization (EMTO) deepens, the issue of negative transfer across tasks has become increasingly prominent, posing a common challenge in current studies. Studies have shown that KT between tasks with low relevance may even lead to a decrease in optimization performance [162], underscoring the critical importance of effective inter-task KT for EMTO. Therefore, addressing the problem of negative transfer across tasks is key to ensuring the success of EMTO. Mitigating the impact of the negative transfer on EMTO primarily requires consideration of two aspects: first, identifying the appropriate tasks for KT, and second, improving methods to elicit more useful knowledge during the transfer process.

Regarding the identification of tasks suitable for KT, the MFEA-II algorithm proposed by Bali et al. uses online transfer parameter estimation to dynamically adjust the KT probability between tasks, increasing KT among highly correlated tasks and

mitigating the influence of negative transfer [159]. In the study by Yang et al. [163], a two-stage pairing method is employed, considering the similarity between tasks to ensure that KT only occurs between tasks with high relevance, thereby reducing the likelihood of negative transfer. Similarly, Liaw et al.'s eco-symbiotic-based evolutionary multi-task approach [151] adopts a similar strategy by modeling the correlation between tasks and selectively transferring knowledge to mitigate the influence of negative transfer. Regarding improving the KT process to extract more useful knowledge, the MFEA-II algorithm [159] introduces online transfer parameter estimation to dynamically optimize the KT process by adjusting the probability of KT to ensure more useful knowledge is transferred among highly correlated tasks. On the other hand, the studies by Yang et al. [163] and Liaw and Ting [151] leverage the similarity between tasks and task characteristics to optimize the selection and crossover methods of transferred individuals and construct task mappings, thereby extracting more useful knowledge and mitigating the influence of negative transfer.

Existing research on MTOP has been summarized by scholars. Tan et al. provides a comprehensive introduction to evolutionary transfer optimization, reviewing various categories of optimization problems such as uncertain environments, multi-objective optimization, etc. [164]. Xu et al. reviews the research progress in MTOP over the past five years. The article examines various techniques, including chromosome encoding and decoding, intra-population reproduction, inter-population reproduction, as well as evaluation and selection methods. [141]. Osaba and Wei et al. provides a systematic analysis and summary of evolutionary multi-task optimization methods [165], [166]. Gupta et al. discusses six case studies of evolutionary multi-task processing in practical applications, emphasizing its practical applications and effects in various fields. The article showcases the potential and value of evolutionary multi-task processing through case analysis [167].

In these two years, there have been some novel research in KT within MTOP. Jiang et al. proposed a evolutionary algorithm based on knowledge structure preservation, which extracts useful structure-preserved knowledge from similar source tasks [154].

Wang et al. discovered that although many explicit transfer strategies have been developed to enhance positive transfer between optimization tasks, most of these methods achieve knowledge transfer by migrating the best solutions from the source task to the target task, neglecting the proper use of information from the target task in solution selection [155]. To address this issue, they proposed a lower confidence bound solution selection method based on evolutionary multitasking optimization [155]. Additionally, Lin et al. integrated various domain adaptation methods for knowledge transfer in EMT [156]. In terms of applications in evolutionary multitasking optimization, Zhou et al. proposed an evolutionary multitask convolutional neural architecture search framework [157]. Feng et al. used multitasking approaches to solve multi-objective high-dimensional feature selection problems [158].

These studies indicate significant progress in methods for knowledge transfer between optimization tasks and demonstrate their potential applications in various fields.

CHAPTER 3

HISTORICAL INFORMATION-BASED DIFFERENTIAL EVOLUTION FOR DYNAMIC OPTIMIZATION PROBLEM

3.1 Introduction

In the fast-paced and ever-changing world, DOP play a crucial role. DOP refer to optimization problems where the objective function, constraints, and environmental parameters change over time. Such problems are widely encountered in various domains, including intelligent traffic management [168], Internet of Things [170], operations management [171], logistics [114], power systems [115], financial markets [116], manufacturing [117], and unmanned systems such as autonomous cars [118], ships [119], and drones [120]. Research in these areas has been extensive. With economic and technological developments, the emergence of DOP studies in more fields, such as greenhouse control in agriculture, is being witnessed [121], electric vehicles [122] [123], healthcare [124], environmental monitoring [125], and telecommunications networks [126].

Unlike static optimization problems, the uncertainty and variability of dynamic environments pose challenges to traditional optimization methods. Static optimization algorithms often assume that problem parameters and constraints remain static, making them unsuitable for direct application in dynamic environments. However, in practical applications, changes in environmental parameters can render static optimization solutions ineffective, thereby affecting system performance and efficiency. For instance, in intelligent traffic management systems, parameters such as traffic flow, road conditions, and vehicle destinations frequently change. Failure to promptly adapt traffic signal optimization schemes to these changes can result in traffic congestion, energy wastage, and reduced travel efficiency [168].

DOP faces various challenges due to the dynamic nature of the objective function, constraints, and environmental parameters over time, leading to the dynamicity of the

search space, the failure of existing solutions, and the increased complexity of the problem. For example, optimizing traffic signal lights involves fluctuations in traffic and pedestrian flow [168]. DOP is inherently complex and multimodal because it exhibits shifting peaks, where local optima at different times can lead to the problem easily getting trapped in local optima [165] [169]. Furthermore, DOP demands algorithms with high requirements, necessitating strong robustness to cope with environmental changes, noise interference, and uncertainty. Lastly, DOP also faces challenges similar to expensive optimization problems because it often requires frequent updates and adjustments to solutions, which can result in higher computational costs.

Evolutionary computation (EC) was initially developed for solving static optimization problems, with traditional EC algorithms such as Genetic Algorithm (GA) [181], Differential Evolution (DE) [44], and Particle Swarm Optimization (PSO) [45] originally designed for such static optimization problems. However, as attention to optimization problems in dynamic environments has increased, researchers have begun applying EC algorithms to solve DOP. EC has certain advantages in addressing DOP, primarily in adaptability and robustness [25], parallel computing [44], distributed computing [102], diversity and exploratory behavior [168], parameter adaptability [172], and reuse of historical information [106].

This thesis primarily focuses on addressing the challenges posed by the constantly changing problem distributions in DOP using EC, aiming to alleviate the difficulties associated with such problems. While DOP present significant challenges for optimization, careful observation of real-world instances reveals three key characteristics. Firstly, despite the dynamic nature of the problems, they often exhibit stability periods, allowing them to be temporarily treated as static problems within certain time intervals. For example, stability around specific values can be observed within a time range. Another example is the relative stability of variations within specific time periods. The second characteristic is that DOP can be decomposed into multiple static optimization problems, as changes between two static environments are

not excessively drastic. These two characteristics provide an opportunity to enhance search efficiency by utilizing historical information. For instance, traffic signal control problems under different traffic volumes remain similar [168], enabling the utilization of historical data. Given these characteristics, the focus of the research is on effectively leveraging historical information from past environments in the context of new environments [169]. To address this issue, the Historical Information-based DE (HIDE) is proposed. This chapter introduces a Region-based Subpopulation Initialization (RSI) method to create balanced subpopulations in new environments by initializing subpopulations in various areas of the exploration space to enhance population variety. Additionally, an Archive-based Historical Information Reuse (AHIR) method is proposed to use previously discovered peaks as historical information to aid in tracking and discovering new peaks.

The chapter follows this structure. Section 3.2 provides an overview of relevant research on DOP. Next, Section 3.3 introduces the methodology and research framework of the HIDE algorithm. Section 3.4 presents comparative experiments and experimental data, and Section 3.5 offers conclusions and prospects.

3.2 Related Work

3.2.1 Dynamic Optimization Problem

DOP distributions are inherently complex and multimodal, with the dynamic nature and multimodality of DOP often being correlated. This correlation arises because changes in dynamic environments may increase the diversity of the objective function, leading to the existence of multiple local optima in the solution space. Consequently, most EC methods still face the challenge of falling into local optima, even global optimization algorithms like GA, ACO, PSO, and DE algorithms. When addressing DOP, several approaches are worth considering: modeling dynamic environments [24], adaptive parameter tuning [96], [108], employing multi-population strategies [173], [96], [97], reusing historical information [171], [176], [108], and parallel and distributed computing [102]. Zhan et al. mentioned in their review of complex optimization problems that reducing problem complexity, increasing algorithm

diversity, and accelerating convergence speed are effective strategies, which are also applicable to DOP. However, categorizing existing research based on these strategies poses difficulties because many novel algorithms simultaneously increase algorithm diversity and convergence speed. In an earlier review, Thanh et al. mentioned common methods for solving DOP, including introducing diversity at the onset of change, maintaining diversity during the search process, memory-based methods, prediction-based methods, adaptive methods, and multi-population methods [24]. This classification is based on methods mentioned in existing research but does not consider the interrelationships between these six methods, such as introducing diversity at the onset of change and maintaining diversity during the search process, both of which increase diversity, nor does it consider further subdivisions of methods. Therefore, integrating these two reviews to classify existing EC methods for solving DOP. Subsequently, elaborating on multi-population methods in detail.

The first method involves introducing diversity, including static and dynamic diversity introduction. Static diversity introduction entails randomly initializing populations, while dynamic diversity introduction maintains diversity during the search process, such as through random perturbation, crossover swapping, and other diversity maintenance strategies [179]. The second method involves reusing historical information, also known as memory-based methods, such as archiving strategies [171], [96], and knowledge pools [178], to store the historical best solutions. Additionally, memory can be explicit or implicit. Explicit memory involves explicitly storing past information through a mechanism for direct access and utilization in the future. Implicit memory refers to the algorithm's ability to adapt to environmental changes through its evolutionary process or behavior without the need for explicit storage of past information. The third method involves predicting environmental changes using models such as neural networks [176]. The fourth method involves adaptive parameter tuning. The fifth method involves using multi-population strategies, including homogeneous and heterogeneous populations [103]. Chapter 2 has provided detailed explanations of multi-population methods. In this chapter, the second and fifth methods are primarily

adopted to address the problems.

3.2.2 Differential Evolution

DE is an algorithm in EC that uses differences among individuals to guide evolution. It integrates the concepts of crossover and mutation from GA while also drawing inspiration from the learning aspects of PSO [44]. Compared to other evolutionary algorithms, DE maintains a population-based global search strategy, employing a simple mutation operation based on differences in real-number coding [44]. Additionally, DE adopts a one-to-one competitive survival strategy, simplifying the complexity of genetic operations. DE possesses unique memory capabilities, dynamically tracking the current search status and adjusting search strategies as needed. Due to its strong global convergence ability and robustness, DE is suitable for addressing various kinds of conventional optimization problems, including DOP [90], without relying on specific problem features. A detailed introduction to DE is provided in Section 2.1.1.

3.2.3 Multi-population Methods

In the field of EC, multi-population methods are common strategies used to enhance algorithm performance and efficiency. These methods decompose the entire optimization process into the evolution of multiple subpopulations, akin to dividing a class of students into smaller groups. Each subpopulation can evolve independently and periodically share information or migrate individuals to facilitate global search and avoid local optima [177]. This approach offers two advantages. Firstly, multi-population methods divide the entire population into several subpopulations, each with potentially different characteristics, parameter settings, or evolution strategies. These subpopulations can evolve independently or periodically exchange information or individuals to promote global search and prevent premature convergence. Secondly, in multi-population methods, information or individuals are often exchanged regularly between subpopulations to enhance diversity and global search. Forms of information sharing may include individual migration, solution exchange, and parameter adjustment, aimed at accelerating global search and improving algorithm robustness. In practice,

multi-population methods also involve considerations of population homogeneity and heterogeneity, as detailed in Chapter 2. Additionally, some multi-population methods are adaptive, dynamically adjusting the number, size, or parameter settings of subpopulations based on problem characteristics or search progress. This adaptive capability enables better adaptation to different problem domains and search environments, enhancing algorithm robustness and adaptability [171], [96].

As a popular algorithm in the field of Evolutionary Computation (EC) in recent years, some researchers have explored the application of the Differential Evolution (DE) algorithm and employed the multi-population approach to address Dynamic Optimization Problems (DOP) [174]. The CEC 2009 benchmark set has been used as a dynamic testing problem set to evaluate the performance of evolutionary computation in dynamic environments, with Li et al. introducing a General Dynamic Benchmark Generator (GDBG) to construct dynamic environments across three solution spaces [174]. Thus, many existing studies use the CEC 2009 benchmark set as the test problem set. Existing research on DOP solutions can be roughly categorized into several approaches: adaptive parameters [172], [173], history-based [175], prediction-based [176], and parallel and distributed methods [102]. In terms of adaptive parameters, Brest et al. proposed an adaptive control parameter setting method to dynamically adjust control parameters related to differential evolution [172]. Based on this method, three years later, Brest et al. applied an adaptive control parameter multi-population differential evolution algorithm to solve DOP [173]. The algorithm demonstrated strong performance on CEC 2009 dynamic optimization benchmark functions. Mendes et al. proposed a multi-population DE algorithm, DynDE, tailored for DOP without requiring F or CR parameters [175]. Experimental evidence supports the effectiveness of this algorithm in solving dynamic peak benchmark functions. In history-based methods, Halder et al. employed a multi-population approach and proposed a Cluster-based Dynamic DE with an External Archive algorithm [96]. This method divides the entire population into clusters based on the spatial location of the experimental solution, allowing for the sharing of local information during the optimization process. In

prediction-based methods, Liu et al. suggested that when a new environment is closely related to the previous one, transferring information can accelerate the acquisition of high-quality solutions in the new environment. Thus, they proposed a neural network-based information transfer method [176]. In parallel and distributed approaches, Zhan et al. introduced a two-layer heterogeneous differential evolution algorithm in a cloud computing distributed environment [102]. This method, called Cloudde, facilitates simultaneous operation and adaptive migration of parameters or operators in different populations. It utilizes Message Passing Interface MPI technology to achieve distributed computing by sending different populations to different slave processes. Each slave process performs mutation and crossover operations with different evolution strategies independently during the evolutionary process. Subsequently, these slave processes return the results to the master process, which performs migration operations based on adaptive probability to facilitate information exchange and population evolution. It can be seen that Cloudde also adopts an adaptive approach. Furthermore, Li and Zhan et al. used Cloudde to solve cloud-based DOP [103].

The multi-population approach can be viewed in part as diversity maintenance. At the same time, the multi-population approach also involves memory and adaptability. The multi-population method is to divide a large population into several small sub-populations, and each sub-population performs its duties and evolves independently. There are mainly two types of multi-population methods. In the first type, subpopulations are assigned different tasks, some subpopulations are responsible for finding the global optimal solution, and some subpopulations focus on tracking changes in the environment. These two subpopulations can share information and cooperate to guide the population to evolve better. A co-evolutionary algorithm based on the PSO algorithm and DE algorithm (CESO) is proposed by Lung et al. in [180]. In the CESO algorithm, crowding DE, a variant of the DE algorithm, is used to maintain the diversity of the population and avoid premature convergence. The PSO algorithm is used as a local search operator for fast convergence. The CESO algorithm balances exploration and exploitation and achieves good results. The second type, which is the focus of this

chapter, uses multiple homogeneous populations to locate and track distinct peaks. In this multi-population approach, the entire search space is divided into different regions. Each region may contain one or more peaks, and each subpopulation is responsible for searching a given region to find the optimal solution to achieve a global search. In [93], Yang et al. A clustering-based method of clustering particle swarm optimization CPSO to divide a large population into multiple subpopulations is proposed. However, partitioning leads to uneven distribution of subpopulations. When insufficient computing resources are allocated, some regions may not be fully utilized. Therefore, a more balanced approach to population division is needed.

3.3 Framework of HIDE Method

3.3.1 RSI Strategy (region-based subpopulation initialization)

In the context of multi-population methods, the quality of population partitioning is crucial as it directly impacts the performance and efficiency of the algorithm. A good partitioning scheme possesses the following characteristics: balance, diversity, adaptability, and coverage of the entire search space. Addressing this issue, the Region-based Subpopulation Initialization (RSI) strategy is proposed for initializing and generating subpopulations in new environments. As mentioned earlier, the problem arises when subpopulation partitioning is uneven, leading to certain regions being underused due to insufficient computational resources. However, existing partitioning methods often result in imbalanced population sizes. For instance, clustering-based partitioning depends on the distribution of individuals in the search space, leading to some subpopulations being disproportionately large while others are too small. This issue becomes more severe when the partitioned population sizes are too small, as some promising regions may not be thoroughly explored and evaluated for their fitness.

To address the initialization of subpopulations, the RSI strategy is introduced. In the RSI approach, clustering is no longer simply used to partition subpopulations; instead, subpopulations are generated with predefined cluster centers. This process is akin to sowing seeds. Firstly, the number of subpopulations is specified, denoted as N, and the subpopulation size, denoted as M. Next, an archive is used to guide the

initialization process of subpopulations. Each seed is considered as the center of a cluster to generate new subpopulations. The set of seeds, contained within the archive, comprises all seeds used to generate subpopulations. Initially, this archive is empty. The algorithm checks if the seed set is empty. If it is empty, *N* seed individuals are generated by DE according to Equation (3.1), and these generated seeds are placed into the seed set. If the archive is not empty, the existing seeds in the archive are used. If the number of seeds in the archive is less than N, randomly generated seeds are added to the seed set until *N* is reached. Subsequently, individuals are generated in each subpopulation according to Equation (3.1) until the subpopulation size *M* is reached. It is important to note that each individual added to a subpopulation should satisfy the condition that its distance from the seed individual of the subpopulation is smaller than that from other seeds. Once all populations are generated, the algorithm evaluates all individuals and clears the archive. This concludes the initialization process of subpopulations.

Algorithm 1 RSI Strategy

input:	Archive, N, M
output:	*N* subpopulations of size *M*
1	Begin
2	Seeds={};
3	**If** *Archive* is empty
4	Generate *N* individuals according to Eq. (3.1) and add them into *Seeds*;
5	Else
6	Add individual in *Archive* into *Seeds*;
7	Generate (*N*-size(*Archive*)) individuals and add into *Seeds*;
8	End If
9	**For** $seed_i$ in Seeds
10	Generate *M* individuals around the $seed_i$, which satisfy that the distance to $seed_i$ is less than the distance to $seed_j$ ($j \neq i$);
11	End For
12	**If** all subpopulations' sizes are enough (*M*)
13	Evaluate the individuals in the population;
14	*Archive*={};
15	End If
16	End

As shown in **Algorithm 1**, this algorithm is used to initialize *N* subpopulations,

each containing M individuals. The algorithm first checks if the archive is empty. If it is, then N individuals are generated as seeds according to a specific equation, and M individuals are generated around each seed. Otherwise, individuals are selected from the archive as seeds, and additional individuals are generated to fill the required number of seeds. Next, for each seed individual, M individuals are generated, ensuring that their distance to the seed is less than to any other seed, and all generated individuals are evaluated. Finally, the archive is cleared for the next evolutionary iteration.

3.3.2 DE Optimization Process

After generating the subpopulations, each subpopulation is optimized using the DE algorithm. Since diversity is already maintained by multiple subpopulations, it is necessary to enhance the local search capability to achieve a balance between exploration and exploitation. Therefore, the DE/best algorithm, which has a strong local search capability, is adopted. In the DE/best algorithm, the mutation operation is not based on the differences between the current individual and other individuals, but on the differences between the best individual (i.e., the best individual) in the population and other individuals. This strategy aims to guide the algorithm to converge faster to the proximity of the optimal solution, as it directly uses the best information in the population.

$$x_{i,j,0} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min}) \tag{3.1}$$

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } rand_{i,j}[0,1] \le Cr \text{ or } j = jrand, \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{3.2}$$

The scaling factor F for the mutation operation is set to 0.5, which is a common setting in DE. According to Equation (3.2), the crossover probability Cr is set to 0.9, and the mutated individuals are mixed with the original individuals to construct tests separately.

3.3.3 Subpopulation Merge

The purpose of merging subpopulations is to focus the energy of subpopulations on the best solutions in the current region, enabling the sharing of information among subpopulations to accelerate the convergence of the current local region and promote

the global search process. Firstly, population merging can enhance search diversity, accelerate the convergence of the current local region, and promote the global search process. Secondly, it speeds up the convergence of the local region. By merging subpopulations with similar best individuals, their individual information can be effectively used to speed up the convergence rate of the current local region. This helps to adapt to changes in dynamic environments faster, improving the responsiveness of the algorithm. Finally, it promotes the global search process. Population merging helps to share the best individual information of each subpopulation in the global search process, thereby providing more global search capabilities for the entire optimization process and increasing the chances to find the global optimal solution. Therefore, population merging is a strategy aimed at improving the search efficiency and performance of optimization algorithms in dynamic environments, enabling them to better adapt to environmental changes and quickly find the optimal solution.

In practical operation, after all subpopulations have run for one generation, merging may converge to the same peak subpopulation. The specific operation process is as follows.

Step 1: Predefine the parameters "*max_subsize*" and the threshold point "th_{merge}" for the merging process. Represent the set including all subpopulations as $S = \{subpop_1, \ldots, subpop_n\}$, where $n \leq N$. This set comprises the best individuals from each subpopulation, denoted as $G = \{x_{best,1}, \ldots, x_{best,n}\}$.

Step 2: Calculate the distance between each pair of best individuals in G and find the minimum distance "*distmin*" along with its corresponding subpopulations i and j ($i \neq j$). If *distmin* is below the threshold "th_{merge}" and the sum of the sizes of subpopulations i and j is less than "*max_subsize*," it is considered that the distance between these two subpopulations is sufficiently small, and their merging will not cause the population size to exceed the limit. Consequently, the subpopulation with inferior x_{best} is merged with the one having superior x_{best}. Through this merging operation, subpopulations with poorer search histories are combined with those having better search histories, thereby enhancing the search capability and convergence speed of the entire population.

Subsequently, the subpopulation with inferior x_{best} is removed from the subpopulation set S to ensure that they no longer affect subsequent operations. Repeat this process until no valid subpopulation mergers occur. This process effectively uses the individual information among subpopulations, promotes collaborative cooperation among populations, speeds up the convergence rate of the entire population, and enhances the search efficiency and performance in dynamic environments.

The parameter "*max_subsize*" is set to limit the size of subpopulations, thereby requiring an appropriate upper limit to control the scale of subpopulations. If the subpopulations are too large, they will consume too many limited resources on local optimal solutions, which is inconsistent with the original intention of the multi-population method. If "*max_subsize*" is too small, the subpopulations converging to the same peak will not be able to communicate and share individual information in a timely manner. Regarding the conditions for subpopulation merging, when the distance between the best individuals in two subpopulations is less than the threshold "th_{merge}," and the sum of their population sizes is less than the upper limit of "*max_subsize*," merging them can fully use the information of their subpopulations and accelerate the convergence of the current local region. This condition ensures that the merging does not exceed the population size limit and ensures that the distance between the two subpopulations is close enough to perform the merging operation. However, when the distance between the best individuals in two subpopulations is greater than the threshold "th_{merge}," it is considered that the subpopulations are seeking different peaks or moving towards different local regions, and it is not appropriate to perform the operation at this time.

3.3.4 Archive-based historical information reuse (AHIR)

To fully make use of the historical information to guide the search, the AHIR strategy is proposed. Suppose that the set denoted as $S_{prev\_env} = \{subpop_1, ..., subpop_n\}$ contains n subpopulations from the previous environment. Two parameters is defined: the threshold for duplicate removal and the convergence radius, which are denoted as th_{dr} and r_{conv}, respectively. First, the searching radius for every subpopulation is

calculated in the S_{prev_env} according to:

$$radius_i = \frac{1}{size(subpop_i)} \sum_{j \in subpop_i} d(X_j, X_{center}) \tag{3.3}$$

In this formula, the function $d(\cdot,\cdot)$ calculates the Euclidean distance between two vectors, while the function *size* ( ) returns the size of the subpopulation. X_{center} represents the arithmetic mean of all individuals in the subpopulation.

Specifically, the AHIR strategy comprises several key steps:

Algorithm 2 AHIR Strategy

input:	S_{prev_env}={subpop$_1$, ..., subpop$_n$}, th_{dr}, r_{conv}
output:	Archive
1	Begin
2	Archive={};
3	**For** subpop$_i$ in S_{prev_env}
4	**If** radius$_i$ < r_{conv}
5	*Archive* += $x_{best,i}$;
6	End If
7	End For
8	While True
9	**If** there exists i, j ($i \neq j$) that $d(Archive[i], Archive[j]) < th_{dr}$
10	Remove the worse individual from the *Archive*;
11	**Else**
12	Break;
13	End If
14	End While
15	End

Step 1, Archive Construction: In the previous environment, the best individual from each subpopulation is added to the archive to record the excellent solutions from the previous environment. Step 2, Search Radius Computation: For each subpopulation in the archive, its search radius is computed. The search radius represents the average distance between individuals in the subpopulation and the centroid of that subpopulation, used to determine if the subpopulation is in a convergent state. If the search radius of a subpopulation is less than the threshold r_{conv}, it is considered to be in a convergent state, and the best individual from the subpopulation is added to the

archive. Step 3, Duplicate Removal Operation: The duplicate removal operation is proposed to remove similar individuals from the archive. If there exist two individuals in the archive with a distance between them less than the threshold th_{dr}, the inferior individual is removed from the archive. This removal operation is repeated until there are no similar individuals based on th_{dr} in the archive. Finally, Extraction of Local Optima: When the search radius of a subpopulation is less than the convergence radius, the best individual from that subpopulation is added to the archive to record the local optimum of that subpopulation in the current environment. Detailed information of the AHIR strategy is presented in **Algorithm 2**.

3.3.5 The Whole HIDE Algorithm

In this section, the overall procedure of the proposed HIDE algorithm is presented. The following are the steps of the **Algorithm 3**.

Algorithm 3 HIDE

input: max_FEs, N, M, th_{merge}, th_{dr}, r_{conv}
output: best individual
Begin
Archive={};
S=RSI strategy(*N*, *M*, Archive);
While FEs < max_FEs
    For $subpop_i$ in *S*
        DE($subpop_i$);
    End For
    S=Subpopulation merge(*S*, th_{merge});
    If the environment has changed
        Archive=AHIR strategy(*S*, th_{dr}, r_{conv});
        S=RSI strategy(*N*, *M*, *Archive*);
        Update *FEs*;
    End If
End While
End

The population is initialized using the RSI strategy, creating initial subpopulations for evolution. Throughout the evolution process, each subpopulation undergoes independent updates using the DE process, striving for improved solutions. The

algorithm iterates until it reaches the preset maximum number of function evaluations (*max_FEs*). To enhance global search capabilities and reduce overlap between populations, a subpopulation merging process is employed, facilitating efficient exploration of the solution space. While environment detection techniques can be applied to check for changes in each generation, this study does not focus on such detection, assuming environmental changes occur after a fixed number of function evaluations. In case of an environment change, the AHIR strategy is activated. This strategy preserves historical information by storing the best individuals from the previous environment in an archive, which is then used to generate subpopulations in the new environment, facilitating the location and tracking of moving peaks. The detailed implementation of these steps, including the entire procedure of the HIDE algorithm, can be found in Algorithm 3.

3.4 Experiment

3.4.1 Experiment Setting

In the experimental section of this chapter, the Moving Peaks Benchmark (MPB) was used to evaluate the performance of the optimization algorithms. MPB serves as a standardized test suite for evaluating the performance of optimization algorithms on DOP. It was proposed by Branke et al. [25] and has been widely adopted in academic research due to its high configurability and ease of implementation, enabling fair comparisons and evaluations of different algorithms. MPB allows for easy parameterization, and after parameter setting, the fitness landscape of MPB's objective function changes over time (after a certain number of function evaluations), reflecting variations in the number, positions, heights, and widths of peaks. The characteristic feature of MPB is the dynamically changing peaks over time, which can simulate the characteristics of many real-world DOP, such as dynamic resource allocation and mobile target tracking. The mathematical expression of MPB's fitness function is as follows:

$$F(\mathbf{x},t)=\max_{i=1,\ldots,P}\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}(x_j(t)-X_{ij}(t))^2} \tag{3.4}$$

where P is the number of peaks in the environment, that is, the number of local optimal regions; $x(t) = [x_1,..., x_D]$ represents a vector of D-dimensional search space in the t-th environment, also known as a candidate solution or individual; $H_i(t)$, $W_i(t)$, $X_i(t) = [X_{i1},..., X_{iD}]$ represent the height, weight, and position of the i-th peak in the t environment, respectively. The height and weight obey the random Gaussian distribution $\sigma \sim N(0,1)$ and are affected by the parameter height disturbance degree S_H and the weight disturbance degree S_W respectively,

$$H_i(t) = H_i(t-1) + S_H \cdot \sigma \tag{3.5}$$

$$W_i(t) = W_i(t-1) + S_W \cdot \sigma \tag{3.6}$$

The expressions for height and width imply that both the height and width of the peaks are randomly sampled from a standard normal distribution with a mean of 0 and a standard deviation of 1. This means that the height and width of the peaks are random and exhibit characteristics of a normal distribution. Parameters S_H and S_W control the extent to which these random values affect the peak height and width, determining the range of fluctuation for both height and width. Here, t-1 denotes the state of the previous environment, indicating that the state of the previous environment influences the peak height and width in the current environment.

The position is changed by a velocity vector:

$$\boldsymbol{X}_i(t) = \boldsymbol{X}_i(t-1) + \boldsymbol{v}_i(t) \tag{3.7}$$

$$\boldsymbol{v}_i(t) = \frac{S_X}{\left|(1-\lambda)\boldsymbol{r} + \lambda \boldsymbol{v}_i(t-1)\right|}((1-\lambda)\boldsymbol{r} + \lambda \boldsymbol{v}_i(t-1)) \tag{3.8}$$

The purpose of this vector is to predict and adjust the position of peak i in the next environment based on the state of the previous environment, $Xi(t\text{-}1)$. r is a random vector; λ is a correlation coefficient, which controls the correlation between the old environment and the new environment. The moving vector $v_i(t\text{-}1)$ is normalized and then multiplied by the moving length S_X to get the moving vector $v_i(t)$ of the next environmental change; S_X controls the intensity of the environmental change. A series of parameter settings for MPB are listed in Table 1.

In this chapter, the parameters of the proposed HIDE algorithm include the number of initial seeds N, the size of subpopulation M, the size of the largest subpopulation *max_subsize*, subpopulation merging threshold th_{merge}, duplication removal threshold th_{dr}, and the convergence radius r_{conv}. Their corresponding settings are: $N = 10$, $M = 25$, $max\_subsize = 50$, $th_{merge} = th_{dr} = 5.0$, and $r_{conv} = 1.0$.

Table 3.1 Parameter Setting for MPB

Parameter	Value
Max FEs, *max_FEs*	5e5
Number of peaks, P	10
Environmental change frequency	5000
Number of changes, C	100
Height severity, S_H	7.0
Width severity, S_W	1.5
Peak shape	Cone
Correlation coefficient, λ	0.0
Shift length, S_X	5.0
Dimension, D	5
Search range	[0, 100]
Height range	[30.0, 70.0]
Width range	[1.0, 12.0]

The algorithms are run on the MPB test suite, and different combinations of MPB parameters form 12 DOPs with different environmental changing characteristics. Each problem is composed of 100 continuously changing environments. After every 5000 *FEs*, the environment changes. Therefore, the complete execution of the algorithm contains 5e5 *FEs*. To reduce the error caused by randomness, each algorithm is run 20 times on each corresponding MPB problem instance. Each MPB instance will be reinitialized according to the given parameter settings. The average value and standard deviation of the error obtained are taken as the final output. The comparing algorithms include standard DE algorithms with different mutation strategies and different Cr parameter settings, which are DE/best/0.1, DE/best/0.9, DE/rand/0.1, and DE/rand/0.9, and some competitive multipopulation approaches and DE variants in global optimization, which are jDE [20], Cloudde [27], CESO [31], and CPSO [32]. To ensure

fairness of the comparison, the parameters of the comparing algorithms are consistent with the settings in the literature.

3.4.2 Performance Measure

In this chapter, two widely used indicators are used to evaluate the performance of the algorithm, which are offline error (*Eo*) and the best before change error (*Eb*). The calculation formula of *Eo* is shown as

$$E_o = \frac{1}{C \times N} \sum_{i=1}^{C} \sum_{j=1}^{N} E_{ij} \tag{3.9}$$

where C is the number of environmental changes, N is the *FEs* spent in each environment, E_{ij} is the error of the j-th *FE* under the i-th environment, and the error is defined as the difference between the fitness value of the best solution found by the algorithm and the real optimal solution under the current environment. E_o reflects the ability of the algorithm to react to environmental changes. The E_b calculation formula is shown in

$$E_b = \frac{1}{C} \sum_{i=1}^{C} E_{i,best} \tag{3.10}$$

where $E_{i,best}$ is the best error obtained in the i-th environment. E_b reflects the global search ability of the algorithm in dynamic environments. The combination of these two indicators can make a comprehensive evaluation and measurement of the performance of the DOP algorithms.

3.4.3 Results and Discussions

The experimental results are presented in Table 3.2, Table 3.3, and Table 3.5, considering cases where P=5, 10, and 20. The parameter λ is divided into 0 and 1, while S_X is considered as 1 and 5. Each row corresponds to the offline error E_o and the best error before environmental change E_b. Standard deviations over 20 runs are indicated in parentheses. The best results for this set of test cases are shown in bold font in the table. Additionally, a Wilcoxon rank-sum test at the 0.05 significance level is conducted to assess whether there are significant differences in the comparison results. The symbols “+”, “=”, and “-” indicate whether the performance of the HIDE algorithm is

significantly better than, equal to, or worse than the compared algorithms.

Table 3.2 Experimental Results on the MPB with P=5

P	λ	S_X	Error	DE/best/0.1	DE/best/0.9	DE/rand/0.1	DE/rand/0.9	jDE	Cloudde	CESO	CPSO	HIDE
5	0	1	E_o	17.61(3.36)+	13.19(2.44)+	23.50(5.29)+	18.19(3.65)+	37.48(7.97)+	29.44(5.49)+	8.47(3.02)+	3.86(0.85)+	**2.48(0.52)**
			E_b	10.44(2.38)+	11.34(2.27)+	11.08(2.43)+	9.29(2.40)+	9.34(2.58)+	8.35(2.22)+	6.33(3.22)+	1.38(0.41)+	**0.33(0.38)**
		5	E_o	17.28(3.41)+	13.09(2.37)+	23.36(5.38)+	18.05(3.74)+	37.42(8.04)+	29.48(5.60)+	15.72(4.72)+	9.16(1.94)+	**5.63(0.97)**
			E_b	10.21(2.14)+	11.25(2.25)+	10.85(2.50)+	9.11(2.40)+	9.26(2.62)+	8.33(2.32)+	8.83(3.54)+	2.04(0.79)+	**0.50(0.42)**
	1	1	E_o	17.54(3.69)+	12.99(2.41)+	23.44(5.17)+	18.07(3.67)+	37.14(7.37)+	29.33(5.82)+	8.47(2.94)+	3.93(0.90)+	**2.48(0.65)**
			E_b	10.41(2.61)+	11.14(2.27)+	11.13(2.64)+	9.14(2.48)+	9.34(2.50)+	8.24(2.36)+	6.13(3.13)+	1.38(0.58)+	**0.34(0.69)**
		5	E_o	17.74(3.54)+	13.18(2.52)+	23.64(5.37)+	18.01(3.80)+	37.26(7.55)+	29.57(6.36)+	16.35(4.25)+	9.28(2.12)+	**5.68(1.05)**
			E_b	10.63(2.50)+	11.31(2.34)+	11.23(2.75)+	8.95(2.74)+	9.28(2.50)+	8.32(2.65)+	8.91(3.13)+	2.32(1.07)+	**0.56(0.67)**

Table 3.3 Experimental Results on the MPB with P=10

P	λ	S_X	Error	DE/best/0.1	DE/best/0.9	DE/rand/0.1	DE/rand/0.9	jDE	Cloudde	CESO	CPSO	HIDE
10	0	1	E_o	16.42(3.24)+	13.96(2.82)+	20.52(3.53)+	16.43(3.07)+	34.30(6.42)+	29.23(5.65)+	7.97(2.53)+	4.17(0.67)+	**2.68(0.38)**
			E_b	11.53(2.79)+	12.72(2.74)+	12.14(2.66)+	10.42(2.66)+	10.86(2.81)+	9.74(2.41)+	5.92(2.39)+	1.90(0.38)+	**0.74(0.50)**
		5	E_o	16.53(3.14)+	14.08(2.68)+	20.54(3.66)+	16.49(3.11)+	34.16(6.32)+	29.12(5.44)+	14.89(3.42)+	8.37(1.30)+	**5.69(0.83)**
			E_b	11.61(2.54)+	12.82(2.57)+	12.04(2.64)+	10.47(2.69)+	10.73(2.58)+	9.75(2.38)+	9.79(2.81)+	2.61(0.68)+	**1.39(0.68)**
	1	1	E_o	16.76(2.84)+	14.22(2.42)+	20.56(3.56)+	16.60(3.06)+	34.36(6.34)+	28.92(5.08)+	8.83(2.82)+	4.19(0.73)+	**2.81(0.69)**
			E_b	11.89(2.23)+	12.95(2.33)+	12.12(2.50)+	10.59(2.49)+	10.83(2.64)+	9.66(2.17)+	6.75(2.73)+	1.96(0.61)+	**0.87(0.80)**
		5	E_o	17.08(3.04)+	14.35(2.22)+	20.84(3.37)+	16.84(2.91)+	34.79(6.21)+	29.70(5.53)+	15.31(3.86)+	8.58(1.54)+	**5.66(0.95)**
			E_b	12.23(2.58)+	13.08(2.16)+	12.43(2.52)+	10.84(2.50)+	11.13(2.56)+	9.99(2.45)+	9.89(3.31)+	2.72(0.75)+	**1.37(0.90)**

Table 3.4 Experimental Results on the MPB with P=20

P	λ	S_X	Error	DE/best/0.1	DE/best/0.9	DE/rand/0.1	DE/rand/0.9	jDE	Cloudde	CESO	CPSO	HIDE
20	0	1	E_o	15.76(1.63)+	14.53(1.86)+	19.25(1.63)+	15.37(1.53)+	31.75(2.95)+	27.57(2.84)+	9.32(2.92)+	4.19(0.54)=	**3.89(1.01)**
			E_b	11.42(1.49)+	13.50(1.83)+	11.74(1.32)+	10.14(1.35)+	10.31(1.35)+	9.41(1.27)+	7.52(2.91)+	2.17(0.41)=	**2.07(0.92)**
		5	E_o	15.81(1.57)+	14.19(1.57)+	19.26(1.63)+	15.27(1.74)+	31.63(3.13)+	28.06(2.65)+	14.07(2.20)+	7.91(0.85)+	**5.93(0.70)**
			E_b	11.53(1.52)+	13.17(1.53)+	11.82(1.38)+	10.04(1.69)+	10.33(1.45)+	9.59(1.35)+	9.87(2.04)+	2.90(0.62)+	**2.05(0.55)**
	1	1	E_o	15.88(1.63)+	14.14(1.52)+	19.51(1.59)+	15.52(1.66)+	31.85(2.80)+	27.84(2.81)+	9.83(1.75)+	4.27(0.54)+	**3.81(0.77)**
			E_b	11.53(1.60)+	13.10(1.52)+	11.93(1.30)+	10.38(1.59)+	10.36(1.24)+	9.43(1.30)+	7.84(1.85)+	2.23(0.46)=	**1.92(0.74)**
		5	E_o	15.98(1.42)+	14.32(1.39)+	19.66(1.70)+	15.54(1.54)+	32.10(2.99)+	28.45(2.97)+	15.58(3.01)+	7.96(0.90)+	**5.84(0.73)**
			E_b	11.68(1.28)+	13.28(1.41)+	12.15(1.33)+	10.38(1.42)+	10.55(1.31)+	9.67(1.41)+	11.04(2.75)+	2.88(0.65)+	**2.00(0.63)**

It can be observed that in all 12 MPB test cases, the HIDE algorithm outperforms all other algorithms, with the minimum values of E_o and E_b obtained for each instance indicating the strong performance of the HIDE algorithm in handling environmental changes and conducting global searches. Furthermore, our proposed HIDE algorithm adopts a strategy of generating subpopulations based on predefined or stored cluster centers (seeds), which outperforms CPSO, which partitions the population into subpopulations. These results suggest that the strategy of forming subpopulations

proposed in this study is effective and competitive among multi-population methods for DOP.

3.5 Effect of the AHIR Strategy

This section investigates the impact of the AHIR strategy on the performance of the HIDE algorithm. When the AHIR strategy is removed, all seeds initialized after each environment change are randomly generated. Table V shows the results of the proposed algorithm with or without AHIR, which are denoted as HIDE and HIDE-w/o-AHIR respectively, on MPB with different parameter settings.

Table 3.5 Experimental Results of the Effect of the AHIR Strategy on the MPB Test Suite

	HIDE	HIDE-w/o-AHIR
P=5 λ=0 S_X=1	**2.48(0.52)** **0.33(0.38)**	18.77(3.37) 5.01(1.65)
P=5 λ=0 S_X=5	**5.63(0.97)** **0.50(0.42)**	18.69(3.40) 4.92(1.90)
P=5 λ=1 S_X=1	**2.48(0.65)** **0.34(0.69)**	18.56(3.60) 4.85(1.82)
P=5 λ=1 S_X=5	**5.68(1.05)** **0.56(0.67)**	18.68(3.48) 5.03(1.77)
P=10 λ=0 S_X=1	**2.68(0.38)** **0.74(0.50)**	15.59(2.45) 5.94(1.42)
P=10 λ=0 S_X=5	**5.69(0.83)** **1.39(0.68)**	15.66(2.24) 5.99(1.38)
P=10 λ=1 S_X=1	**2.81(0.69)** **0.87(0.80)**	15.71(2.23) 6.01(1.27)
P=10 λ=1 S_X=5	**5.66(0.95)** **1.37(0.90)**	15.70(2.27) 5.92(1.70)
P=20 λ=0 S_X=1	**3.89(1.01)** **2.07(0.92)**	13.90(1.33) 5.78(1.14)
P=20 λ=0 S_X=5	**5.93(0.70)** **2.05(0.55)**	13.99(1.41) 5.89(1.17)
P=20 λ=1 S_X=1	**3.81(0.77)** **1.92(0.74)**	13.92(1.43) 5.83(1.17)
P=20 λ=1 S_X=5	**5.84(0.73)** **2.00(0.63)**	14.23(1.54) 6.05(1.39)

The upper row of each cell in the table is E_o, the lower row is E_b, and the result in brackets is the standard deviation. By comparison, the better results are expressed in bold font. It can be seen that the HIDE algorithm with AHIR is much better than the HIDE algorithm without AHIR, in terms of the E_o or E_b. This shows that the HIDE algorithm with AHIR has brought great benefit to the HIDE algorithm, and the AHIR strategy is the core of the algorithm because it realizes the utilization of historical information.

3.6 Conclusion

In this study, three strategies, namely RSI, AHIR, and HIDE, were investigated for DOP. RSI demonstrated effective resource allocation among subpopulations during the initial search phase, promoting a balanced exploration of the search space. AHIR leveraged past information to accelerate search processes in new environmental conditions, enhancing the efficiency of the algorithm. HIDE exhibited rapid adaptation to environmental changes while maintaining robust global search capabilities.

Looking ahead, there are avenues for further improvement and exploration in dynamic optimization research. Firstly, parameters such as N and M in the HIDE algorithm may require fine-tuning to optimize performance. Developing an adaptive mechanism to adjust these parameters dynamically could enhance the algorithm's adaptability and efficiency. Secondly, while the subpopulations in HIDE operate collaboratively, there is potential to explore the independent exploitation of partial search spaces within each subpopulation. Implementing constraints to guide subpopulations in exploring specific subspaces could lead to more targeted and efficient search processes. These future directions hold promise for advancing the effectiveness and applicability of DOP algorithms. Additionally, integrating solutions for DOP with addressing challenges such as scalability, multi-objectivity, and high computational costs could provide comprehensive approaches for tackling complex optimization tasks.

This work, entitled “Historical information-based differential evolution for dynamic optimization problem,” was published in the Proceedings of the IEEE Congress on Evolutionary Computation (CEC) in August 2021.

CHAPTER 4

MULTI-CRITERIA EVOLUTIONARY ALGORITHM FOR MULTI-TASK OPTIMIZATION

4.1 Introduction

MTOP [27]-[150] is a promising research area. The core assumption of MTOP is that knowledge gained from optimizing one task could be applied to improve the performance of other tasks [30]-[185]. This is because, in certain dimensions, if the optimal solutions of two different tasks exhibit similarity, the knowledge of optimal solutions from one task can guide the evolutionary search for another. Many real-world optimization problems [186]-[190], such as vehicle routing problems [186]-[187], support this assumption. For example, in daily life, experienced drivers are preferred for facing new tasks each time. They are preferred because they can use past experiences to solve new problems more quickly and effectively. Therefore, dealing with multi-task problems is more efficient than focusing on optimizing a single task. In recent years, MTOP have attracted increasing attention and related research from more scholars. A search using the keyword “multitasking” in the two most prestigious journals in the IEEE Xplore and EC domains, IEEE Transactions on Cybernetics (TCYB) and IEEE Transactions on Evolutionary Computation (TEVC), reveals that the research interest in MTOP has been steadily increasing over the past decade, with a particularly significant rise in the last five years on IEEE Xplore, and an unprecedented increase in attention to this issue in the top EC journals in the last five years as well, as Table 4.1 and Figure 4.1 shows.

In dealing with MTOP, EC emerges as a promising approach [164]. Due to its robust search capabilities and straightforward implementation, EC has been effectively utilized to address numerous complex optimization challenges. Widely used EC algorithms include GA [2] [4] [41] [59], PSO [51] [53] [55] [56], ACO [114], DE [32] [44]-[50], ES [128], and EDA [139]-[166] [195]. However, it is important to highlight

that traditional EC solvers typically initiate the search without any prior knowledge of the tasks they are tackling. Nevertheless, given that problems rarely occur independently, addressing one problem can provide insights beneficial for solving other correlated problems. In recent years, there has been growing interest in ETO: An approach that merges EC solvers with knowledge acquisition and transfer from pertinent domains to improve optimization efficiency and effectiveness. [27].

Table 4.1 Number of papers of MTOP in the last ten years

Year	IEEE Xplore	TCYB	TEVC
2014	105	8	0
2015	116	15	0
2016	139	13	1
2017	173	16	2
2018	207	14	1
2019	295	27	8
2020	300	29	8
2021	610	45	8
2022	1060	84	29
2023	1329	77	72

In existing approaches, multi-task problems are often treated as separate problems rather than components or subproblems of the entire MTOP. Whether using single-population or multi-population algorithms, solving MTOP typically involves designing KT strategies. However, designing effective KT strategies is a challenging task. In this chapter, a novel idea that considers multi-tasks as multi-criteria optimization problems is proposed. The populations are evolved to address multiple related criteria simultaneously, enabling a single search run to obtain optimal solutions for different tasks. Inspired by this concept, the attempt is made to address MTOP using the MCOP approach. For example, consider a classroom composed of high school students who need to study subjects such as Mathematics, English, Physics, and Chemistry. Traditional multi-population methods divide students into several groups, each equivalent to a subpopulation, focusing on studying a specific subject. These groups then exchange learning experiences, representing KT. In contrast, our multi-criteria approach can be likened to evaluating students' learning performance using exam scores. Exam scores not only serve as indicators of students' learning performance but also

reflect their mastery and proficiency levels in different subject areas. Consequently, teachers can assess students' learning status in various subjects based on exam scores and provide targeted guidance and assistance as needed. This problem-solving approach is relatively novel, as there was no relevant literature in the EC field before embarking on this work.

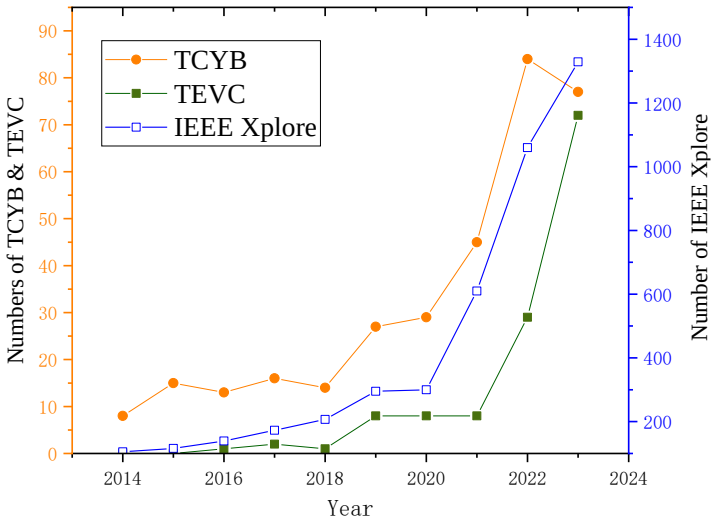


Figure 4.1 Number of papers of MTOP in the last ten years

This problem-solving approach is relatively novel, as there was no relevant literature in the EC field before embarking on this work. The utilization of different criteria was alternated, and despite this simplicity, experimental results demonstrated promising performance compared to other algorithms. Subsequently, through discussions and insights gained during conference exchanges, our idea received academic recognition. Encouraged by the acknowledgment received at the GECCO conference, we endeavored to increase the complexity of the problem. We combined multi-objective optimization with MTOP, forming multi-objective MTOP (MO-MTOP). To address MO-MTOP, a multi-objective multi-criteria optimization problem (MO-MCOP) approach was employed, and a criterion selection strategy was proposed to enrich the algorithm and enhance its efficiency. This achievement was published in a journal after the conference.

4.2 Related Work

Some EC algorithms have been used to address MTOP, and existing research can

be broadly categorized into two categories. One category is based on single-population multi-factorial methods [150], [191]-[195], while the other category is based on multi-population algorithms [194]-[195].

The first category of methods is exemplified by the multi-factorial evolutionary algorithm (MFEA) [150], which has provided a research paradigm for subsequent studies. MFEA is an approach that uses a single population with multiple subpopulations to solve MTOP. The MFEA framework divides a population into multiple subpopulations, each focusing on optimizing a different task. These subpopulations may share information during the evolution process to facilitate collaborative learning and KT among tasks. Building upon this framework, some enhanced variants of MFEA have been developed for further research and solving MTOP. For instance, the cognizant MTOP, referred to as MFEA-II, leverages data generated during the processing of multiple tasks to learn relationships among these tasks [191].

The second category primarily addresses multi-task problems by maintaining multiple populations [142]-[145]. In this approach, multiple populations are typically used, with each population responsible for solving a specific task. Each population independently executes the optimization process, attempting to find the optimal solution to meet the requirements of its corresponding task. Chen et al. [142] introduced an adaptive archive-based evolutionary method specifically designed for multi-task optimization scenarios, demonstrating its effectiveness in enhancing multi-task optimization performance. Building upon this, Huang et al. [143] introduced an auxiliary agent-based evolutionary framework that combines adaptive KT mechanisms, thereby improving optimization efficiency across different tasks. Liu et al. [144] made contributions in this field by proposing an auxiliary agent-based multi-task evolutionary algorithm that leverages agent models to enhance multi-task optimization performance. Similarly, Chen et al. [32] introduced KT crossover to transfer knowledge among subpopulations during the evolution process. Gong et al. [146] explored evolutionary multi-task strategies with a focus on dynamic resource allocation, providing insights

into optimizing multiple tasks under dynamically allocated resources. Finally, Wei and Zhong [145] conducted preliminary research on KT mechanisms in multi-classification tasks using gene expression programming, offering potential avenues for KT in complex optimization scenarios. These studies collectively advance our understanding and capabilities in utilizing EC techniques to address the challenges of multi-task optimization.

Regarding MTOP, recent research findings have been summarized in several representative reviews [196]-[167]. However, existing MTOP algorithms, whether employing a single population of multiple groups or utilizing multiple populations, still face limitations in treating the multiple tasks of multi-task problems as separate issues. Therefore, existing MTOP algorithms must be used in conjunction with carefully designed KT strategies between tasks. However, designing an effective KT strategy is a challenging problem and may even lead to negative transfer across tasks [162]. To address this limitation, populations can be evolved correspondingly using multiple relevant criteria, thus searching for optimal solutions for all different tasks in a single run. MTOP is viewed as MCOP [196], allowing for more efficient resolution of MTOP by fully using knowledge across different tasks using multiple criteria in MCOP for environmental selection and population evolution. Through this approach, addressing the complex challenge in MTOP of identifying valuable knowledge and transferring it across various related multi-objective tasks transforms into a more manageable task: employing multiple assessment criteria to direct environmental selection and population evolution, thus generating optimal solutions that satisfy all tasks' different criteria. Furthermore, this approach has been applied to address the MO-MTOP. Therefore, this research direction holds tremendous potential, providing important methodologies for handling MTOP and making substantial contributions to the advancement of related research domains.

4.3 Treating Multi-task Optimization as Multi-Criteria Optimization

4.3.1 Introduction

MTOP is regarded as MCOP to effectively address MTOP. Specifically, we

approach MTOP, which encompasses multiple tasks, as MCOP with multiple evaluation criteria (i.e., fitness functions) for individual selection and population evolution. This reframing of MTOP addresses a significant challenge: how to acquire and transfer valuable knowledge among diverse tasks. It simplifies the issue into one of utilizing multiple evaluation criteria to guide selection operations and population evolution, facilitating the discovery of optimal solutions for various tasks.

For example, consider a class composed of high school students who need to study various subjects such as mathematics, English, physics, and chemistry, each subject representing a task. Past multi-population methods have divided students into several groups, with each group acting as a subpopulation focusing on studying a specific subject. These groups exchange learning experiences, which is akin to KT. The multi-criteria approach can be likened to evaluating students' learning performance using exam scores. Exam scores not only serve as indicators of students' learning performance but also reflect their proficiency and competence levels in different knowledge domains. The overarching goal is to optimize the performance of all tasks, similar to students achieving excellent scores in multiple subjects.

The work in this chapter represents the first attempt to address MTOP by viewing it as MCOP, which was presented at the GECCO 2021 conference. To avoid ambiguity, it is explicitly stated that in this chapter, MCOP refers to a problem with multiple evaluation criteria, rather than a problem where the criteria themselves need to be optimized. The chapter adopts a cyclic multi-criteria approach, where the selection of which criterion to use is typically done in a predefined rotating order, rather than being randomly selected. In subsequent work, more efficient methods for criterion selection are explored.

For experimental investigation, a multi-criteria differential evolution (MCDE) algorithm was developed, employing multi-criteria strategies and utilizing the differential evolution algorithm as the optimizer. To assess the proposed algorithm's efficacy, thorough experiments were conducted using widely adopted benchmark datasets for MTOP and compared with some of the latest EMTO algorithms.

4.3.2 Method

Multiple tasks correspond to multiple fitness functions, which can serve as evaluation criteria for environmental selection and individual evolution. More importantly, the optimal solutions of fitness functions in different tasks may exhibit similarities in certain dimensions. For example, consider two tasks: one aims to minimize costs, while the other aims to maximize profits. Although the optimization objectives of these two tasks differ, in practice, there may exist a trade-off between minimizing costs and maximizing profits. For instance, reducing costs typically leads to an increase in profits, and vice versa. Therefore, in such cases, although the optimization objectives of the two tasks are different, their optimal solutions may exhibit similarities in certain dimensions, such as a balance point between costs and profits. Hence, in such scenarios, using a single fitness function as the guiding criterion for evolution not only aids in optimizing the corresponding task but also facilitates the optimization of other related tasks. Consequently, MTOP can be viewed as MCOP, and appropriate criteria can be selected at different stages to guide optimization.

To efficiently use multiple criteria, a multi-criteria strategy employing a round-robin approach is proposed. Specifically, the multiple criteria are sequentially activated in a round-robin manner to guide evolution. To achieve this, a parameter G is introduced to control the number of generations each criterion remains active. Each criterion is activated in turn and serves as the current fitness function to guide evolution for G generations, and every $K \times G$ generations constitute a complete cycle where all K criteria are activated once. Furthermore, to enhance the diversity in criterion usage, whenever all K criteria have been used once, the order of these K criteria is randomly shuffled. By doing so, the next $K \times G$ generations will select the K criteria in a different order. It is important to note that the population should be re-evaluated using the new criterion each time a criterion is switched

4.3.3 Experiment

In this experiment, the DE algorithm is adopted due to its simplicity, gradient-free nature, robustness, minimal parameter settings, and good parallelism, as detailed in

Section 2.1.1 regarding the introduction of the DE algorithm.

Six complex MTOP (i.e., P1-P6) from commonly used benchmark tests [162] are used to evaluate the proposed algorithm. Regarding the parameters of MCDE, the population size is configured as 50, the scaling factor F is specified as 0.5, and the crossover rate CR is defined as 0.6, following the recommendations in [160]. Additionally, G is set to 150.

For comparisons, the Wilcoxon rank-sum test with a significance level of α=0.05 [197] and performance metrics [162] are used. The symbols "+", "≈", and "-" denote that performance of MCDE is notably superior to, approximately equivalent to, and significantly inferior to other algorithms, respectively. In this section, MCDE was compared with the latest algorithms such as MFEA-I [30], MFEA-II [159], and EMT-EGT [160]. To ensure a fair comparison, the total maximum available evaluation times for each algorithm in each run is set to 1×105 [160]. To minimize statistical errors, each algorithm is executed independently 20 times, and the average results are then compared.

The comparison results presented in Table 4.2 demonstrate the effectiveness of MCDE. As shown in Table 4.2, MCDE achieves the best score in 4 out of 6 problems, while MFEA-I, MFEA-II, and EMT-EGT achieve the best score in 0, 1, and 1 problem, respectively.

Table 4.2 Comparison with State-of-the-Art Algorithms

Statistical term	MCDE	MFEA-I	MFEA-II	EMT-EGT
+/≈/-/	NA	**7**/0/5	**7**/0/5	**9**/0/3
# Number of best scores	**4**	0	1	1

According to the results of the Wilcoxon rank-sum test, the MCDE algorithm significantly outperformed MFEA-I, MFEA-II, and EMT-EGT on 7, 7, and 9 tasks, respectively. However, it produced inferior results on 5, 5, and 3 tasks, respectively. Therefore, the experimental results validate the efficiency of MCDE, indicating its potential to address MTOP as MCOP.

4.3.4 Conclusion

In this Section, the study aimed to treat MTOP as MCOP and efficiently solve them.

Experimental results suggest that this approach might be a promising direction for addressing MTOP. Regarding the limitations of the study, a cyclic multi-criteria strategy was employed, where the algorithm sequentially selects each evaluation criterion, akin to students taking exams in different subjects in a classroom. However, since the difficulty level varies across subjects, better selection of evaluation criteria, such as using different evaluation standards to guide environmental selection and population evolution in different generations or stages, could be a future research direction. In Section 4.4, methods for selecting criteria were explored. Furthermore, dynamic MTOP is also a potential research direction. For instance, in the early stages of optimization, more emphasis might be placed on coarse exploration and seeking global solutions, while in the later stages, finer tuning and convergence to local optima might be prioritized. Therefore, different sequences of evaluation criteria could be chosen.

4.4 MCOP for Multi-Objective Multi-Task Optimization

4.4.1 Introduction

MO-MTOP is a complex and challenging task that involves considering multiple optimization objectives and tasks, which may be interrelated or mutually influential. The complexity of MO-MTOP is mainly manifested in several aspects. Firstly, there may be objective conflicts, where improving one objective may result in the deterioration of another. Secondly, as mentioned earlier, there are task correlations, where different tasks may be interrelated or share some resources or information, posing challenges related to KT. Additionally, the scale of MO-MTOP problems can be significant, often involving large search spaces and complex problem structures. Therefore, efficient optimization algorithms need to be designed to address such complexity and scale. For example, consider a classroom of students, each of whom needs to study multiple subjects and achieve good grades. Each subject represents a task with its objectives and requirements. In this example, each student faces the challenge of multiple tasks and objectives. They need to study different subjects and strive to meet the objectives and requirements of each subject. Employing multi-objective optimization methods can assist students in balancing their learning and

performance across various subjects, thereby maximizing their learning outcomes. However, simultaneously handling multiple tasks during the evolutionary process in MO-MTOP offers distinct advantages over traditional multi-objective optimization problems. This is because tasks often exhibit correlations, allowing different tasks to share information and thereby enhancing optimization efficiency.

Below are listed the main innovative points and contributions of this chapter. “Multi-objective” refers to each subject having its objectives and requirements. For example, English learning may involve expanding vocabulary, improving listening, speaking, reading, and writing abilities, and achieving high scores in exams. In this example, each student faces the challenge of multiple tasks and objectives. They need to study different subjects and strive to meet the objectives and requirements of each subject. In such scenarios, multi-objective optimization methods can be utilized to help students balance their learning and performance across different subjects, maximizing their learning objectives. However, even in this case, handling MO-MTOP, which simultaneously deals with multiple tasks during the evolutionary process, has some advantages over traditional multi-objective optimization problems because tasks are correlated, and different tasks can share information, thereby enhancing optimization efficiency.

Existing research on MO-MTOP, whether adopting a single population with multiple groups like MFEA or utilizing multi-population methods, aims to reduce problem complexity by grouping tasks. In particular, multi-population methods can be likened to dividing students into different groups in a classroom to tackle various tasks, with each group having its objectives and tasks. Each group can be viewed as a subpopulation, focusing on solving specific learning tasks or problems. Meanwhile, multi-criteria evaluation can be analogous to using exam scores to assess students' learning performance. Exam scores not only serve as indicators of students' learning performance but also reflect their mastery and proficiency across different knowledge domains. Consequently, teachers can evaluate students' learning status in various subjects based on exam scores and provide targeted guidance and tutoring as needed.

Compared to the MCOP approach for addressing MTOP problems discussed in Section 4.3, the research problem in this chapter combines multiple objectives with MTOP, thereby increasing the complexity of the problem. In terms of methodology, in addition to continuing the idea of MCOP, a probability-based criterion selection approach is introduced instead of the cyclic criterion selection. This change allows for more flexible adaptation to different optimization requirements by selecting more appropriate evaluation criteria. Similar to classroom exams, the selection process is no longer based on the sequence of subjects but rather on the improvement of learning abilities for each subject, where subjects with better performance are more likely to be chosen.

Therefore, the contributions of this study are outlined below.

Firstly, this work attempts to solve MO-MTOP as MO-MCOP, providing a novel and promising approach for handling MO-MTOPs. Additionally, this work is the first to attempt to solve MO-MTOP by treating it as MO-MCOP.

Secondly, a Probability-based Criterion Selection Strategy (PCSS) is proposed, which selects and uses multiple evaluation criteria derived from corresponding probabilities. This strategy allows different criteria to have varying opportunities for selection, guiding individual selection and population evolution.

Thirdly, an Adaptive Parameter Learning (APL) method is further introduced to adaptively learn the selection probabilities of each criterion in PCSS. By employing APL, the algorithm can acquire suitable probabilities to help determine which criterion to aid in determining the criterion applicable for the current generation.

Fourthly, by integrating the above components, a Multi-objective Multi-criteria Evolutionary Algorithm (MO-MCEA) is established for addressing MO-MTOPs.

4.4.2 Method of MO-MCEA

4.4.2.1 Definition of MO-MTOP

A Problem Definition

MO-MTOP is a diagram for solving multiple multi-objective optimization tasks together. MO-MTOP can be defined as follows.

Given K multi-objective optimization tasks (assuming the objectives in every task are all minimization problems), denoted as $T_1, T_2, ..., T_K$, where the k^{th} task has M_k ($M_k > 1$) objective functions $F_k(x) = [f_1(x), f_2(x), \ldots, f_{Mk}(x)]$. The search space and the objective space of the k^{th} task are Ω_k and Ψ^{M}_k, respectively, and they satisfy that $Fk : \Omega_k \rightarrow \Psi^{M}_k$. The aim of a minimization MO-MTOP is to find the optimal solution set $\{x_k\}$ for each task T_k, such that $\{x_k\}$ satisfies

$$\{ x_k \} = \text{argmin } F_k (x_k | x_k \in \Omega_k) , \text{k} = 1, 2, 3, ..., K \quad (4.1)$$

As each F_k has multiple objectives, the following important concepts for each task Tk are considered to determine whether a solution is optimal according to the related definitions in the literature of multi-objective optimization [213, 214].

Definition 1 *Pareto domination* Given any two objective fitness vectors $u = [u_1, u_2, ..., u_M]$ and $w = [w_1, w_2, ..., w_M]$ in the objective space Ψ^M, u dominates w if $u_m \leq w_m$ for all $m = 1, 2, \ldots, M$ and $u \neq w$, denoted as u≤w.

Definition 2 Pareto optimal A solution vector $x \in \Omega$ is Pareto optimal if there is no $x^* \in \Omega$, such that $F(x^*)$ dominates $F(x)$.

Definition 3 Pareto set The Pareto set (*PS*) is a set of the Pareto optimal solutions, which can be represented as

$$PS = \{x \in \Omega \text{ and } x \text{ is Pareto optimal}\}. \quad (4.2)$$

Definition 4 Pareto front The Pareto front (*PF*) is composed of the solutions in PS, as

$$PF = \{F(x) | x \in \text{PS}\}. \quad (4.3)$$

Based on the above definitions, the optimal solution set $\{x_k\}$ for each task T_k is the PS of the T_k.

B. Task Similarity and KT

Studies have shown that KT between tasks with low correlation may even lead to a decrease in optimization performance [162]. Regarding the correlation between tasks, it can be categorized into three types. Firstly, if the Pareto sets of two tasks are highly similar, the population P can naturally maintain sufficient diversity, thus becoming the ideal Pareto optimal solution set for both tasks T_i and T_j. In this scenario, treating MO-

MTOP as MO-MCOP can enable finding the Pareto sets of both tasks using a single population. Secondly, if there is some similarity between the Pareto sets of two tasks, the population P will become a partial Pareto optimal solution set for both tasks T_i and T_j. However, since real-world problems typically require an adequate number of Pareto optimal solutions rather than all optimal solutions, a population P with enough Pareto optimal solutions can still be considered an acceptable Pareto optimal solution set for both tasks T_i and T_j. Thirdly, if the Pareto sets of two tasks are significantly different, and they share limited knowledge, theoretically, it is not advisable to integrate them into a single MO-MTOP.

When defining the similarity between two tasks, two indicators are used: intersection and similarity. The complete intersection (CI) and partial intersection (PI) indicate whether the Pareto frontiers of two internal tasks are similar in all dimensions and some dimensions, respectively. High similarity (HS), moderate similarity (MS), and low similarity (LS) represent high, moderate, and low similarity between two tasks determined by Pearson correlation measures of their function landscapes.

4.4.2.2 Framework of MO-MCEA

MO-MTOP involves multiple multi-objective functions. For example, assuming there are K tasks and M objectives, where $F_i(x)$ and $F_j(x)$ represent the fitness functions of tasks i and j, respectively, defined as $F_i(x)= [f_1(x), f_2(x), \ldots, f_{Mi}(x)]$ and $F_j(x)= [f_1(x), f_2(x), \ldots, f_{Mj}(x)]$. These functions serve as evaluation criteria for individual selection and population evolution. Our assumption for MTOP is that the optimal solutions (or Pareto optimal solutions) of different tasks may exhibit similarity. As demonstrated in Section 4.2, even with cyclic multi-criteria selection, rotating through each objective function corresponding to a task can outperform other multi-population algorithms. To further optimize the MCEA method, a more appropriate fitness function of a task is selected as the guiding criterion for evolution to refine the MCEA algorithm.

The overall framework of MO-MCEA is illustrated in Figure 4.2, consisting of three main parts. The first part is the collection of available fitness evaluation functions (i.e., a set of different evaluation criteria). The second part involves criterion selection

based on PCSS and APL, i.e., selecting one evaluation function from the set as the evaluation criterion for different evolutionary generations to evolve the population. The third part encompasses population evolution and problem optimization based on the selected criterion. In the following sections, PCSS, APL, and the complete algorithm will be introduced step by step.

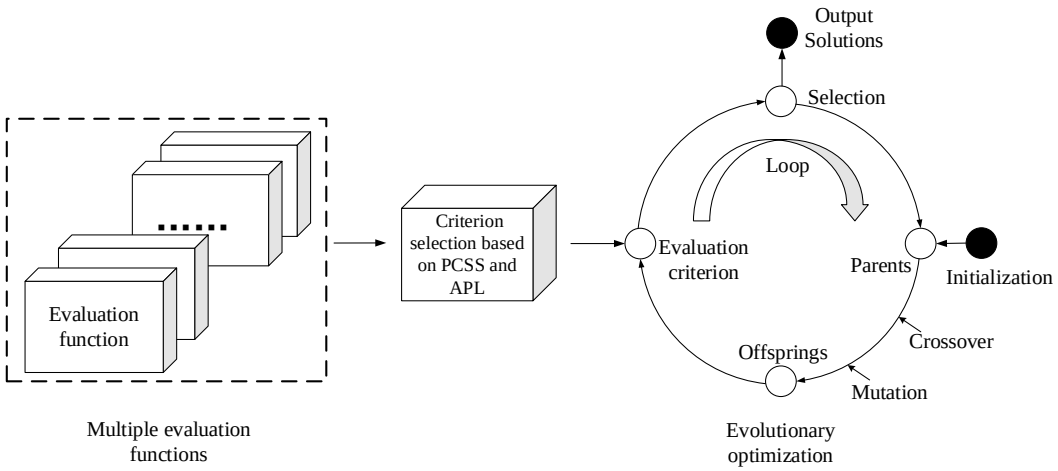


Figure 4.2 Framework of MO-MCEA

4.4.2.3 Probability-based Criterion Selection Strategy

The introduction of PCSS aims to select one evaluation function from multiple multi-objective evaluation functions as the current evaluation criterion for comparing individual fitness and population evolution. The concept behind PCSS is as follows: each multi-objective function has a criterion selection probability (denoted as *csp*). For instance, the criterion selection probability for multi-objective evaluation function F_i is represented as csp_i, where multi-objective functions with larger *csp* values will have a higher chance of being selected as the evaluation criterion. Consequently, given K multi-objective functions F_1, F_2, ..., and F_K, along with their respective selection probabilities csp_1, csp_2, ..., and csp_K, the criterion selection process will be implemented using a roulette wheel method based on csp_1, csp_2, ..., and csp_K. Mathematically, the criterion selection can be expressed as follows:

$$cid = roulette(csp) \tag{4.4}$$

Where *cid* is the index number of the selected criterion corresponding to the multi-objective function (i.e., the selected criterion represented as F_{cid}), and the roulette

function returns the index result based on the roulette wheel selection method. During initialization, the *csp* for different functions is set to 1/*K*, where *K* is the total number of functions, indicating that initially, each function has an equal probability of being selected.

4.4.2.3 APL Adaptive Parameter Learning

APL updates *csp* based on the improvement of each generation's population, allowing PCSS to select criteria more appropriately. Generally, if the population shows better improvement under the current criterion after one generation of evolution, then this criterion may be more suitable for the current evolutionary stage, and vice versa. Therefore, APL updates *csp* based on the population improvement of each generation. Specifically, if the *g*-th generation's population (denoted as P_g) uses the *k*-th multi-objective function (denoted as F_{cid}) as the evaluation criterion, then APL updates csp_{cid} using the following formula:

$$csp_k = \begin{cases} csp_k + \Delta, & \text{if } P_{g+1} \text{ is better than } P_g \\ csp_k - \Delta, & \text{otherwise} \end{cases} \tag{4.5}$$

Here, Δ is a fixed update amount, and "better" can be determined by comparing various indicators of P_{g+1} and P_g regarding the population performance over two generations. However, these indicators should not rely on the true Pareto front, such as hypervolume indicators. Traditional evaluation metrics like hypervolume often depend on the known Pareto front to assess the population's performance, which may not be accessible or determinable in MO-MTOP. Hence, MO-MCEA uses the *C* indicator to compare P_{g+1} and P_g. In addition to csp_k , other *csp* values (i.e., csp_j where $j \neq cid$) are updated accordingly:

$$csp_j = \begin{cases} csp_j - \dfrac{\Delta}{K-1}, & \text{if } P_{g+1} \text{ is better than } P_g \\ csp_j + \dfrac{\Delta}{K-1}, & \text{otherwise} \end{cases} \tag{4.6}$$

K is the total number of available multi-objective functions. If a csp_i is less than 0.1, it will be set to 0.1, and then all *csp* values are normalized to ensure that the *i*-th

function still has a probability of being selected again as the evaluation criterion, and the sum of all *csp* values equals 1.

4.4.2.4 The Complete MO-MCEA

Combining PCSS and APL, the complete pseudocode of MO-MCEA is presented in Algorithm 1. The search space for all multi-objective tasks will be unified within the range $[0, 1]^D$, where D is the maximum variable dimension among all tasks. That is, candidate solutions from different tasks will be mapped to $[0, 1]^D$, a common practice in the literature. After initialization, Algorithm 1 mainly consists of three repetitive procedures. These procedures involve determining the selected criterion through PCSS, population evolution, and parameter learning through APL, as depicted in lines 10 to 15, lines 16 to 18, and lines 19 to 20 of Algorithm 1, respectively. The population evolution procedure in Algorithm 1 can use various carefully designed operators, including efficient crossover, mutation, and selection operators, based on the user's preferences and needs. Thus, MO-MCEA can be extended with powerful state-of-the-art methods and operators to develop more effective algorithms. In this chapter, the crossover and mutation operators used are those from the NSGA-II algorithm. Additionally, the criterion for evaluation is switched by PCSS every G generation. Each time the criterion is switched, the current population is re-evaluated using the new criterion for its fitness before entering evolution. Therefore, NP fitness evaluations are required after each switch, where NP is the population size, as shown in lines 13 and 14 of Algorithm 1. Overall, the algorithm iteratively repeats PCSS, population evolution, and APL until the stopping criteria are met, such as the exhaustion of all available FEs. Furthermore, K sets (denoted as NDS_1, NDS_2, ..., and NDS_K) are used here to respectively record the current nondominated solutions for K tasks. During evolution, after one generation of population evolution using F_{cid} (i.e., line 16 of Algorithm 1), the corresponding NDS_{cid} is updated. That is, all solutions in the current population are merged with those in NDS_{cid}, and only the nondominated solutions in the merged set are retained in NDS_{cid}. After the evolution process concludes, all solutions in the final population are evaluated by K multi-objective functions respectively and

merged with each NDS to update them (similar to line 18), as shown in lines 23 to 24 of Algorithm 1. Finally, the algorithm outputs the best-nondominated solution sets NDS_1, NDS_2, ..., and NDS_K for all different tasks.

Algorithm 1: The Complete MO-MCEA

Input:	T_1, T_2, …, T_K - K optimization tasks;
	F_1, F_2, …, F_K - the fitness function of the T_1, T_2, …, T_K;
	NP - the number of individuals in the population;
	G - the number of generations for using each criterion.
Output:	$NDS1$, $NDS2$, …, $NDSk$ - The current Pareto sets for K tasks.
1	Begin
2	Initialize NP individuals;
3	Initialize NDS_1, NDS_2, …, NDS_k as empty sets;
4	**For** i = 1 to K
5	$csp_i \leftarrow 1/K$; // Initialize the criterion selection probability evenly
6	End For
7	$FEs \leftarrow 0$;
8	$gen \leftarrow 1$; // index of generation
9	**While** ($FEs + NP \times K <$ maximum number of available FEs) **Do**
10	If mod(gen, G) = =1
11	// Probability-based Criterion Selection Strategy
12	$cid \leftarrow$ roulette selection of [1, 2, …, K] with [csp_1, csp_2, …, csp_K];
13	Re-evaluate individuals by the function F_{cid};
14	FEs ← FEs + NP;
15	End If
16	Evolve population for one generation with F_{cid};
17	FEs ← FEs + NP;
18	Update and record the corresponding NDS_{cid};
19	// Adaptive Parameter Learning
20	Update [csp_1, csp_2, …, csp_K] according to Eq.(4.5) and Eq.(4.6);
21	gen←gen+1;
22	End While
23	Evaluate individuals by the K multi-objective functions F_1, F_2, …, and F_K;
24	Update and record NDS_1, NDS_2, …, NDS_k respectively;
25	End

4.4.3 Experiment

To evaluate the effectiveness and efficiency of the proposed algorithm, experiments were conducted comparing it with several state-of-the-art and recently developed

algorithms known for their good performance on six different MO-MTOP problems. Based on the degree of intersection and task similarity, the six problems with different characteristics were classified into different categories, indicated by labels at the problem IDs, such as (CI+HS). The algorithms used in the experiments included MO-MFEA, MO-MFEA-II, and MO-EMTA, along with MO-MCEA proposed in this study. For a fair comparison, all these algorithms used the same widely-used representative multi-objective optimization algorithm (i.e., NSGA-II) as the optimizer. Therefore, the difference between the proposed MO-MCEA and the three comparison algorithms lies only in their approaches to handling MO-MTOPs (e.g., MO-MCEA treats MO-MTOPs as MC-MTOPs, while MO-MFEA handles MO-MTOPs through a multi-factor method). Additionally, the original NSGA-II (i.e., solving each task independently) was also used as a benchmark algorithm, referred to as MO-STEA in the following content. All algorithm settings were consistent with their original papers. For NSGA-II operators, the settings were consistent with existing literature. Moreover, in MO-MCEA, G and Δ were set to 25 and 0.01, respectively. Additionally, the population size for each task was set to 50. Thus, the total population size for MO-MCEA and MO-STEA was 50, while for MO-MFEA, MO-MFEA-II, and MO-EMTA, the total population size was 100. Table 4.3 presents the comparative results of the experiments.

In Table 1, "+", "≈", and "-" respectively indicate that the MO-MCEA algorithm significantly outperforms the comparison algorithms, there is no significant difference, and the MO-MCEA algorithm significantly underperforms compared to the comparison algorithms. From Table 4.3, it can be observed that the MO-MCEA algorithm has a significantly higher number of functions where it outperforms the comparison algorithms than the number of functions where it underperforms. Moreover, the MO-MCEA achieves the highest number of problems with the optimal MSS indicator, indicating that the MO-MCEA algorithm exhibits the best overall performance.

4.4.4 Conclusion and Future Work

In this section, MO-MTOP is treated as MO-MCOP, and effectively solved using this approach. The effectiveness and advantages of treating MO-MTOPs as MO-

MCOPs are then analyzed.

Furthermore, research can be conducted to enhance the adaptability of PCSS and the efficiency of APL to further improve MO-MCEA. Additionally, MO-MCEA can be further extended to handle complex real-world applications.

Table 4.3 Comparisons between the proposed MO-MCEA and state-of-the-art algorithms

Problem			MO-MCEA	MO-MFEA	MO-MFEAII	MO-EMTA	MO-STEA
MO-MTOP1 (CI+HS)	Task1 (T_1)	Mean	1.68E-01	6.05E-01(+)	8.50E-01(+)	1.46E+00(+)	2.14E+00(+)
		Std.	9.46E-02	2.08E-01	3.29E-01	9.34E-01	1.13E+00
	Task2 (T_2)	Mean	9.35E-01	4.94E+00 (+)	2.09E+00 (+)	2.53E+00 (+)	2.43E+00(+)
		Std.	3.28E-01	1.24E+00	5.17E-01	7.92E-01	5.89E-01
	MSS		**-5.99E+01**	3.31E+01	-1.59E+01	1.18E+01	3.09E+01
MO-MTOP2 (CI+MS)	Task1 (T_1)	Mean	1.44E-02	3.56E-02(+)	3.14E-01(+)	3.18E-01(+)	2.70E-01(+)
		Std.	4.27E-03	1.39E-02	2.09E-01	1.47E-01	1.32E-01
	Task2 (T_2)	Mean	1.20E-01	1.82E-01(+)	1.78E-01(+)	1.84E-01(+)	1.80E-01(+)
		Std.	-5.10E+01	4.33E-02	4.09E-02	4.60E-02	4.32E-02
	MSS		**-5.10E+01**	-1.87E+01	2.40E+01	2.75E+01	1.82E+01
MO-MTOP3 (PI+HS)	Task1 (T_1)	Mean	3.04E-01	4.53E-01(+)	2.35E+00(+)	4.07E+00(+)	5.25E+00(+)
		Std.	5.35E-01	4.23E-01	2.50E+00	2.39E+00	1.87E+00
	Task2 (T_2)	Mean	9.79E-01	9.70E-01(+)	9.17E-01(+)	9.16E-01(+)	9.16E-01(+)
		Std.	5.10E-01	3.02E-01	2.13E-02	2.16E-02	2.18E-02
	MSS		**-2.04E+01**	-1.96E+01	-4.12E+00	1.54E+01	2.87E+01
MO-MTOP4 (PI+HS)	Task1 (T_1)	Mean	3.74E+02	5.52E+00(-)	4.01E+00(-)	2.92E+00(-)	3.65E+00(-)
		Std.	5.55E+02	2.10E+00	1.69E+00	8.83E-01	1.12E+00
	Task2 (T_2)	Mean	9.75E+02	3.94E+01(-)	2.85E+01(-)	2.69E+01(-)	2.77E+01(-)
		Std.	6.44E+02	1.20E+01	8.26E+00	1.03E+01	7.34E+00
	MSS		7.84E+01	-1.89E+01	-1.97E+01	**-2.00E+01**	-1.98E+01
MO-MTOP5 (PI+MS)	Task1 (T_1)	Mean	3.87E+02	3.95E+00(≈)	3.31E+00(≈)	2.83E+00(≈)	3.70E+00(≈)
		Std.	5.82E+02	1.56E+00	1.44E+00	1.09E+00	1.20E+00
	Task2 (T_2)	Mean	1.01E+03	3.63E+01(-)	2.65E+01(-)	2.88E+01(-)	2.84E+01(-)
		Std.	6.98E+02	9.43E+00	7.40E+00	9.27E+00	1.01E+01
	MSS		7.75E+01	-1.89E+01	**-1.96E+01**	-1.95E+01	-1.94E+01
MO-MTOP6 (PI+LS)	Task1 (T_1)	Mean	3.18E-01	3.76E-01(+)	2.69E-01(+)	2.46E-01(+)	2.63E-01(+)
		Std.	3.29E-01	6.91E-02	6.20E-02	5.93E-02	5.47E-02
	Task2 (T_2)	Mean	1.75E+01	5.71E+00(-)	1.17E+01(-)	5.08E+00(-)	1.99E+01(+)
		Std.	3.22E+00	1.33E+00	6.11E+00	2.25E+00	2.39E+00
	MSS		2.80E+01	-1.23E+01	-5.69E+00	**-3.86E+01**	2.86E+01
Number of +/≈/-			NA	7/1/4	7/1/4	7/1/4	8/1/3
Number of best MSS			3	0	1	2	0

4.5 Conclusion

This chapter analyzes MTOP characteristics and current research. MCEA was proposed, treating MTOP as multi-criteria problems. Furthermore, the complexity was increased by optimizing the challenges of both multi-objective and MTOP together, leading to the proposition of solving MO-MTOP using the MO-MCEA method. Experimental results indicate that this could be a potential direction for addressing MTOPs.

Regarding future research directions, dynamic multi-task optimization is also a promising area of study. Additionally, attention could be directed towards the practical application of multi-task optimization in engineering problems, such as logistics, supply chain, network optimization, sensor optimization, and traffic optimization, among others.

The content of this chapter, covering the research in Sections 4.3 and 4.4, has been published respectively in the International Conference of Gecco 2021 and the journal Complex & Intelligent Systems.

CHAPTER 5

A KNOWLEDGE LEARNING MEMETIC ALGORITHM FOR USER ROUTE PLANNING IN BIKE-SHARING SYSTEM

5.1 Introduction

Bike-sharing is a service that allows the public to rent bicycles for use, eliminating the need for users to carry their own bikes at all times. Due to its environmentally friendly nature, affordability, independence from fixed public transportation routes, and avoidance of congested road sections, bike-sharing systems are increasingly chosen by people as a means of transportation [199]-[213]. With the widespread use of mobile internet and smartphones, bike-sharing apps provide convenient services for locating, returning, route planning, maintenance, and bike fleet management. For ordinary citizens, bike-sharing has become an integral part of urban public transportation, alongside subways, buses, and ride-hailing services. The high popularity and systematic management of bike-sharing have brought about a series of optimization problems, such as bike redistribution [204]-[208], bike retrieval [209]-[211], and bike lane planning [212]-[214]. A more user-friendly and convenient bike-sharing system can attract more users, thereby increasing the profitability of bike-sharing companies, reducing energy consumption during travel, and contributing to carbon neutrality and smoother traffic flow in cities. Therefore, research on bike-sharing is of great significance for the development of smart cities [216].

However, as mentioned earlier, existing research has mostly focused on the bike-sharing redistribution problem, with limited attention given to the User Route Planning Problem (URPP). Yet, appropriate route planning is crucial for bike-sharing users. In current bike-sharing systems, after paying a certain membership fee, users can enjoy free rides for a period of time. For short-distance trips, route planning needs to consider nearby pick-up and drop-off locations, while for medium to long-distance trips, users may encounter transfer issues if they wish to continue enjoying the free membership

service. Regarding this issue, Zhang et al. proposed to transform URPP into a network flow problem, eliminating all illegal routes through pruning techniques, and then optimizing the travel cost using 0-1 integer programming methods [217]. However, this chapter only considers static scenarios; that is, it assumes that bike station inventory is always sufficient [217]. In reality, some bike stations may lack bikes, making it impossible to arrange them as intermediate stations for cycling routes, while some bike stations without bikes may become available again due to other passengers returning bikes. In other words, the availability of bike stations is dynamic. This dynamic aspect of bike station availability constitutes the research gap in URPP. In real-life scenarios, URPP is a DOP, where the number of bikes at bike stations is subject to change. Ignoring the dynamic availability of bike stations makes it difficult for URPP to have practical significance.

Regarding the route planning problem, vehicle routing optimization is a typical path planning problem, and there has been a large amount of research on similar issues, such as private car and ride-hailing route optimization. However, compared to typical vehicle routing optimization problems, there are significant differences in route optimization for bike-sharing. This is because, in the URPP as a DOP, the difficulty lies not only in the changing availability of bikes at stations but also in the variable number of stations users may visit during their journey. For example, during a 20-kilometer ride, a user may switch bikes 2 or 3 times, leading to variable-length encodings. These different encoding lengths pose challenges for the crossover and mutation operations in EC. Additionally, the URPP differs from classical Traveling Salesman Problems due to these variable-length encodings, presenting challenges in achieving optimal solutions.

Therefore, this chapter models URPP as a discrete optimization problem with constraints that can be used in not only static situations but also dynamic situations. Specifically, in the proposed URPP model, discrete variables are used to encode each bike station, and the solution to the problem is composed of bike station indexes. This encoding method can intuitively and clearly show the intermediate stations and the complete riding route.

Based on the two research challenges mentioned above, our approach is as follows.

First, addressing the encoding difficulty, the URPP is modeled as a constrained discrete optimization problem, applicable to both static and dynamic scenarios. Specifically, in the proposed URPP model, discrete variables are used to encode each bike station, and the solutions to the problem consist of indices of bike stations. This encoding method provides an intuitive and clear representation of intermediate stations and complete biking routes. Methods for handling variable encoding lengths are also provided.

Secondly, for the route planning optimization algorithm, a novel and efficient MA within the EC framework is selected. As an efficient optimization algorithm in the EC field, the MA combines Genetic Algorithms with local search methods, continuously introducing new "memes" (knowledge fragments) to balance the global and local search for more effective exploration of solution space [218]. Existing MA algorithms have been effectively utilized to address various optimization challenges such as linear ordering problems [219], traveling salesman problems [220], and high-dimensional feature selection problems [221]. In MA, EC is typically employed for global search owing to its strong global search capability and diversity of parallelism [222]-[224]. Therefore, EC algorithms perform well on many complex optimization problems [225]-[227], including single-objective optimization [228]-[229], multi-objective optimization [230]-[232], multi-task optimization [233]-[236], multimodal optimization [237]-[240], expensive optimization [241]-[243], and DOP [244]-[247]. Thus, EC algorithms are used as the global optimizer to compose the MA. Local search in MA is aimed at improving the solutions obtained by the global optimizer. However, as mentioned earlier, the challenge of the variable solution dimensionality in URPP means that the optimal solution dimensionality is uncertain, implying that different biking routes may have different numbers of stations (i.e., different solutions have different dimensions). This makes traditional EC operators such as crossover, mutation, and local search unsuitable for URPP. Therefore, knowledge learning-based crossover (KLC) is proposed, random pruning-based mutation (RPM), and two-phase local search

(TPLS) to develop knowledge learning and random pruning-based MA (KLRP-MA), making a contribution to developing more suitable operators for efficient MA to solve URPP. Specifically, KLC allows each individual to cross with the current best individual, thereby learning solution knowledge from the current best individual, including dimensionality scale knowledge (i.e., how many stations to choose) and station selection knowledge (i.e., which stations to choose). RPM shortens the individual length by randomly pruning redundant stations to perform mutation on each individual. Additionally, TPLS can use two types of local search in two phases to further improve individual quality. Therefore, KLRP-MA is a typical and efficient MA, and the first MA proposed for solving URPP.

Finally, addressing the DOP, this chapter considers not only solving URPP in static scenarios but also in dynamic scenarios. This means that the inventory of bike stations may change, leading to changes in the availability status of bike stations (e.g., if there are no bikes available, the bike station will be unavailable). Therefore, dynamic inventory will result in the activation and deactivation of bike stations during user biking, and KLRP-MA may not work properly in dynamic scenarios. Consequently, this chapter further proposes a dynamic version of KLRP-MA (referred to as DyKLRP-MA) to solve URPP in dynamic scenarios. The proposed DyKLRP-MA uses the operators proposed in KLRP-MA (i.e., KLC, RPM, and TPLS) to efficiently find optimal solutions in changing environments, thus responding quickly to dynamic changes. Furthermore, depending on the type of dynamic change, the optimal solutions from previous environments will be reused to help obtain the best solutions in new dynamic environments more quickly, thereby further improving the algorithm's performance in dynamic scenarios, i.e., through KT, the reuse of historical information. Therefore, DyKLRP-MA is suitable for solving URPP in dynamic scenarios.

The innovations and contributions of this chapter include the following aspects:

1. A novel KLRP-MA algorithm is proposed, which efficiently solves URPP by utilizing KLC and RPM for global optimization and TPLS for local optimization. The proposed KLRP-MA is efficient and the first MA designed to solve URPP.

Experimental results demonstrate that the proposed KLRP-MA can find optimal solutions for URPP in a short time.

2. Unlike existing URPP models designed for static scenarios, the URPP model proposed in this chapter is applicable not only to static scenarios but also to dynamic scenarios. Building upon KLRP-MA, DyKLRP-MA is introduced to address URPP in dynamic scenarios, enabling rapid response to dynamic changes and enhancing the algorithm's performance under dynamic conditions.

The subsequent sections of this chapter are organized as follows. Section 5.2 illustrates the proposed URPP model. Then, Section 5.3 provides a detailed description of the proposed KLRP-MA method. Section 5.4 describes DyKLRP-MA for dynamic scenarios. Section 5.5 presents and analyzes the experimental results. Finally, Section 5.6 concludes the findings and delineates future prospects.

5.2 URPP Model

For membership users of the bike-sharing app, riding bicycles for a certain period (e.g., 30 minutes) incurs additional charges [217]. To avoid extra fees, users need to change bicycles during their ride, while also getting adequate rest and supplies. Minimizing the total riding time for membership users without incurring additional charges requires proper route planning. For instance, the biking route $a \rightarrow b \rightarrow c \rightarrow d \rightarrow h$ depicted in Figure 5.1 may result in a longer time compared to a direct biking route $a \rightarrow h$. Therefore, it is necessary to plan reasonable routes for users. Such a problem is referred to as User Route Planning Problem (URPP).

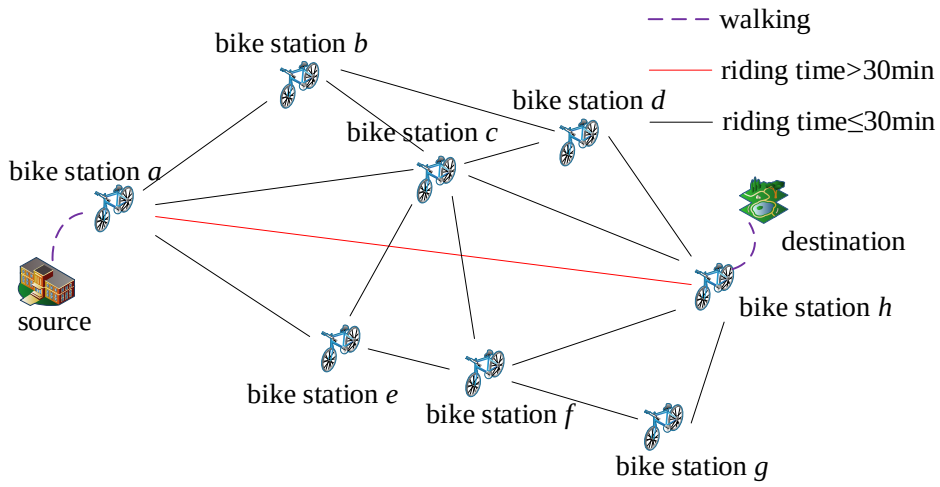


Figure 5.1 An example of riding route planning

The objective of URPP is to minimize travel time without paying additional fees. Given the user's starting point, ending point, and all available bike stations between the starting point and the ending point, the model for URPP is formulated as

$$\begin{aligned} &\min\ f(x) = walk\_time(x) + ride\_time(x) \\ &\text{s.t. } ride\_time_{(x_i, x_{i+1})} \le threshold\ \ \forall i = 1, 2, ..., D-1 \end{aligned} \tag{5.1}$$

where x=[x_1, x_2, …, x_D] represents a planning riding route (i.e., a candidate solution). That is, the user starts from the available bike Station x_1, rides to the available bike Station x_2 to change the bike, …, and finally rides to the available station x_D, where D represents the solution dimensions (i.e., the number of bike stations in this solution). $f(x)$ represents the objective function, and the value of f is called the function fitness. *walk_time* represents the walking time from the starting point *source* to x_1 and the walking time from x_D to the ending point *dest*, *ride_time* represents the riding time from x_1 to x_D. The specific calculation formulas are shown as

$$\begin{aligned} walk\_time(x) &= walk\_time_{(source, x_1)} + walk\_time_{(x_D, dest)} \\ &= \frac{dist(source, x_1)}{walk\_speed} + \frac{dist(x_D, dest)}{walk\_speed} \end{aligned} \tag{5.2}$$

$$\begin{aligned} ride\_time(x) &= \sum_{i=1}^{D-1} ride\_time_{(x_i, x_{i+1})} + transfer\_time(x) \\ &= \sum_{i=1}^{D-1} \frac{dist(x_i, x_{i+1})}{ride\_speed} + num\_transfer \times time_{one\_transfer} \end{aligned} \tag{5.3}$$

where *dist*(*m*, *n*) represents the distance between location *m* and location *n*, and *walk_speed* and *ride_speed* represent the walking speed and riding speed, respectively. When calculating the riding time, the time for transferring the bike is also considered, which is the multiply of the number of transfers (denoted as *num_transfer*) and the time of each transfer (denoted as $time_{one\_transfer}$). As a kind of routing problem, the URPP is similar with the vehicle routing problem, while the URPP is different in that it does not require to reach all bike stations in the planned route (the vehicle routing problem requires that all customers are served in the planned route) and therefore the optimal dimension of the solution is uncertain. Moreover, in the URPP, the availability of bike stations may change in a dynamic situation.

5.3 KLRP-MA Approach

5.3.1 Encoding Scheme

The planned riding route $x=[x_1, x_2, \ldots, x_D]$ is a D-dimensional vector, which shows that the user starts from the available bike Station x_1, rides to the transfer bike Station x_i (i=2, …, D–1), and finally arrives at the available bike station x_D. Note that the first dimension of each solution in the population is the available bike station *start_bs* nearest to the starting point *source*, and the last dimension is the available bike station *end_bs* nearest to the ending point *dest*. It is worth noting that the dimensions of different solutions may be different (i.e., different riding routes may contain different numbers of bike stations). To avoid frequent transferring, the dimensions of all solutions cannot exceed the maximum dimension D_{max}. The calculation formulas of D_{max} are shown as Eq.(5.4) and Eq.(5.5).

$$D_{candi} = 6 \times \left\lceil \frac{num\_bs}{18} \right\rceil \tag{5.4}$$

$$D_{max} = \min(num\_bs, D_{candi}) \tag{5.5}$$

where D_{candi} represents the candidate value of the maximum dimension, and *num_bs* represents the number of available bike stations between the starting point and the ending point. To avoid the dimensions of the solution being larger than the number

of available bike stations, D_{max} is set as the minimum between *num_bs* and D_{candi}.

5.3.2 Population Initialization

First, the KLRP-MA initializes the first dimension of all individuals in the population as *start_bs*. Second, the KLRP-MA generates a candidate set cs_{ind} for each individual *ind* in the population, where cs_{ind} stores all the available bike stations (except for *start_bs* and *end_bs*). Then, for i=2, …, D_{max}–1, the KLRP-MA generates a random number r_i among (0, 1). If r_i is smaller than the generation probability *Pg*, the KLRP-MA randomly selects an available bike station in cs_{ind} to add to the end of the solution vector (i.e., the end of the current riding route) and then removes the selected bike station from cs_{ind}. Otherwise, no bike station will be added to the solution vector. Finally, the KLRP-MA adds *end_bs* to the end of the solution vector. Note that the individuals who violate constraints will be reinitialized.

Figure 5.2 illustrates the individual initialization process in the situation of Figure 5.1. As the available bike Station *a* is nearest to the starting point, the first dimension of *x* is set as *a*. Assume that the randomly generated numbers r_3 and r_5 are less than *Pg*; therefore, two available bike stations are randomly selected from *cs* (i.e., bike Station *f* and bike Station *d*) to assign to *x*. Finally, the available bike Station *h* is added to the end of the solution vector to generate an initial planning route *a*→*f*→*d*→*h*.

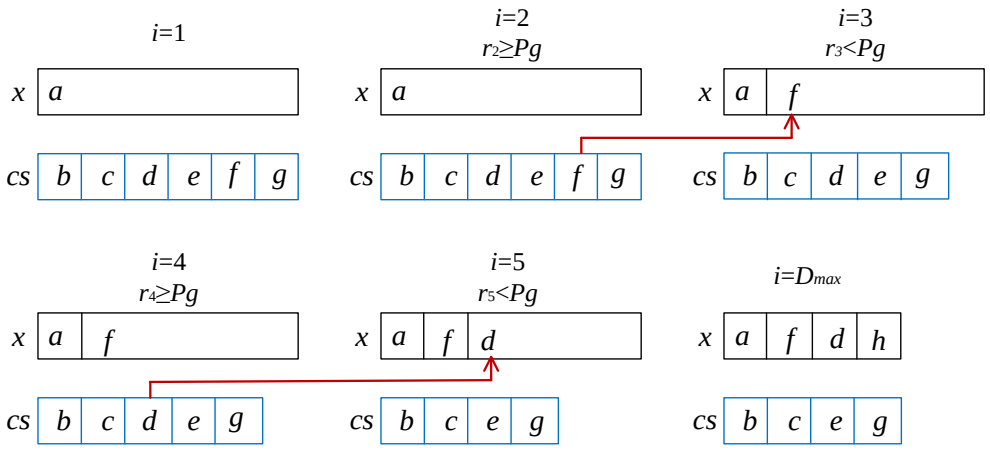


Figure 5.2 Illustration of individual initialization.

Due to the characteristics of encoding, different solutions may have different dimensions (i.e., a different number of used bike stations), making it impossible to

directly use the traditional crossover operator and mutation operator to generate offspring. Therefore, this chapter proposes the KLC and the RPM. Each individual (solution) in the population generates offspring through these two strategies.

5.3.3 Knowledge Learning-based Crossover

The KLC aims to help individuals learn the knowledge from the current-best individual to generate more promising individuals. In the literature, learning and transferring solution knowledge via evolutionary operators have attracted increasing attention and have been successful in various optimization problems [235], especially in problems with variable-sized dimensions [236][246][247]. As the URPP is also a problem with uncertainty dimensions, the KLC is proposed to help the individual learn and transfer the solution knowledge from the global best individual.

Unlike EC algorithms in existing research that directly consider the knowledge of solution value, KLC involves dimension scale knowledge and station selection knowledge. For example, if the best-performing individual in the population has five dimensions, represented as *x_best*=[b, c, f, g, h], the remaining individuals in the population will learn the dimension scale (i.e., 5) from *x_best*, which is Dimension Adaptation, and the knowledge of station selection (i.e., b, c, f, g, and h), which is Station Transfer. Therefore, KLC generates new individuals using two steps: learning dimension scale knowledge for dimension adaptation and learning station selection knowledge for station transfer. If the size of the current individual is different, size adaptation adjusts the size of the current individual to approximate the size of the best individual, as described in the following three cases: when the individual's dimension is greater than, equal to, or less than the optimal individual's dimension. After dimension adaptation, bike station information is transferred from the route represented by *x_best* to the route represented by the current individual to achieve KT.

To provide a better illustration, the KLC for an individual, denoted as x_{ind}, is given as an example in the following, with the pseudocode given as Algorithm 1.

Algorithm 1: KLC(x_{ind}, x_{best})

Input: x_{ind}(the current individual), x_{best}(the current best individual)
Output: x_{new}(the new individual generated by crossover with x_{ind} and x_{best})

```
1  Begin
2  di ← the dimension of x_ind;
3  db ← the dimension of x_best;
4  x_new ← x_ind;
5  Flag ← True; // to indicate whether the x_new will be feasible
6  If di ≠ db Then
7    If di>db Then // the dimension of x_ind is larger than x_best
8      i← a random integer in [1, di-1]; // a random dimension of x_ind
9      j← a random integer in [1, db-1]; // a random dimension of x_best
10     For k=1 to db
11       If x_best,j or x_best, j+1 equal to an element in x_ind except x_ind,i and x_ind,i+1 Then
12         Flag ← False; // the crossover will make the generated x_new infeasible
13       End If
14       If Flag=True Then
15         break the For loop;
16       End If
17       j←((j+1)%db)+1; // select next random dimension of x_best
18     End For
19     If Flag=True Then // the generated x_new can be feasible
20       Compress x_new,i and x_new,i+1 to be one element as x_best,j;
21       Evaluate x_new and update its fitness;
22     End
23   Else // the dimension of is x_ind smaller than x_best
24       i← a random integer in [1, di-1]; // a random dimension of x_ind
25       j← a random integer in [1, db]; // a random dimension of x_best
26     For k=1 to db
27       If x_best,j equals to an element in x_ind Then
28         Flag ← False; // the crossover will make the generated x_new infeasible
29       End If
30       If  Flag=True Then // the generated x_new can be feasible
31         break the For loop;
32       End If
33         j←((j+1)%db)+1; // select next random dimension of x_best
34     End for
35       If Flag=True Then // the generated x_new can be feasible
36         Insert x_best,j between x_new,i and x_new,i+1;
37         Evaluate x_new and update its fitness;
38       End
39       End If
40   Else // the dimension of is x_ind equal to x_best
41       i← a random integer in [1, di-1]; // a random dimension of x_ind
42       j← a random integer in [1, db-1]; // a random dimension of x_best
43     For k=1 to db
44       If x_best,j or x_best, j+1 equal to an element in x_ind except x_ind,i and x_ind,i+1 Then
45         Flag ← False; // the crossover will make the generated x_new infeasible
46       End If
47       If Flag=True Then
48         break the For loop;
49       End If
50       j←((j+1)%(db-1))+1; // select next random dimension of x_best
51     End For
52     If  Flag=True Then // the generated x_new can be feasible
53       Replace x_new,i and x_new,i+1 with x_best,j and x_best,j;
54       Evaluate x_new and update its fitness;
55     End
56   End If
57 End
```

1) If the dimension of x_{ind} (e.g., x_{ind}=[a, c, b, e, g, h]) is greater than the dimension of x_{best} (i.e., [b, c, f, g, h]), the dimension adaptation will randomly select two adjacent intermediate bike stations in x_{ind} and compress them into one new bike station, so as to reduce the dimension of x_{ind} to get close to the dimension of x_{best}. For example, as shown in Figure 5.3(a), b and e in x_{ind} are compassed to generate the new individual, which can be represented by x_{new}=[a, c, ?, g, h]. Then, to determine the station of "?", a random intermediate bike station (which is also not in the compressed x_{ind}) will be transferred from x_{best} to "?" in x_{new}. For example, the "f" in x_{best} will be transferred to x_{new}, and the final x_{new} is finally [a, c, f, g, h].

2) If the dimension of x_{ind} (e.g., x_{ind}=[a, b, c, h]) is smaller than the dimension of x_{best} (i.e., [b, c, f, g, h]), then the dimension adaptation will extend with a random position between the begin and end station to become x_{new}, e.g., x_{new}=[a, b, c, ?, h]. Then, as shown in Figure 5.3, similar to the procedure in 1), a random intermediate bike station (which is also not in the x_{ind}) will be transferred from x_{best} to determine the "?" in x_{new}, e.g., transfer the "f" in x_{best} to x_{new} to obtain x_{new} =[a, b, c, f, h].

3) If the dimension of x_{ind} (e.g., x_{ind}=[a, b, f, d, h]) is equal to the dimension of x_{best} (i.e., [b, c, f, g, h]), then the dimension adaptation does not need to be conducted. In this case, two random two adjacent intermediate bike stations in x_{best} will be transferred to replace two random adjacent intermediate bike stations in x_{ind} to generate x_{new}. Note that, if the replacement makes the x_{new} infeasible, the replacement is not carried out. For example, as shown in Figure 5.3(c), b and f in x_{ind}=[a, b, f, d, h] are replaced by f and g transferred from x_{best}, and the final generated x_{new} is [a, c, f, d, h].

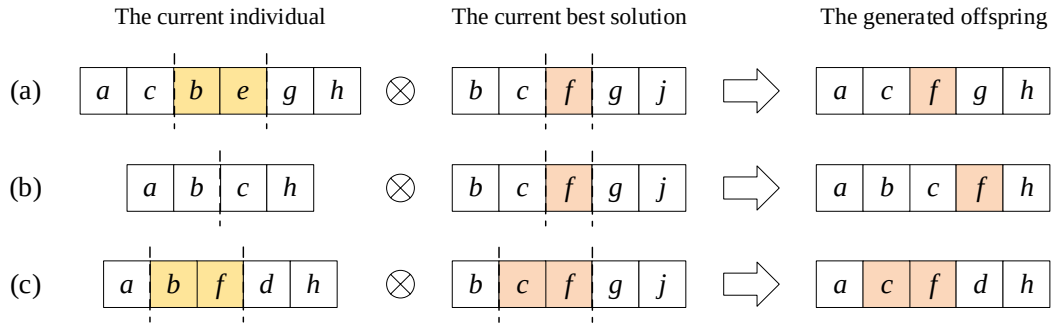


Figure 5.3 Three examples of KLC.

After the above, if no feasible x_{new} can be generated, the x_{new} is set as x_{ind}. Otherwise,

the newly generated feasible solution x_{new} is evaluated and replaced the x_{ind} if the fitness value of x_{new} is better.

5.3.4 Random Pruning-based Mutation

The purpose of RPM is to remove unnecessary station selections from the current individual to reduce the total travel time of the route. This operation involves randomly selecting an intermediate bike station from the current individual and deleting it to generate a new individual. Similar to KLC, if the newly generated individual is feasible and its fitness value improves, the original individual will be updated with the newly generated solution. It is important to note that RPM is only applied to solutions with dimensions greater than 3.

5.3.5 Two-phase Local Search

The KLRP-MA contains not only the global optimizer via KLC and RPM, but also the local optimizer named TPLS. The TPLS contains two local search phases to further enhance the current individuals, where each phase considers the local search in different search spaces. The first phase considers the local search within the selected bike stations in the individual. Specifically, given an individual, two intermediate bike stations in the individual (i.e., excluding the begin and end stations) are selected randomly and then their positions are exchanged to generate a new individual. The second phase considers all the bike stations that have not been selected by the individual, which randomly selects a bike station in the individual and then replaces it with the nearest available bike station.

Each individual goes through the above two phases in turn. After each phase, if the newly generated individual is feasible and has a better fitness value, the newly generated individual will be used to update the original individual. Otherwise, the newly generated individual will be discarded. Note that the first phase is only performed on the solutions whose dimension is greater than three.

5.3.6 The Completed Algorithm

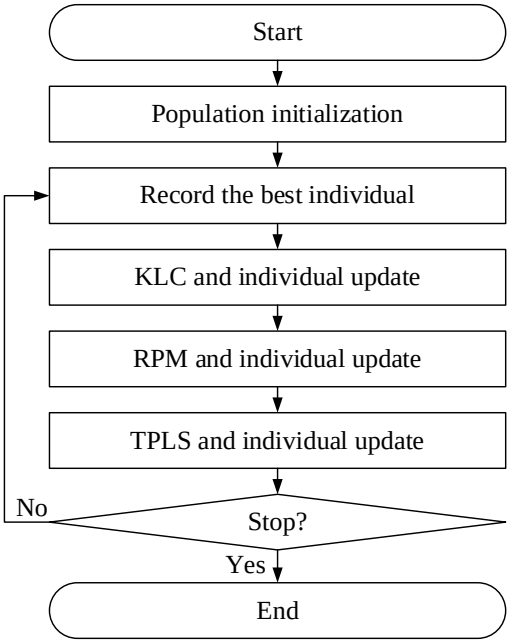


Figure 5.4 The Flowchart of KLRP-MA.

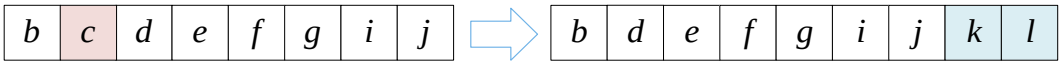
Figure 5.4 is the flowchart of the KLRP-MA. First, KLRP-MA initializes and evaluates the individuals. Then, the KLRP-MA enters the main loop of the algorithm, that is, iteratively performs the KLC, RPM, and TPLS one by one. The main loop will be repeated until the stop conditions are met. Finally, the solution with the best fitness value is output as the final planning route.

5.4 DyKLRP-MA Approach

In the practical situation, the bikes at available bike stations may be used by other users, making the available bike stations unavailable due to insufficient inventory. In addition, some unavailable bike stations may become available again because of the bikes newly parked by other users. Therefore, the number of available bike stations will be dynamically decreased and increased during user riding. If the unavailable bike stations belong to the adopted planning route, the planning route will become invalid. In addition, if some bike stations become available during the riding time, they may be used in the riding route to shorten the traveling time. In this case, the original best solution obtained by KLRP-MA cannot remain the best solution. To solve the above issues, this chapter proposes the DyKLRP-MA for solving the URPP in a dynamic situation.

The idea to solve this dynamic situation is as follows. First, the planning route is obtained under the initial situation by executing KLRP-MA. Then, if the availability of the bike stations changes, execute DyKLRP-MA to update the planning route; otherwise, ride according to the current planning route.

DyKLRP-MA uses the set S_c to store the bike stations that are accessible in the current situation, which excludes those bike stations that have been visited. After every situation changes, all the newly disabled bike stations will be deleted from S_c, and all the newly enabled stations will be added to S_c, which results in the set S_n that stores the available bike stations in the new situation. If the current planning route is affected by the dynamic change (i.e., some unvisited bike stations of the current planning route are disabled or some candidate bike stations are enabled), the route re-planning process will be immediately triggered.



the S_c for the current situation　　the S_n for the new situation

Figure 5.5 Illustration of S_c and S_n for the current situation and new situation.

Figure 5.5 gives an example to show how to update S_c to S_n. Given ten available bike stations (denoted as $a \sim j$), and the current planning route x_{best_c}=[a, h, j, g, c]. Assuming the user has reached the available bike Station h, and the availability of the bike stations does not change in the current static situation, then S_c is {b, c, d, e, f, g, i, j} as shown in Figure 5.5(a). When the user is riding from h to j, bike Stations a and c are disabled, while bike Stations k and l are enabled. As a does not exist in S_c, only c will be deleted from S_c. Moreover, k and l are added to S_c to form S_n, and then the final S_n is {b, d, e, f, g, i, j, k, l}, as shown in Figure 5.5(b).

Algorithm 2 is the pseudocode of DyKLRP-MA. First, DyKLRP-MA initializes *Dy_N* individuals to be the population Q, and fixes the first *n_index* dimensions of all individuals to be the same as the first *n_index* dimensions of x_{best_c} (the *n_index* is the index of the currently arriving bike station in x_{best_c}), because the first *n_index* stations have already been visited and do not need to change. Moreover, the last dimension of all individuals is fixed as the available bike station nearest to the destination (i.e.,

end_bs) in the updated *available_bs*. Then, the remaining dimensions of the individuals are reinitialized by the population initialization method mentioned in Section 5.3.2 with the candidate set *cs* as the *available_bs* without *end_bs*. Note that in the dynamic situation, the maximum dimensionality of the individual is still limited by D_{max} as mentioned in Eq. (5.5). After the initialization, the fitness of each individual in the population Q is calculated (as shown in Line 2 of Algorithm 1), and then the KLC, RPM, and TPLS are performed one by one to evolve the population Q iteratively (as shown in Line 3 to Line 7 of Algorithm 1). When the number of iterations exceeds Dy_T_{max}, the best solution x_{best_n} is obtained as the planning route in the new situation. Note that if S_c (the station set before change) is the subset of S_n (the station set after change), the best solution for the current situation (i.e., x_{best_c}) is still feasible for the new situation. Therefore, in this case, Algorithm 1 will replace x_{best_n} with x_{best_c} if x_{best_c} has a better fitness than x_{best_n}, as shown in Lines 10-16 of Algorithm 1.

Algorithm 2: DyKLRP-MA(S_c, S_n, x_{best_c}, Dy_N, Dy_ T_{max})

Input: S_c (station set for current situation), S_n (station set for the new situation),
x_{best_c} (the best planning route in the current situation),
Dy_N (population size in the new situation),
Dy_T_{max} (the maximum number of iterations)
Output: x_{best_n} (the best planning route in the new situation)
1: Initialize Dy_N individuals as the population Q;
2: Evaluate individuals in Q;
3: For T=1 to Dy_T_{max}
4: *global_best* ← the best individual in Q;
5: Perform KLC;
6: Perform RPM;
7: Perform TPLS;
8: End For
9: x_{best_n} ←the current best individual in Q;
10: **If** $S_c \subseteq S_n$ **Then** //the x_{best_c} is still feasible
11: **If** the fitness of x_{best_n} is better than the fitness of x_{best_c} **Then**
12: $x_{best\_n} \leftarrow x_{best\_c}$;
13: End If
14: End If
15: Return x_{best_n}

5.5 Experimental Results

5.5.1 Experimental Design

Both medium-distance and long-distance instances are considered in the experiments. The distance between the starting point and the ending point (i.e., *distance*) is set to 12 km for the medium distance and 20 km for the long-distance. The starting point is set to (0, 0) and the ending point is set to (*distance*, 0). The bike stations are randomly distributed on the map with a size of [0, *distance*]×[-6, 6].

The experiments are divided into static situations and dynamic situations. In the static situation, experiments are tested on both the medium-distance and long-distance instances. In the dynamic situation, experiments are tested on the long-distance instances to better challenge the proposed algorithm. The test instances at different distances are divided into small-scale, medium-scale, and large-scale categories, where the number of available bike stations (i.e., *num_bs*) is set as 10, 20, and 50, respectively. The experiments at each scale contain 30 randomly generated test instances, that is, 30 different bike station distributions. For clarity and simplicity, each instance is named in the format of A-B-C, where A represents the medium-distance (i.e., A=M) or long-distance (i.e., A=L), B represents the small-scale (i.e., B=S), medium-scale (i.e., B=M), and large-scale (i.e., B=L) category, and C represents the index of the instance in the corresponding category. For example, M-S-1 represents the first test instance in the medium-distance and small-scale instance categories.

For the fitness function of each instance problem (refer to Eq. (5.1) and Eq. (5.3)), the *threshold* is set to 30 min, *walk_speed* is set to 5 km/h, *ride_speed* is set to 18 km/h, and $time_{one\_transfer}$ is set to 1 min. Note that the unit of distance is the kilometer and the unit of time is the minute for calculating the function fitness. For KLRP-MA, Pg is set to 0.8, population size N is set to 20, and the maximum number of iterations T_{max} is set to 50. For DyKLRP-MA, Dy_N is set to 20, Dy_T_{max} is set to 20, and the remaining parameters are the same as those in KLRP-MA. All algorithms run 20 times independently on each test instance and the statistical results are used for analysis.

Moreover, two parameters are used to generate the bike station change in dynamic

situations, including P_{first_change}=0.8 which indicates the probability of bike station changes, and P_{second_change}=0.7, which indicates the probability of a second change after the first change. The process of the dynamic change is as follows: If a random number $r_1 \geq P_{first\_change}$, then no change will occur. Otherwise, if $r_1 < P_{first\_change}$ and another random number $r_2 \geq P_{second\_change}$, o_1 bike stations will be randomly enabled or disabled (but not both), while if $r_1 < P_{first\_change}$ and another random number $r_2 < P_{second\_change}$, o_1 bike stations will be enabled and o_2 bike stations will be disabled, where both o_1 and o_2 are random integers belonging to [1, o_{max}], and o_{max} represents the maximum number of changed bike stations and can be represented as

$$o_{max} = \left\lceil \frac{num\_current\_bs}{10} \right\rceil \tag{5.6}$$

where *num_current_bs* represents the number of available bike stations in the current situation. Note that each newly enabled bike station will be randomly distributed in the [0, *distance*]×[-2, 2].

5.5.2 Experimental Results under Static Situations

As this Chapter is the first to propose the MA to solve the URPP, a greedy algorithm and an enumeration method are used for comparison. Moreover, as the dimension of the optimal solution is unknown, the implementations of the greedy method and the enumeration method are described as follows.

First, in these competitor methods, the first dimension of the solution is set to the available bike station *start_bs* nearest to the starting point, and the last dimension is set to the available bike station *end_bs* nearest to the ending point. Second, in the greedy method, assume the dimensions of the solution are 3, 4, …, up to *num_bs*. In each assumed dimension, two solutions are obtained according to the following two greedy strategies. The first greedy strategy starts with the bike station in the first dimension and chooses the nearest unvisited bike station as the next dimension repeatedly until it reaches the assumed dimension. The second greedy strategy starts with the bike station in the first dimension and chooses the nearest unvisited bike station to *end_bs* that is accessible to the current bike station as the next dimension repeatedly until it reaches

the assumed dimension. In this way, a total of 2×(*num_bs*–2) solutions are built (as there are two greedy strategies), and the solution with the best fitness is selected as the solution obtained by the greedy method. In addition, in the enumeration method, all possible solutions will be generated, and the best solution among them is regarded as the final solution, which is actually the real global optimal solution of the problem.

Table 5.1 The experimental results between KLRP-MA and competitor methods on the medium-distance and small-scale test instances

Test instance	Greedy method	Enumeration method	KLRP-MA	Test instance	Greedy method	Enumeration method	KLRP-MA
M-S-1	52.6520	**51.8913**	**51.8913**	M-S-16	**54.8386**	**54.8386**	**54.8386**
M-S-2	71.6806	**68.5978**	**68.5978**	M-S-17	74.9605	**74.9605**	76.3093
M-S-3	94.0827	**92.4959**	**92.4959**	M-S-18	74.1424	**67.8647**	**67.8647**
M-S-4	90.6652	**80.2666**	81.2166	M-S-19	**92.1081**	**92.1081**	93.3415
M-S-5	**68.5535**	**68.5535**	68.9791	M-S-20	61.9282	**61.7869**	61.8152
M-S-6	**58.6853**	**58.6853**	**58.6853**	M-S-21	107.828	**106.946**	107.2100
M-S-7	80.3379	**79.0350**	79.0415	M-S-22	**67.5303**	**67.5303**	**67.5303**
M-S-8	84.9594	**80.8133**	80.9834	M-S-23	**85.2568**	**85.2568**	**85.2568**
M-S-9	90.1771	**84.3752**	**84.3752**	M-S-24	**98.2756**	**98.2756**	**98.2756**
M-S-10	61.9360	**61.1748**	**61.1748**	M-S-25	95.4848	**95.2801**	**95.2801**
M-S-11	57.0987	**50.8631**	51.3888	M-S-26	69.6845	**68.6136**	68.6671
M-S-12	79.1271	**79.0012**	**79.0012**	M-S-27	61.2547	**59.5600**	59.8142
M-S-13	80.1535	**78.1813**	**78.1813**	M-S-28	78.7486	**78.2241**	**78.2241**
M-S-14	**75.0700**	**75.0700**	**75.0700**	M-S-29	**90.7250**	**90.7250**	91.1207
M-S-15	**87.0149**	**87.0149**	**87.0149**	M-S-30	66.8113	**65.1081**	65.3607

Table 5.2 The experimental results between KLRP-MA and competitor methods on the medium-distance and medium-scale test instances

Test instance	Greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA
M-M-1	**59.9528**	60.2560	M-M-11	81.1852	**79.1799**	M-M-21	75.9139	**72.8263**
M-M-2	97.5751	**97.3354**	M-M-12	77.2515	**76.6066**	M-M-22	86.4848	**85.2704**
M-M-3	**63.1605**	63.1852	M-M-13	**113.828**	113.875	M-M-23	80.6237	**79.6352**
M-M-4	85.2785	**85.2267**	M-M-14	78.0762	**77.8335**	M-M-24	84.8765	**84.2151**
M-M-5	93.5649	**92.0777**	M-M-15	68.4442	**67.3662**	M-M-25	85.1789	**82.4892**
M-M-6	**75.7488**	75.8091	M-M-16	74.2283	**74.0354**	M-M-26	**101.692**	101.718
M-M-7	**63.7517**	64.2728	M-M-17	72.1250	**68.4100**	M-M-27	100.567	**99.5029**
M-M-8	96.0293	**90.5164**	M-M-18	85.8888	**85.4967**	M-M-28	**79.2305**	79.6745
M-M-9	78.2761	**77.6066**	M-M-19	93.1040	**89.4831**	M-M-29	71.8424	**71.8424**
M-M-10	102.809	**97.0502**	M-M-20	78.8271	**76.8394**	M-M-30	**65.6666**	66.0240

Table 5.3 The experimental results between KLRP-MA and competitor methods on the medium-distance and large-scale test instances

Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA
M-L-1	66.2695	**64.0927**	M-L-11	58.4579	**58.1563**	M-L-21	74.5157	**74.2784**
M-L-2	58.3499	**55.1941**	M-L-12	62.0149	**58.1667**	M-L-22	58.1792	**57.3695**
M-L-3	74.8173	**74.6563**	M-L-13	**66.2812**	66.4338	M-L-23	58.3318	**56.9933**
M-L-4	55.4424	**55.2051**	M-L-14	71.2747	**71.2670**	M-L-24	61.1710	**61.3044**
M-L-5	61.8389	**61.8007**	M-L-15	**59.8665**	60.2019	M-L-25	**54.0756**	54.5772
M-L-6	75.5564	**73.9086**	M-L-16	56.3731	**55.9977**	M-L-26	56.9090	**56.6509**
M-L-7	80.1910	**80.0624**	M-L-17	60.4991	**59.5487**	M-L-27	54.6087	**53.3714**
M-L-8	60.9766	**60.5787**	M-L-18	67.3189	**66.9636**	M-L-28	64.0283	**63.0352**
M-L-9	57.8430	**54.5254**	M-L-19	**59.5935**	59.7938	M-L-29	**81.7327**	81.7752
M-L-10	64.7589	**63.8456**	M-L-20	61.7252	**61.5662**	M-L-30	**61.5761**	62.1307

Table 5.1, Table 5.2, and Table 5.3 show the experimental results between KLRP-MA and the competitor methods on small-scale, medium-scale, and large-scale test instances when *distance*=12 (i.e., the medium distance). Table 5.1, Table 5.1, Table 5.2, and Table 5.3 show the experimental results between KLRP-MA and the competitor methods on small-scale, medium-scale, and large-scale test instances when *distance*=12 (i.e., the medium distance), respectively. 5.2, and Table 5.3 show the experimental results between KLRP-MA and the competitor methods on small-scale, medium-scale, and large-scale test instances when *distance*=12 (i.e., the medium distance), respectively. 5.4, Table 5.5 and Table 5.1, Table 5.2, and Table 5.3 show the experimental results between KLRP-MA and the competitor methods on small-scale, medium-scale, and large-scale test instances when *distance*=12 (i.e., the medium distance), respectively. show the experimental results between KLRP-MA and the competitor methods on small-scale, medium-scale, and large-scale test instances when *distance*=20, respectively. Note that on the medium-scale and large-scale test instances, the enumeration method cannot finish in the acceptable time (i.e., 3 minutes). Therefore, the enumeration method is only compared on small-scale test instances. In addition, on each test instance, the KLRP-MA result in the table is the average of 20 independent runs. The best result on each test instance is marked in **boldface**.

Table 5.4 The experimental results between KLRP-MA and competitor methods on the long-distance and small-scale test instances

Test instance	greedy method	enumeration method	KLRP-MA	Test instance	greedy method	enumeration method	KLRP-MA
L-S-1	146.368	**145.607**	**145.683**	L-S-16	**144.611**	**144.611**	**144.611**
L-S-2	166.176	**163.093**	**163.093**	L-S-17	**158.557**	**158.557**	159.350
L-S-3	185.805	**184.218**	**184.218**	L-S-18	166.416	**158.889**	159.265
L-S-4	180.979	**170.580**	173.285	L-S-19	**181.706**	**181.706**	182.216
L-S-5	**155.339**	**155.339**	155.943	L-S-20	152.143	**152.001**	**152.015**
L-S-6	150.814	**150.814**	152.040	L-S-21	185.747	**185.458**	**185.458**
L-S-7	**177.749**	**177.749**	177.786	L-S-22	**156.661**	**156.661**	**156.661**
L-S-8	164.614	**160.446**	160.808	L-S-23	**169.953**	**169.953**	**169.953**
L-S-9	169.857	**168.431**	**168.431**	L-S-24	**191.069**	**191.069**	**191.069**
L-S-10	150.688	**149.927**	**149.927**	L-S-25	178.791	**178.586**	178.855
L-S-11	151.377	**145.141**	**145.667**	L-S-26	165.013	**163.942**	163.996
L-S-12	175.125	**174.999**	**174.999**	L-S-27	156.426	**154.731**	154.985
L-S-13	164.258	**162.286**	**162.286**	L-S-28	158.705	**158.181**	158.218
L-S-14	**163.034**	**163.034**	**163.034**	L-S-29	**160.882**	**160.882**	161.409
L-S-15	**193.222**	**193.222**	**193.222**	L-S-30	158.307	**156.604**	156.730

Table 5.5 The experimental results between KLRP-MA and competitor methods on the long-distance and medium-scale test instances

Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA
L-M-1	182.172	**151.745**	L-M-11	164.650	**157.954**	L-M-21	167.028	**158.704**
L-M-2	**180.370**	180.753	L-M-12	193.188	**163.981**	L-M-22	181.053	**177.942**
L-M-3	**157.675**	158. 079	L-M-13	213.896	**200.739**	L-M-23	178.447	**163.330**
L-M-4	192.355	**174.624**	L-M-14	170.510	**167.331**	L-M-24	192.296	**171.036**
L-M-5	200.180	**183.618**	L-M-15	170.610	**158.056**	L-M-25	188.101	**178.068**
L-M-6	180.857	**149.732**	L-M-16	174.404	**166.159**	L-M-26	206.555	**195.827**
L-M-7	161.444	**158.335**	L-M-17	187.395	**156.761**	L-M-27	180.744	**168.672**
L-M-8	175.794	**170.853**	L-M-18	165.116	**163.053**	L-M-28	**161.676**	162.120
L-M-9	161.961	**160.531**	L-M-19	181.719	**174.936**	L-M-29	179.326	**165.033**
L-M-10	177.456	**175.119**	L-M-20	166.130	**165.622**	L-M-30	182.275	**153.519**

For the medium-distance test instances, when the number of available bike stations is 10 (i.e., small-scale), the greedy method, enumeration method, and KLRP-MA obtain the global optimal solution on 10, 30, and 17 test instances, respectively. Moreover, when the number of available bike stations is 20 (i.e., medium-scale), the greedy method and KLRP-MA obtain the best solution on 8 and 22 test instances, respectively.

In addition, when the number of available bike stations is 50 (i.e., large-scale), the greedy method and KLRP-MA obtain the best solution on 6 and 24 test instances, respectively. Although the enumeration method can find the global optimal solution in the acceptable time for small-scale test instances, the running time of the enumeration method is affected by the scale of the test instances and cannot work well on larger-scale problems. At the same time, with the increase in the problem scale, the performance of KLRP-MA is obviously better than that of the greedy method, which indicates the efficiency of KLRP-MA in solving URPP.

Table 5.6 The experimental results between KLRP-MA and competitor methods on the long-distance and large-scale test instances

Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA	Test instance	greedy method	KLRP-MA
L-L-1	205.486	**161.580**	L-L-11	156.764	**146.867**	L-L-21	169.864	**167.581**
L-L-2	155.254	**151.140**	L-L-12	174.803	**147.628**	L-L-22	156.597	**152.587**
L-L-3	160.072	**156.928**	L-L-13	183.723	**153.419**	L-L-23	154.935	**150.506**
L-L-4	157.532	**148.430**	L-L-14	172.212	**154.239**	L-L-24	168.420	**145.153**
L-L-5	164.043	**146.193**	L-L-15	173.966	**153.175**	L-L-25	162.001	**149.335**
L-L-6	168.737	**162.315**	L-L-16	152.307	**142.097**	L-L-26	160.858	**147.455**
L-L-7	211.420	**163.571**	L-L-17	181.566	**141.894**	L-L-27	193.759	**148.514**
L-L-8	231.163	**159.418**	L-L-18	159.789	**151.141**	L-L-28	174.118	**162.989**
L-L-9	154.948	**150.355**	L-L-19	165.210	**155.069**	L-L-29	178.577	**165.325**
L-L-10	168.075	**147.628**	L-L-20	183.143	**147.992**	L-L-30	201.066	**152.043**

For the long-distance test instances, when the number of available bike stations is 10 (i.e., small-scale), the greedy method, enumeration method, and KLRP-MA obtain the best results on 11, 30, and 16 test instances, respectively. Moreover, when the number of available bike stations is 20 (i.e., medium-scale), the greedy method and KLRP-MA obtain the best results on 8 and 22 test instances, respectively. In addition, when the number of available bike stations is 50 (i.e., large-scale), KLRP-MA performs best on all 30 test instances. Based on the above comparison with the greedy and enumeration methods, the effectiveness of the KLRP-MA has been verified.

Table 5.7 The running time of KLRP-MA and the enumeration method on the medium-distance test instances (Unit: Second)

Test instance	num_bs =10(*=S)		num_bs =20(*=M)		num_bs =50(*=L)	
	KLRP-MA	enumeration method	KLRP-MA	enumeration method	KLRP-MA	enumeration method
M-*-1	**0.0104**	1.3630	**0.0195**	NAN	**0.0369**	NAN
M-*-2	**0.0113**	1.4270	**0.0184**	NAN	**0.0385**	NAN
M-*-3	**0.0115**	1.4010	**0.0234**	NAN	**0.0241**	NAN
M-*-4	**0.0126**	1.3830	**0.0157**	NAN	**0.0221**	NAN
M-*-5	**0.0108**	1.3780	**0.0172**	NAN	**0.0236**	NAN
M-*-6	**0.0089**	1.3180	**0.0173**	NAN	**0.0227**	NAN
M-*-7	**0.0103**	1.4060	**0.0144**	NAN	**0.0243**	NAN
M-*-8	**0.0107**	1.4590	**0.0201**	NAN	**0.0237**	NAN
M-*-9	**0.0089**	1.3230	**0.0151**	NAN	**0.0336**	NAN
M-*-10	**0.0110**	1.4010	**0.0153**	NAN	**0.0261**	NAN
M-*-11	**0.0102**	1.3320	**0.0187**	NAN	**0.0257**	NAN
M-*-12	**0.0105**	1.3780	**0.0193**	NAN	**0.0226**	NAN
M-*-13	**0.0113**	1.3720	**0.0243**	NAN	**0.0230**	NAN
M-*-14	**0.0118**	1.4210	**0.0150**	NAN	**0.0256**	NAN
M-*-15	**0.0113**	1.3250	**0.0154**	NAN	**0.0330**	NAN
M-*-16	**0.0110**	1.3690	**0.0160**	NAN	**0.0256**	NAN
M-*-17	**0.0111**	1.3230	**0.0176**	NAN	**0.0250**	NAN
M-*-18	**0.0110**	1.3250	**0.0164**	NAN	**0.0281**	NAN
M-*-19	**0.0106**	1.3600	**0.0154**	NAN	**0.0223**	NAN
M-*-20	**0.0095**	1.3580	**0.0146**	NAN	**0.0260**	NAN
M-*-21	**0.0111**	1.3310	**0.0155**	NAN	**0.0264**	NAN
M-*-22	**0.0110**	1.3520	**0.0148**	NAN	**0.0239**	NAN
M-*-23	**0.0105**	1.4410	**0.0175**	NAN	**0.0298**	NAN
M-*-24	**0.0082**	1.3150	**0.0155**	NAN	**0.0266**	NAN
M-*-25	**0.0109**	1.4570	**0.0156**	NAN	**0.0237**	NAN
M-*-26	**0.0107**	1.4140	**0.0150**	NAN	**0.0266**	NAN
M-*-27	**0.0083**	1.3490	**0.0158**	NAN	**0.0315**	NAN
M-*-28	**0.0104**	1.3970	**0.0180**	NAN	**0.0245**	NAN
M-*-29	**0.0108**	1.3310	**0.0155**	NAN	**0.0250**	NAN
M-*-30	**0.0104**	1.3510	**0.0159**	NAN	**0.0352**	NAN

Besides the optimization results, Table 5.7 and Table 5.8 compare the running time of KLRP-MA and the enumeration method on medium-distance and long-distance test instances. Although the running time of the greedy method is short, the greedy method is not compared due to the poor quality of its solution. In addition, NAN in Table 5.7

and Table 5.8 indicates that the enumeration method cannot find the optimal solution within the acceptable time (i.e., 3 minutes). The enumeration method cannot find the optimal solution within even an hour on medium-scale test instances.

Table 5.8 The running time of KLRP-MA and the enumeration method on the long-distance test instances (Unit: Second)

Test instance	num_bs =10(*=S)		num_bs =20(*=M)		num_bs =50(*=L)	
	KLRP-MA	enumeration method	KLRP-MA	enumeration method	KLRP-MA	enumeration method
L-*-1	**0.0127**	1.3460	**0.0152**	NAN	**0.0332**	NAN
L-*-2	**0.0120**	1.3310	**0.0158**	NAN	**0.0336**	NAN
L-*-3	**0.0126**	1.3720	**0.0223**	NAN	**0.0245**	NAN
L-*-4	**0.0125**	1.3310	**0.0170**	NAN	**0.0227**	NAN
L-*-5	**0.0103**	1.3360	**0.0134**	NAN	**0.0236**	NAN
L-*-6	**0.0091**	1.3040	**0.0199**	NAN	**0.0236**	NAN
L-*-7	**0.0119**	1.3130	**0.0153**	NAN	**0.0239**	NAN
L-*-8	**0.0112**	1.3360	**0.0150**	NAN	**0.0250**	NAN
L-*-9	**0.0091**	1.3100	**0.0158**	NAN	**0.0337**	NAN
L-*-10	**0.0109**	1.3110	**0.0218**	NAN	**0.0264**	NAN
L-*-11	**0.0104**	1.3080	**0.0147**	NAN	**0.0273**	NAN
L-*-12	**0.0105**	1.3090	**0.0142**	NAN	**0.0229**	NAN
L-*-13	**0.0111**	1.3070	**0.0155**	NAN	**0.0240**	NAN
L-*-14	**0.0110**	1.3070	**0.0169**	NAN	**0.0266**	NAN
L-*-15	**0.0101**	1.3040	**0.0146**	NAN	**0.0313**	NAN
L-*-16	**0.0118**	1.3170	**0.0151**	NAN	**0.0262**	NAN
L-*-17	**0.0115**	1.3050	**0.0138**	NAN	**0.0251**	NAN
L-*-18	**0.0287**	1.3330	**0.0150**	NAN	**0.0295**	NAN
L-*-19	**0.0104**	1.3080	**0.0147**	NAN	**0.0225**	NAN
L-*-20	**0.0096**	1.3060	**0.0176**	NAN	**0.0273**	NAN
L-*-21	**0.0114**	1.3160	**0.0153**	NAN	**0.0283**	NAN
L-*-22	**0.0122**	1.3020	**0.0155**	NAN	**0.0244**	NAN
L-*-23	**0.0108**	1.3060	**0.0178**	NAN	**0.0314**	NAN
L-*-24	**0.0080**	1.3060	**0.0153**	NAN	**0.0242**	NAN
L-*-25	**0.0108**	1.3440	**0.0155**	NAN	**0.0240**	NAN
L-*-26	**0.0109**	1.3240	**0.0178**	NAN	**0.0266**	NAN
L-*-27	**0.0090**	1.3050	**0.0156**	NAN	**0.0325**	NAN
L-*-28	**0.0113**	1.3160	**0.0176**	NAN	**0.0253**	NAN
L-*-29	**0.0110**	1.3050	**0.0154**	NAN	**0.0278**	NAN
L-*-30	**0.0108**	1.3040	**0.0154**	NAN	**0.0371**	NAN

As shown in Table 5.7, the running time of KLRP-MA slightly increases with the increment of the number of available bike stations (i.e., the problem scale). On the medium-distance and large-scale test instances, the maximum running time is 0.0385 s, which is within the user's acceptable time. In addition, when the number of available bike stations is 10, the running time of KLRP-MA is much less than that of the enumeration method, i.e., less than 0.8% of the time cost of the enumeration method on most test instances. For the long-distance test instances, as shown in Table 5.8, the running time of KLRP-MA is significantly better than that of the enumeration method on all test instances with different scales. In addition, as seen from Table 5.7 and Table 5.8, the running time of KLRP-MA increases with the increment of the number of available bike stations on the test instances with different distances. Meanwhile, on the test instances with the same scale, the running time of the long-distance test instances is longer than that of the medium-distance test instances. Although the search space of these two experiments is the same, the distribution of the bike stations is relatively sparse on the long-distance test instances, which makes the solutions more likely to violate the constraints and need to be re-executed, thus taking more time. Based on the above, the efficiency of KLRP-MA has been verified.

5.5.3 Experimental Results under Dynamic Situations

Table 5.9 shows the results of DyKLRP-MA before and after dynamic change on small-scale, medium-scale, and large-scale test instances when *distance*=20 (i.e., the long distance). The distribution of available bike stations of the test instances is consistent with those in Section 5.4.3, and the results before the change are the best results obtained by KLRP-MA.

As shown in Table 5.9, when compared with the results before the dynamic change, DyKLRP-MA can find solutions with better fitness or at least not worse fitness on almost all test instances, and only produces worse results on two instances, i.e., L-S-8 and L-S-12. The reason for the worse results in these two instances may be that the small-scale instances only have ten bike stations to choose from, and impor stations are disabled during the dynamic change, while the rest of the existing and newly enabled

bike stations are not ideal for substituting the disabled bike stations to generate a good solution. However, in instances with more available bike stations (i.e., L-M-* and L-L-* instances), DyKLRP-MA can always find a better solution after the dynamic change. Based on the above, the problem-solving ability of DyKLRP-MA for dynamic URPP instances has been verified.

Table 5.9 The experimental results of DyKLRP-MA on the long-distance test instances

Test instance	Results before change	Results after change	Test instance	Results before change	Results after change	Test instance	Results before change	Results after change
L-S-1	**145.607**	**145.607**	L-M-1	151.743	**90.087**	L-L-1	161.154	**92.647**
L-S-2	163.093	**113.065**	L-M-2	**179.905**	**179.905**	L-L-2	**150.712**	**150.712**
L-S-3	**184.218**	**184.218**	L-M-3	157.427	**91.130**	L-L-3	156.850	**119.994**
L-S-4	170.580	**143.295**	L-M-4	174.442	**143.130**	L-L-4	147.997	**78.141**
L-S-5	**155.339**	**155.339**	L-M-5	182.403	**128.431**	L-L-5	146.176	**98.676**
L-S-6	150.814	**139.424**	L-M-6	149.695	**104.259**	L-L-6	162.164	**156.805**
L-S-7	177.749	**98.536**	L-M-7	**158.099**	**158.099**	L-L-7	163.546	**104.075**
L-S-8	**160.446**	167.643	L-M-8	**170.813**	**170.813**	L-L-8	159.201	**105.527**
L-S-9	168.431	**119.146**	L-M-9	**160.531**	**160.531**	L-L-9	150.264	**96.877**
L-S-10	149.927	**145.141**	L-M-10	**175.097**	**175.097**	L-L-10	146.940	**85.569**
L-S-11	**174.999**	**174.999**	L-M-11	157.326	**139.932**	L-L-11	146.860	**86.021**
L-S-12	**162.286**	162.707	L-M-12	**193.927**	**193.927**	L-L-12	147.524	**78.374**
L-S-13	**163.034**	**163.034**	L-M-13	200.739	**190.423**	L-L-13	**153.235**	**153.235**
L-S-14	**117.893**	**117.893**	L-M-14	**167.231**	**167.231**	L-L-14	152.630	**88.311**
L-S-15	**193.222**	**193.222**	L-M-15	157.647	**149.588**	L-L-15	152.152	**114.135**
L-S-16	144.611	**103.299**	L-M-16	166.108	**123.331**	L-L-16	**142.091**	**142.091**
L-S-17	**158.557**	**158.557**	L-M-17	156.761	**103.625**	L-L-17	141.584	**101.127**
L-S-18	**158.889**	**158.889**	L-M-18	162.977	**151.339**	L-L-18	151.066	**93.044**
L-S-19	181.706	**137.015**	L-M-19	**174.790**	**174.790**	L-L-19	**154.912**	**154.912**
L-S-20	152.001	**133.623**	L-M-20	**165.325**	**165.325**	L-L-20	147.969	**89.547**
L-S-21	**185.458**	**185.458**	L-M-21	158.665	**95.079**	L-L-21	167.541	**115.816**
L-S-22	**156.661**	**156.661**	L-M-22	177.805	**105.825**	L-L-22	152.543	**107.593**
L-S-23	169.953	**121.699**	L-M-23	162.068	**157.430**	L-L-23	150.396	**105.275**
L-S-24	**191.069**	**191.069**	L-M-24	171.022	**137.088**	L-L-24	144.422	**95.477**
L-S-25	**178.586**	**178.586**	L-M-25	**177.897**	**177.897**	L-L-25	148.512	**103.226**
L-S-26	**163.942**	**163.942**	L-M-26	**195.452**	**195.452**	L-L-26	147.125	**141.623**
L-S-27	**154.731**	**154.731**	L-M-27	**168.469**	**168.469**	L-L-27	148.382	**108.920**
L-S-28	**158.181**	**158.181**	L-M-28	161.676	**119.153**	L-L-28	162.799	**149.843**
L-S-29	160.882	**131.783**	L-M-29	164.910	**126.064**	L-L-29	165.263	**152.802**
L-S-30	156.604	**142.026**	L-M-30	153.270	**95.370**	L-L-30	150.842	**98.973**

5.5.4 Experiments on Real-world Bike-sharing Instances

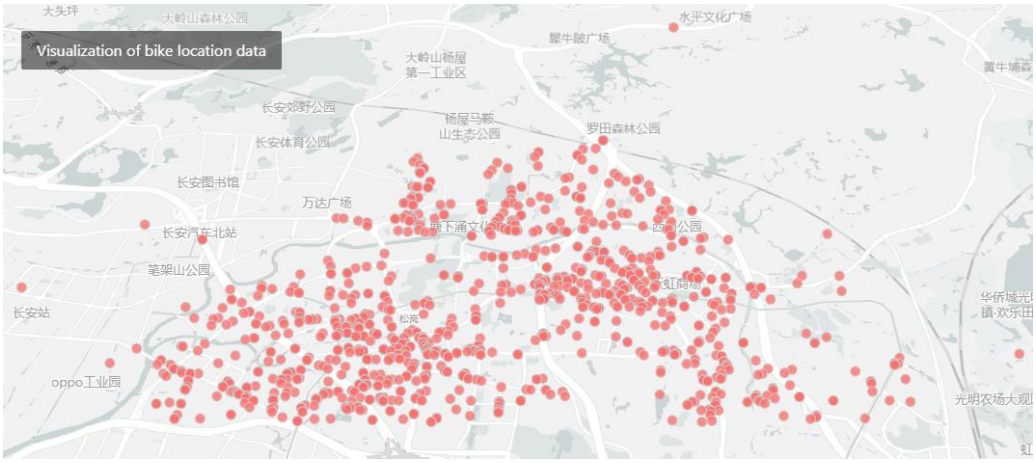


Figure 5.6 Visualization of bike location data in real-world test instances.

To further investigate the proposed DyKLRP-MA, this part compares the DyKLRP-MA with state-of-the-art approaches on some real-world test instances. The real-world instances are based on bike-sharing data from Shenzhen, China [248]. Specifically, Figure 5.6 hows all the bike location data used in this chapter. The test instances are construed as follows. To begin with, all the bike locations are clustered by K-means into 50 groups, where the center of each group represents a bike station. Then, 10 user records (including the source and destination of the same user) are used to generate 10 test instances for route planning based on the 50 bike stations. For simplicity, the 10 instances are denoted as R-1, R-2, …, and R-10, respectively. In addition, as re-initialization and using archive are two state-of-the-art and popular dynamic optimization approaches [244]-[245], the proposed DyKLRP-MA is compared with the KLRP-MA variants using re-initialization and archive. The compared two variants are denoted as MA-R and MA-A for short, respectively. The MA-R re-initializes the population when the problem is dynamically changed, while the MA-A stores all the best solutions found in every previous dynamic environment in an archive and re-evaluates the solutions in the archive when the problem is changed again. For better comparisons, the best-before-change error metric (denoted as E_B) is used herein, which is a widely-used indicator designed for dynamic optimization [176]. The calculation of E_B is as follows:

$$E_B = \frac{1}{C}\sum_{i=1}^{C} E_{b,i} \tag{5.7}$$

where $E_{b,i}$ is the fitness of the best solution found before the i^{th} dynamic change, and C is the total number of changes during the optimization. Each algorithm runs 20 runs to obtain the results for comparison. During each run, each algorithm will evolve for 400 generations. Every 20 generations, all the bike stations will have a probability of 0.1 to become unavailable. That is, there will be 20 dynamic changes in each run. The experimental results on the real-world test instances are given in Table 5.10. As can be seen, the DyKLRP-MA obtains the best results on 8 of the 10 problems, while the MA-R and MA-A can only obtain the best results on 0 and 1 problems, respectively. This suggests that the proposed DyKLRP-MA is very suitable for solving the URPP in dynamic situations.

Table 5.10 The experimental results of DyKLRP-MA, ma-r, and ma-a on the real-world test instances

Test instance	DyKLRP-MA	MA-R	MA-A
R-1	**86.0332**	86.2664	86.5661
R-2	**137.678**	140.664	138.027
R-3	**52.2539**	54.9427	56.1207
R-4	**157.785**	157.851	157.929
R-5	**89.1900**	91.6647	91.1149
R-6	77.9556	80.3473	**75.7700**
R-7	**184.270**	189.564	185.992
R-8	**23.4204**	27.9302	26.5034
R-9	**137.310**	137.536	138.627
R-10	70.5298	70.2346	**70.2020**

5.5.5 Component Analysis

To further study the influence of the KLC, RPM, and TPLS, three variants of the KLRP-MA are designed, namely the KLRP-MA without KLC, the KLRP-MA without RPM, and the KLRP-MA without TPLS. For simplicity, these three algorithms are denoted as KLRP-MA-w/o-KLC, KLRP-MA-w/o-RPM, and KLRP-MA-w/o-TPLS, respectively. The experiment is conducted on small-scale, medium-scale, and large-scale test instances with *distance*=20.

Table 5.11 The experimental results among KLRP-MA variants on the long-distance and small-scale test instances.

Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS	Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS
L-S-1	145.683	145.645	146.683	**145.607**	L-S-16	**144.611**	145.141	145.290	**144.611**
L-S-2	**163.093**	163.401	165.615	163.555	L-S-17	**159.350**	159.906	166.281	159.876
L-S-3	**184.218**	184.358	185.295	184.338	L-S-18	159.265	159.830	160.040	**159.202**
L-S-4	173.285	172.018	178.571	**171.734**	L-S-19	**182.216**	182.467	185.685	182.556
L-S-5	155.943	**155.339**	160.137	155.424	L-S-20	**152.015**	152.022	155.461	152.223
L-S-6	**152.040**	153.499	162.624	152.272	L-S-21	**185.458**	185.459	186.066	185.459
L-S-7	**177.749**	177.779	178.732	177.785	L-S-22	**156.661**	157.245	163.980	157.036
L-S-8	160.808	160.660	162.227	**160.472**	L-S-23	**169.953**	170.255	171.846	170.075
L-S-9	**168.431**	168.574	170.076	**168.431**	L-S-24	**191.069**	191.069	193.886	191.148
L-S-10	**149.927**	150.041	151.184	150.117	L-S-25	178.855	179.020	180.372	**178.731**
L-S-11	**145.667**	147.605	151.534	145.930	L-S-26	**163.996**	167.144	166.060	164.305
L-S-12	**174.999**	175.131	175.595	175.045	L-S-27	**154.985**	156.147	159.127	**154.985**
L-S-13	**162.286**	162.483	164.341	**162.286**	L-S-28	**158.218**	158.360	160.004	158.270
L-S-14	**163.034**	163.251	165.543	163.359	L-S-29	**161.409**	161.445	164.072	161.445
L-S-15	**193.222**	**193.222**	197.569	**193.222**	L-S-30	**156.730**	156.985	158.574	**156.730**

Table 5.12 The experimental results among KLRP-MA variants on the long-distance and medium-scale test instances.

Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS	Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS
L-M-1	**151.745**	151.755	160.652	151.759	L-M-16	**166.159**	166.359	181.164	166.284
L-M-2	180.753	**180.596**	204.168	181.265	L-M-17	**156.761**	158.077	172.800	157.729
L-M-3	158. 079	**157.801**	178.955	157.944	L-M-18	163.053	**163.032**	175.342	163.070
L-M-4	**174.624**	174.723	191.755	174.806	L-M-19	174.936	174.922	181.165	**174.893**
L-M-5	**183.618**	183.627	201.651	182.943	L-M-20	**165.622**	165.914	188.786	166.497
L-M-6	**149.732**	149.752	162.619	149.734	L-M-21	158.704	158.704	168.389	**158.697**
L-M-7	**158.335**	158.570	176.063	158.634	L-M-22	**177.942**	178.281	189.962	178.574
L-M-8	**170.853**	171.059	183.684	171.175	L-M-23	163.330	163.006	181.266	**162.678**
L-M-9	**160.531**	161.502	179.280	160.546	L-M-24	**171.036**	171.290	183.729	171.357
L-M-10	**175.119**	175.223	189.831	175.388	L-M-25	**178.068**	178.366	191.277	178.256
L-M-11	157.954	**157.883**	173.137	158.287	L-M-26	195.827	195.536	205.150	**195.500**
L-M-12	**163.981**	164.044	170.893	163.991	L-M-27	**168.672**	168.961	183.239	168.849
L-M-13	**200.739**	201.266	216.789	201.355	L-M-28	162.120	**161.948**	182.096	162.564
L-M-14	**167.331**	167.706	183.756	167.437	L-M-29	165.033	**165.008**	179.273	165.108
L-M-15	**158.056**	158.151	172.444	158.498	L-M-30	153.519	153.907	167.854	**153.511**

Table 5.13 The experimental results among KLRP-MA variants on the long-distance and large-scale test instances.

Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS	Test instance	KLRP-MA	KLRP-MA-w/o-KLC	KLRP-MA-w/o-RPM	KLRP-MA-w/o-TPLS
L-L-1	**161.580**	161.733	221.976	161.614	L-L-16	**142.097**	142.195	195.381	142.192
L-L-2	151.140	**150.857**	212.062	150.944	L-L-17	141.894	**141.728**	191.577	141.797
L-L-3	**156.928**	156.978	220.983	157.237	L-L-18	**151.141**	151.237	208.436	151.272
L-L-4	148.430	148.938	203.483	**149.153**	L-L-19	**155.069**	155.116	219.126	155.791
L-L-5	**146.193**	146.222	197.404	146.194	L-L-20	**147.992**	148.106	203.097	148.115
L-L-6	162.315	162.343	210.326	**162.216**	L-L-21	**167.581**	167.618	226.688	167.601
L-L-7	**163.571**	163.602	217.838	163.677	L-L-22	**152.587**	152.969	206.170	153.123
L-L-8	**159.418**	159.530	214.666	159.431	L-L-23	**150.506**	150.587	195.281	150.721
L-L-9	**150.355**	150.663	212.608	150.527	L-L-24	145.153	**144.884**	195.139	145.100
L-L-10	147.628	**147.084**	202.815	147.255	L-L-25	149.335	148.951	199.760	**148.701**
L-L-11	**146.867**	146.919	199.303	146.920	L-L-26	**147.455**	147.467	200.671	147.574
L-L-12	147.628	**147.623**	198.857	147.671	L-L-27	**148.514**	148.805	206.939	148.722
L-L-13	**153.419**	153.569	213.253	153.523	L-L-28	162.989	162.980	209.753	**162.957**
L-L-14	**154.239**	154.378	209.416	154.273	L-L-29	**165.325**	165.343	217.156	165.405
L-L-15	153.175	**153.124**	203.778	153.425	L-L-30	152.043	152.389	208.967	**151.596**

Table 5.14 The optimization results and running time of different KLRP-MA variants with different maximum dimensions on the long-distance and medium-scale test instances.

Optimization results					Running time				
Test instance	KLRP-MA	KLRP-MA-1	KLRP-MA-2	KLRP-MA-3	Test instance	KLRP-MA	KLRP-MA-1	KLRP-MA-2	KLRP-MA-3
L-M-1	**151.745**	151.752	151.836	151.839	L-M-1	0.01455	0.02305	0.01330	**0.01095**
L-M-2	**180.753**	181.601	180.847	180.807	L-M-2	0.01510	0.02415	0.01330	**0.01135**
L-M-3	**158. 079**	158.411	158.759	158.784	L-M-3	0.02090	0.27080	0.01525	**0.01100**
L-M-4	**174.624**	174.929	**174.624**	174.644	L-M-4	0.01705	0.03550	0.01345	**0.01105**
L-M-5	**183.618**	184.023	184.158	184.213	L-M-5	0.01690	0.05035	0.01555	**0.01175**
L-M-6	**149.732**	150.121	149.740	149.738	L-M-6	0.01750	0.10340	0.01385	**0.01200**
L-M-7	**158.335**	158.365	158.680	158.570	L-M-7	0.01310	0.02105	0.01215	**0.01025**
L-M-8	**170.853**	171.004	170.934	170.907	L-M-8	0.01850	0.12720	0.01535	**0.01100**
L-M-9	**160.531**	161.127	160.806	161.001	L-M-9	0.01460	0.02245	0.01230	**0.01060**
L-M-10	**175.119**	175.223	**175.119**	175.206	L-M-10	0.01455	0.02595	0.01270	**0.01060**
L-M-11	**157.954**	158.189	158.640	158.170	L-M-11	0.01535	0.03680	0.01365	**0.01125**
L-M-12	**163.981**	164.256	164.256	**163.981**	L-M-12	0.02115	0.18830	0.01615	**0.01160**
L-M-13	**200.739**	**200.739**	**200.739**	200.752	L-M-13	0.02840	0.19975	0.02120	**0.01280**
L-M-14	**167.331**	167.431	167.431	**167.331**	L-M-14	0.01430	0.02505	0.01285	**0.01075**
L-M-15	**158.056**	158.610	158.140	158.152	L-M-15	0.01420	0.02440	0.01260	**0.01080**

Table 5.15 The optimization results and running time of different KLRP-MA variants with different generation probabilities on the long-distance and medium-scale test instances.

Test instance	Pg=0.8	Pg=0	Pg=0.2	Pg=0.4	Pg=0.6	Pg=1.0
L-M-1	**151.745**	151.745	151.744	151.751	151.750	**151.745**
L-M-2	**180.753**	181.905	181.130	181.958	182.928	180.782
L-M-3	**158. 079**	158.427	158.427	158.759	158.128	158.452
L-M-4	**174.624**	174.929	175.442	174.688	174.868	**174.624**
L-M-5	**183.618**	184.673	184.078	184.483	184.745	184.158
L-M-6	**149.732**	150.703	149.799	149.703	149.734	149.751
L-M-7	**158.335**	158.388	159.190	158.455	158.833	158.425
L-M-8	**170.853**	**170.853**	170.894	170.910	170.937	170.924
L-M-9	**160.531**	160.532	160.534	161.133	161.138	160.945
L-M-10	175.119	175.197	175.203	175.211	175.106	**175.100**
L-M-11	157.954	158.545	158.864	157.983	157.983	**157.814**
L-M-12	163.981	164.927	**163.927**	163.981	163.981	163.981
L-M-13	**200.739**	**200.739**	**200.739**	**200.739**	**200.739**	**200.739**
L-M-14	**167.331**	167.431	167.431	**167.331**	**167.331**	167.431
L-M-15	**158.056**	158.658	158.766	158.145	158.461	158.183

Table 5.11, Table 5.12, and Table 5.13 show the experimental results of KLRP-MA and its variants on small-scale, medium-scale, and large-scale test instances with *distance*=20, respectively, where the best result is marked in **boldface**. Specifically, on the small-scale, medium-scale, and large-scale test instances, the original KLRP-MA performs the best on 24, 19, and 19 test instances, respectively, which is significantly better than the compared variants without KLC, RPM, or TPLS. In addition, Figure 5.7 plots the convergence curve of different KLRP-MA variants on a medium-scale test instance. As seen in Figure 5.7, the proposed KLRP-MA has better convergence than the remaining three variants. Therefore, the experimental results show that the KLC, RPM, and TPLS can improve the performance of the KLRP-MA, and removing any of them will decrease the algorithm's performance.

5.5.6 Influence of Parameter

This part studies the parameters D_{max} and Pg. D_{max} controls the maximum dimension of the candidate solution. As the original D_{max} is set according to Eq.(5.4) and Eq.(5.5), which will belong to [*num_bs*/2, *num_bs*/3] when *num_bs* is larger than

6, the KLRP-MA is compared with its variants that use D_{max}=*num_bs*, D_{max}= *num_bs*/2, and D_{max}= *num_bs*/3. For simplicity, the three variants are denoted as KLRP-MA-1, KLRP-MA-2, and KLRP-MA-3, respectively. The experimental results are given in Table 5.14. As can be seen, the KLRP-MA obtains the best optimization results on all the test instances. For the running time, KLRP-MA-3 has less running time, and the time costs of KLRP-MA are between those of KLRP-MA-1 and KLRP-MA-2. This shows that the larger D_{max} is, the lower the running time will be. As the setting of D_{max} in the original KLRP-MA can obtain the best optimization results, the slight additional time costs of the KLRP-MA over the variants KLRP-MA-2 and KLRP-MA-3 are acceptable. Based on the above, the original setting of D_{max} in KLRP-MA (according to Eq.(5.4) and Eq.(5.5)) is recommended in this chapter.

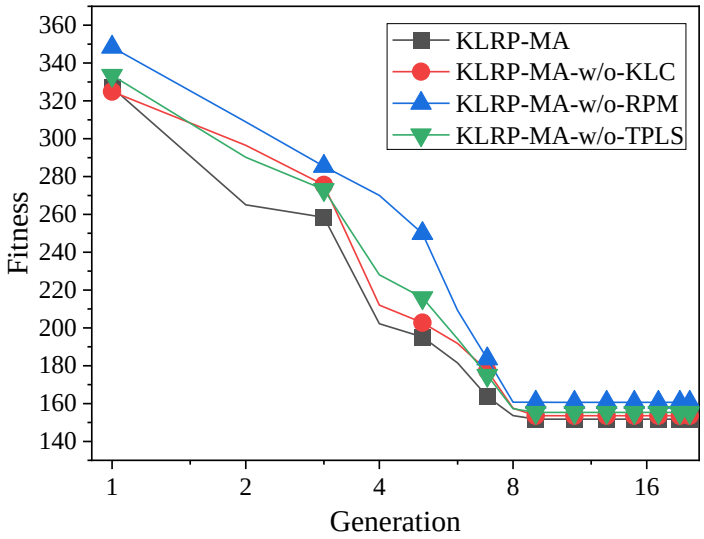


Figure 5.7 The convergence curve of different KLRP-MA variants on the L-M-1 test instance.

Moreover, the generation probability *Pg* is set as *Pg*=0.8 in the original KLRP-MA. Therefore, the original KLRP-MA is compared with its variants with *Pg*=0, *Pg*=0.2, *Pg*=0.4, *Pg*=0.6, and *Pg*=1.0. For simplicity, the original KLRP-MA is directly denoted as *Pg*=0.8, and the five variants are denoted as *Pg*=0, *Pg*=0.2, *Pg*=0.4, *Pg*=0.6, and *Pg*=1.0, respectively. The experimental results are given in

Table 5.15. As can be seen in

Table 5.15, variants with different *Pg* obtain similar results, which indicates that the KLRP-MA is not that sensitive to the setting of *Pg*. As *Pg*=0.8 obtains the best

results on most problems, Pg=0.8 is recommended by this chapter.

5.6 Conclusion

In this chapter, a new URPP model is proposed for route planning for the user in not only static situations but also dynamic situations, and the KLRP-MA is proposed to solve the URPP. As the traditional crossover operator and mutation operator are not suitable for solving the URPP with uncertain dimensions, this chapter proposes and integrates the KLC, RPM, and TPLS into KLRP-MA to generate better individuals. In addition, the KLRP-MA is extended as a dynamic optimization algorithm, and the novel DyKLRP-MA is proposed to handle the dynamic changes of the bike stations in dynamic URPP. Experimental results show that when compared with the greedy method and enumeration method, KLRP-MA can quickly search for the best solution. Specifically, KLRP-MA has more advantages in solving ability than the greedy method and more advantages in running time than the enumeration method. In addition, when the bike stations change dynamically, the DyKLRP-MA can produce promising solutions efficiently.

In future work, the tradeoff between riding cost and riding time is an important direction to extend the URPP model. For example, modelling URPP as a constraint optimization problem where the objective is traveling time and the user-specified cost is set as a constraint. In addition, the cost and time can be also modeled as a multi-objective optimization problem and a new multi-objective algorithm based on the proposed algorithm can be studied. In addition, expanding the scale size of the problem and conducting experiments with more bike stations will be considered. In these scenarios, the techniques for data-driven optimization [249]-[197], multi-objective optimization [195]-[252], and large-scale optimization can be applied [253]-[256].

This work was published on the journal of Memetic Computing.

CHAPTER 6

EVOLUTIONARY MULTITASKING BI-DIRECTIONAL PARTICLE SWARM OPTIMIZATION FOR HIGH-DIMENSIONAL FEATURE SELECTION

6.1 Introduction

In the era of big data, with the advancement of information retrieval technologies, the number of features in data has increased dramatically. Hence, in the industrial field, there is a popular saying that “feature engineering determines the upper limit of generalization ability, while models and algorithms merely approximate this limit” [225]. Feature selection is a common preprocessing method in data analysis [225]. It aims to select a subset of features most relevant to the problem and eliminate redundant or noisy features from the entire feature set. By performing feature selection, accuracy and efficiency in data processing can be enhanced by removing irrelevant and redundant features. Furthermore, in practical applications, feature selection can reduce dimensionality to improve model efficiency, mitigate the risk of overfitting, address noise issues, and enhance model interpretability [225]. Therefore, feature selection plays an increasingly important role in many fields today [257]. Even in deep learning, where features are learned automatically from raw data, effective feature selection remains crucial. Deep learning models can automatically learn higher-level abstract feature representations from raw data, expanding the scope of feature selection beyond the data preprocessing stage [257].

In the context of feature selection, let's use the process of admitting graduate students to a university as an example. Suppose a university is preparing to admit new graduate students. During the admission process, the university needs to select the most outstanding and promising students from a large pool of applicants. This process is akin to the task of feature selection, where various background information of students (such as grades, research experience, recommendation letters, etc.) corresponds to the

features in a dataset. When admitting graduate students, the university may encounter challenges and employ strategies similar to feature selection: selecting important features, eliminating redundant features, balancing feature weights, and handling high-dimensional data. Just like in feature selection, when the dataset's feature dimension is very high, algorithms may face difficulties in training, and high computational complexity, making it challenging to effectively handle and utilize all the feature information.

Most existing feature selection algorithms fall into three categories: filter methods, wrapper methods, and embedded methods [258]. Filter methods evaluate and rank features using statistical tests or correlation analysis before training the model, selecting the most relevant features for the problem. While filter methods have the advantage of lower model iteration overhead, they require features to be mutually independent, overlooking relationships between features. Embedded methods integrate the feature selection process into other machine learning algorithms without separate implementation. However, this approach requires careful model design and entails large computational costs due to the extensive input data. Wrapper methods determine whether a candidate feature subset is optimal by evaluating it within the corresponding problem. Compared to filter methods, wrapper methods may perform better as they consider the impact of feature combinations through classifier evaluation. Additionally, wrapper methods can be used independently without the need for specialized algorithm design as with embedded methods. However, wrapper methods still face challenges such as overfitting and high computational costs [259]. In practical applications, wrapper methods are commonly used for relatively small feature sets. For large-scale datasets, the computational overhead poses a research gap that has attracted the interest of scholars in recent years, leading to attempts to use different algorithms to address this issue.

As one type of wrapper method, swarm intelligence-based algorithms have demonstrated strong capabilities in feature selection problems in recent years [260]. Particle Swarm Optimization (PSO) is one of the most well-known swarm intelligence

algorithms, known for its ease of implementation and fast convergence [25]–[261]. However, PSO is often plagued by the "curse of dimensionality," exhibiting weak search capabilities when dealing with high-dimensional data [262]. Many PSO-based algorithms have been proposed to enhance its performance in high-dimensional feature selection problems, which can be roughly divided into two categories. The first category of PSO-based algorithms focuses on designing efficient particle evolution strategies to help them fully explore the solution space and escape local optima [263]–[265]. The second category of PSO-based algorithms calculates the importance of features using similarity or correlation measures and then focuses on those features with high importance to improve search efficiency [266]–[268]. Overall, the second category of PSO-based algorithms performs better because they can determine which features are worth further exploration to narrow down the search space. However, these algorithms often rely on correlation analysis, which is time-consuming and complex to implement on high-dimensional data. Due to the large number of features in high-dimensional data, conducting correlation analysis requires significant computational resources and time, and may lead to performance degradation or failure to converge.

These two categories can also be exemplified using the university admissions process. In the first category of PSO algorithms, the admissions committee adopts a comprehensive search approach, considering all features of every applicant, including academic records, personal statements, recommendation letters, etc., as equally important. While this method ensures that no potentially promising student is overlooked, it may result in a significant amount of time and effort being spent considering all features, rather than focusing on the most relevant and valuable ones. In the second category of PSO algorithms, the admissions committee may employ a more targeted approach to more effectively select the most promising group of students. The committee might first use correlation analysis methods, such as correlation measures based on academic records, recommendation letters, community service, etc., to determine which features are most highly correlated with successful admissions. Although this method can help the committee identify promising students more quickly,

inaccurate or incomplete correlation measures may lead to some potentially promising students being overlooked. Additionally, computing correlation measures may require a significant amount of time and resources. Overall, the first category of PSO algorithms is similar to a comprehensive search approach considering all features, while the second category of PSO algorithms resembles a targeted approach utilizing correlation information to select specific features.

Recently, Yang et al. proposed a novel framework called bidirectional feature fixing (BDFF) for high-dimensional feature selection [269]. In BDFF, each particle has two different search directions. One direction guides the particle to find a large feature subset with more selected features, while the other direction guides the particle to find a small feature subset with fewer selected features. Based on their search directions, particles can dynamically fix the selection status of features, which also helps narrow down the search space without overly relying on correlation analysis. Therefore, BDFF holds great potential for addressing high-dimensional feature selection problems. However, BDFF still faces some challenges. Firstly, particles may be misled by the search directions, resulting in missing the optimal solution. Since in BDFF, search directions are only inferred and adjusted based on the performance of the current population, some particles' search directions may be incorrect. Secondly, BDFF sometimes struggles to obtain a feature subset with few selected features because the initial number of features always remains around half of the total number of features, which is quite large in high-dimensional data. Therefore, addressing this issue becomes a research challenge.

In recent years, KT has garnered considerable attention in the field of EC due to its ability to accelerate the optimization process, enhance search capabilities, adapt to dynamic environments, and improve algorithm robustness. Taking cues from the concept of KT, the presence of crucial prior knowledge in the field of feature selection is recognized. Firstly, the objective of feature selection is to obtain a feature subset with as few selected features as possible. Secondly, reducing the number of features considered during the search for the optimal feature subset helps mitigate interference

between features and speeds up the feature selection process. However, existing methods for feature selection still inadequately leverage this prior knowledge. This chapter aims to integrate this prior knowledge into the BDFF framework while preserving its original capability for global knowledge discovery by designing different optimization tasks for feature selection. MTOP technology has emerged as a promising approach in high-dimensional feature selection, facilitating KT between different optimization tasks. In multi-task optimization, there may exist certain correlations or similarities between tasks. KT methods facilitate the transfer of valuable expertise and insights among various tasks, ensuring the efficiency and adaptability of the system. Therefore, this chapter introduces, for the first time, the combination of the BDFF framework with MTOP technology, proposing an algorithm called multi-task bidirectional particle swarm optimization (MBDPSO) for high-dimensional feature selection. In comparison with other existing EMTO-based feature selection algorithms, MBDPSO emphasizes the integration of MTOP and BDFF, particularly in leveraging the prior knowledge introduced by MTOP while retaining the global search capability of BDFF. The main contributions of this chapter can be summarized as follows.

Firstly, designing two related tasks, incorporating prior knowledge of feature selection into MBDPSO. Both tasks are aimed at feature selection. However, one task focuses solely on promising features, while the other considers all features to retain global search capability. Therefore, MBDPSO can quickly identify small feature subsets while still retaining the ability to search for feature subsets with different selected feature counts.

Secondly, proposing a novel KT approach to assist particles in searching in a multi-task environment. The KT process combines the advantages of feature fixation and a widely used PSO variant called Bare-Bones Particle Swarm Optimization (BBPSO). Additionally, using a linearly increasing function dynamically adjusts the probability of KT, helping particles strike a balance between mining knowledge within their task and learning from other tasks.

The remaining sections of this chapter are organized as follows. Section 6.2 briefly

introduces related work, while section 6.3 describes the detailed implementation of the proposed MBDPSO. In section 6.4, experimental results and analysis of MBDPSO on public feature selection datasets and real research cases are presented. Finally, section five concludes the chapter.

6.2 Related Work

6.2.1 Feature Selection

Feature selection is the process of selecting a subset of features from the entire feature set to optimize an evaluation function relevant to a specific problem. For instance, in classification problems, the evaluation function could be classification accuracy. Suppose there are D features in the entire feature set, then the feature selection problem for classification can be represented as:

$$\begin{aligned} &\max \ \mathrm{f(x)} \\ &\text{s. t. } x = (x_1, x_2, ..., x_D) \\ &\quad x_d \in \{0,1\},\ d = 1, 2, ..., D \end{aligned} \tag{6.1}$$

Where $f(\cdot)$ represents the classification accuracy, x is the solution to the feature selection problem, and x_d takes a value of 1 to select the dth feature or 0 to not select it. Choosing a sufficiently small feature subset with high classification accuracy is extremely challenging when D is large.

6.2.2 Bidirectional Feature Fixing Framework

The BDFF framework [269] was proposed for PSO to effectively address high-dimensional feature selection problems. It helps particles reduce the search space as they update their positions and can be combined with most PSO variants into a unified framework. In BDFF, each particle is defined with two search directions. One direction, diru, guides the particle to find a large feature subset, while the other direction, dirl, guides the particle to find a small feature subset. Additionally, BDFF divides all features into different feature neighborhoods. Then, if the features meet the feature fixing conditions based on the particle's current search direction, features within the same feature neighborhood will be fixed, maintaining their selection status. Corresponding to the two search directions, there are also two feature fixing conditions. If the direction

is diru and all features in the same neighborhood in the particle's historical best position are selected, feature fixing will be executed. Similarly, if the direction is dirl and no features in the same neighborhood are selected, feature fixing will also be executed. In BDFF, the feature fixing technique reduces the features considered in optimization, narrowing down the search space for each particle and thus improving search efficiency.

Although BDFF shows promising potential in high-dimensional feature selection, it still faces some challenges. The first challenge is the difficulty in adjusting search directions, which may lead to being trapped in the wrong direction. Initially, half of the particles use diru as the search direction, and the other half use dirl, as shown below:

Where $dir(p_i)$ returns the search direction of the ith particle p_i, and P is the total number of particles in the swarm. Only one direction is correct because the optimal solution can only be in one direction. Then, BDFF uses an adaptive direction change strategy to change the direction of particles to fully utilize particles in the wrong direction. However, ensuring the correct direction adjustment is extremely difficult. In some accidental cases, particles may all end up in the wrong direction, resulting in poor optimization results. The second challenge is that BDFF requires a significant amount of resources to search for small feature subsets. According to BDFF's initialization strategy, the number of features selected by each particle is approximately half of the total number of features. If the optimal solution contains a small feature subset, BDFF can only gradually reduce the size of the selected feature subset starting from half of the total features to find the optimal solution, which wastes a considerable amount of computational resources, especially when dealing with high-dimensional data.

6.2.3 Evolutionary Multi-Task Optimization

EMTO has seen significant advancements in recent years, with Multi-Factorial Evolutionary Algorithm (MFEA) being one of the most prominent approaches [30]. It introduces four fundamental concepts—Factorial Cost, Factorial Rank, Scalar Fitness, and Skill Factor—to assess individuals' performance in multi-task environments. MFEA incorporates techniques like assortative mating and vertical cultural transmission to facilitate information exchange among different tasks. By utilizing a

unified search space, MFEA enables the simultaneous optimization of multiple tasks within a single population, offering an efficient framework for EMTO. Various variants based on MFEA have been proposed, such as Multi-Factor Differential Evolution and Multi-Factor Particle Swarm Optimization by Feng et al. [31]. Additionally, efforts have been made to apply MFEA to expensive optimization problems, aiming to extract useful knowledge from cheaper problems and reduce computational costs, as demonstrated by Ding et al. [147]. Further enhancements include KT and resource allocation to improve the efficiency of MFEA [273][274][146][275].

6.3 Framework of MBDPSO

In this section, the detailed implementation of the proposed MBDPSO algorithm is presented. Firstly, two related tasks designed for high-dimensional feature selection problems and the overall framework of MBDPSO are provided. Then, the initialization of the three main components of particles in the population is discussed. To coordinate particles in the two tasks and transfer knowledge between different tasks, the multi-task evolutionary paradigm of MBDPSO is described. Finally, the complete MBDPSO is presented.

6.3.1 Two tasks of feature selection

The design of MBDPSO aims to leverage prior knowledge in the field of feature selection by combining BDFF with EMTO to help particles efficiently find optimal solutions [33]. In feature selection problems, the preference is to obtain feature subsets containing as few features as possible. Additionally, priority is given to features that are more likely to appear in the optimal solution rather than considering all features, which can provide useful information to expedite the optimization process. Based on the aforementioned prior knowledge, two related tasks specifically for feature selection are proposed. The objectives of these two tasks are the same, i.e., to select an optimal feature subset from the same dataset to maximize the function described in Equation (6.1) [33]. However, the feature scope to be considered differs between these two tasks.

The first task only considers promising features that constitute a small portion of the total features. Correlation analysis based on similarity or information entropy is an

effective method for assessing the importance of features for classification problems [276] [277]. Generally, features highly correlated with the classification labels have a higher probability of appearing in the optimal feature subset because they have a greater impact on the classification results. Therefore, these features can be prioritized and considered as promising features. Here, Symmetrical Uncertainty (SU) [278] was adopted to identify promising features, where the SU value between feature F and class label C can be calculated as follows:

$$\mathrm{SU}(F,C)=2\frac{\mathrm{H}(F)-\mathrm{H}(F\mid C)}{\mathrm{H}(F)+\mathrm{H}(C)} \tag{6.2}$$

Here, $H(F|C)$ represents the conditional entropy of F given C, while $H(F)$ and $H(C)$ are the entropies of F and C, respectively. After ranking the features based on their SU values with the class labels in descending order, the top K promising features are considered. Then, the first task selects features only from these promising ones to form a candidate feature subset. It's worth noting that the computation of SU results, used for locating promising features, is also required in BDFF as part of the preparation work for partitioning feature neighborhoods. Therefore, identifying promising features does not require additional computational resources.

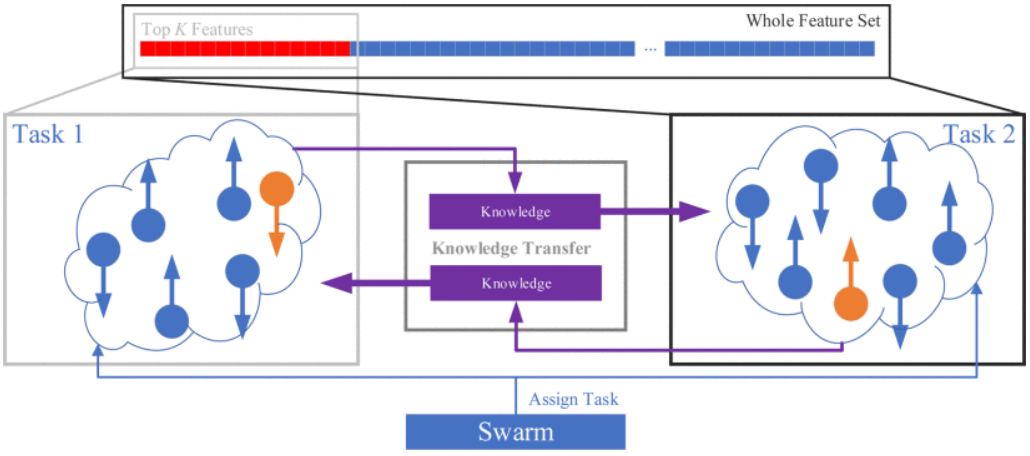


Figure 6.1 The framework of two tasks in feature selection

The second task considers the entire feature set range, which is necessary because the optimal feature subset is often not entirely composed of promising features used in the first task. The correlation analysis in the first task only considers the relationship between individual features and labels, making the collection of promising features

somewhat biased. Therefore, the second task retains the ability to comprehensively consider all features to help PSO reveal the implicit associations among features in the entire feature set.

Based on these two related tasks, the overall framework of MBDPSO is illustrated in Figure 6.1. The first task focuses on promising features with high SU values to incorporate prior knowledge of feature selection, while the second task considers the entire feature set to gain a comprehensive perspective on feature selection. Then, particles in the population are assigned different tasks for optimization. Each particle also possesses a search direction for feature fixing. In Figure 6.1, particles with upward and downward arrows search for solutions with more and fewer features, respectively, similar to BDFF. During evolution, knowledge can be transferred from one task to another, aiding in the search process for both tasks.

Given that both tasks perform feature selection on the same problem, they share a lot of similar knowledge. For example, features selected in one task are likely to be selected in the other task as well. Therefore, KT between these two tasks can be more reasonable and effective.

6.3.2 Multi-Task Knowledge Transfer

The multi-task KT in MBDPSO is based on the MFEA paradigm. In MFEA, the skill factor τ_i denotes the index of the task solved by particle p_i. MBDPSO adopts this definition. If the value of τ_i is 1, it indicates that p_i is assigned to the first task. Otherwise, if the value of τ_i is 2, it indicates that p_i solves the second task. Then, the assortative mating process in MBDPSO is designed as follows:

$$x_{i,d} = \begin{cases} N\left(\frac{pbest_{i,d} + lbest_{\tau_i,d}}{2}, \left|pbest_{i,d} - lbest_{\tau_i,d}\right|\right), & \text{if } r > rmp \\ N\left(\frac{pbest_{i,d} + lbest_{-\tau_i+3,d}}{2}, \left|pbest_{i,d} - lbest_{-\tau_i+3,d}\right|\right), & \text{otherwise} \end{cases} \tag{6.3}$$

In equation 6.3, *rmp* represents the random mating probability, $lbest_\tau$ is the local best position found by the task τ, with $\tau_i \in \{1,2\}$. If the random value r is greater than *rmp*, p_i will generate a new $x_{i,d}$ using the *lbest* of its task. Otherwise, p_i will use the *lbest* of the other task, facilitating KT from one task to the current one. Additionally, to

control the frequency of KT the value of *rmp* is adjusted during the evolution process of MBDPSO using a linearly increasing strategy.

$$rmp = rmp_{\min} + \frac{g}{G} \times (rmp_{\max} - rmp_{\min}) \tag{6.4}$$

Where rmp_{min} and rmp_{max} represent the minimum and maximum values of *rmp*, respectively, *g* is the current iteration number, and *G* is the total number of iterations in the entire evolution process. Gradually increasing the value of *rmp* helps particles focus more on learning from their tasks at the beginning and then facilitates KT later on.

Algorithm 1 Particle Position Update for MBDPSO

Input: The *i*th particle to be updated p_i, the total number of features D, the current position x_i of p_i, the personal best position $pbest_i$ found by p_i so far

Output: The updated particle p_i

Begin

1 **For** $j = 1$ to D **Do**

2 **If** $x_{i,j}$ meets the conditions of feature fixation **Then**

2 $x_{i,j} \leftarrow pbest_{i,j}$;

4 **Else**

5 Update $x_{i,j}$ with Eq. (6.4);

6 **End If**

7 **End For**

8 **Return** p_i;

End

The strategy for MBDPSO to update particle positions is described in Algorithm 1. For each dimension $x_{i,j}$ of xi, if $x_{i,j}$ meets the conditions of feature selection mentioned in Section 6.2.2, its selection state will be fixed and directly use the value of $pbest_{i,j}$. If $x_{i,j}$ is not fixed, then it will be updated adopting the assortative mating in Eq. (6.4).

6.3.3 Complete MBDPSO

The complete implementation of MBDPSO is described in Algorithm 2. It begins with sorting all features based on their SU values with the class labels. Then, it initializes the search directions, assigns tasks, and initializes the positions of each particle. In each iteration, particles search for the optimal solution based on their respective task directions. Through vertical cultural transmission, particles can change their tasks by utilizing knowledge from other tasks when generating new positions. The

evaluation of particles depends on their tasks, with particles in the first task focusing on promising features, and those in the second task considering the entire feature set. After each iteration, the local best (lbest) and global best (gbest) for each task are updated. Finally, when the maximum evaluation count is reached, MBDPSO returns gbest and terminates the algorithm.

Algorithm 2 MBDPSO

Input: The total number of features D, the number of promising features K, the maximum number of generations G, the size of the swarm P, the maximum number of fitness evaluations MAX_FE

Output: The global optimal solution ***gbest*** found by the swarm

Begin

1 Rank features with SU and generate two tasks;
2 Initialize search direction with Eq. (6.2);
3 Randomly assign a task for each particle;
4 Randomly initialize the position x of each particle;
5 Set the number of fitness evaluations $FEs \leftarrow 0$;
6 $g \leftarrow 1$;
7 **While** $FEs < MAX\_FE$ **Do**
8 **For** $i = 1$ to P **Do**
9 Update x_i of particle p_i with Algorithm 1;
10 Employ vertical cultural transmission to p_i;
11 Evaluate xi according to τi and then update $pbest_i$;
12 **End For**
13 Update $lbest_1$, $lbest_2$, and gbest;
14 $g \leftarrow g + 1$;
15 **End while**
16 **Return** *gbest*;

End

The time complexity of MBDPSO is $O(G \times P \times (T_u + T_e))$, where G is the maximum number of iterations, P is the population size, T_u is the time complexity for updating each particle's new position, and T_e is the time complexity for evaluating each particle. With the feature fixation technique, Tu is less than $O(D)$ since not all features need to be updated. Additionally, using the k-nearest neighbors (k-NN) method for evaluating candidate solutions in classification problems [297], with S samples in the dataset, the time complexity of k-NN is $O(S^2K)$ in the first task where only K (K<D) promising

features are considered. Overall, the time complexity is greater than O(S^2K) but less than O(S^2D).

6.4 Experiment Results and Discussion

In this section, experiments are conducted to evaluate the performance of the proposed MBDPSO algorithm. Ten commonly used feature selection datasets are employed to compare MBDPSO with other Particle Swarm Optimization (PSO)-based feature selection algorithms.

6.4.1 Datasets

Basic information for the ten public datasets is listed here, with "#" representing the number of respective items, as detailed in Table I. All datasets used are for classification problems and are sourced from [271] and [276]. These datasets include face image data and biological data, characterized by small sample sizes and large numbers of features. Therefore, the classification on these datasets poses a challenging task. Additionally, the datasets used contain both discrete and continuous data, allowing for comprehensive testing of MBDPSO's performance across different data types without the need for a specific design for each data type.

6.4.2 Experimental Setup

Six Particle Swarm Optimization (PSO)-based feature selection algorithms were selected for comparison to test the performance of the proposed MBDPSO algorithm. BPSO [279] serves as the baseline method, being the first PSO algorithm applicable to feature selection problems. BBPSO [272] is a widely used PSO variant that, after simple discretization, can be employed for feature selection. MIBBPSO [266], ISBPSO [267], and HFS-C-P [268] are recently proposed PSO-based algorithms that have shown good performance on high-dimensional feature selection problems. BBPSO-ACJ-BDFF [269] is a specific implementation version of the BDFF framework, demonstrating good search performance on high-dimensional feature sets. The parameter settings for the algorithms used for comparison are consistent with those in their original papers. However, for fair comparison, the maximum fitness evaluation times (MAX_FE) was limited for each algorithm to 5000, as the values of the population size P differ across

algorithms. For MBDPSO, P was set to 20, the value of rmp ranges from 0 to 0.6, and K is set to $0.2D$.

Table 6.1 Basic information of 10 data sets

Dateset	#Samples	#Features	#Classes	Data Type
Colon	62	2000	2	discrete
WarpAR10P	130	2400	10	continuous
GLIOMA	50	4434	4	continuous
Leukemia_1	72	5327	3	discrete
9_Tumor	60	5726	9	continuous
TOX_171	171	5748	4	continuous
Brain_Tumor_1	90	5920	5	continuous
Nci9	60	9712	9	discrete
Arcene	200	10000	2	continuous
Orlraws10P	100	10304	10	continuous

K-NN was chosen as the classifier for the classification problem, with the parameter k set to 5. In the experiment, 70% of the samples from each dataset were used as the training dataset, while the remaining 30% were used as the testing dataset. A 5-fold cross-validation method was employed, using k-NN to evaluate the classification accuracy of all algorithms on the training dataset during the training process. Then, during the testing process, the best feature subset found by each algorithm was tested on the testing dataset to obtain the classification accuracy for comparison using k-NN. To reduce experimental errors, each algorithm was independently run 20 times with different random seeds. Wilcoxon rank-sum test was employed 错误!未找到引用源。 to analyze significant differences between MBDPSO and other algorithms, with a significance level set to 0.05. Additionally, Feature Selection Toolbox 3 [281], an open-source library based on C++, which implemented all algorithms on a platform with an Intel Core i7-10700F CPU @2.90GHz and a total memory of 8GB was used.

6.4.3 Compare the results and discussion

In Table II, the average classification accuracy of the best feature subset found by MBDPSO and other comparison algorithms in 20 runs on 10 public datasets is recorded. The numbers in parentheses indicate the ranking of the algorithms. Symbols “+” , “≈”, and “-” indicate whether MBDPSO achieved significantly higher, similar, or significantly lower classification accuracy compared to the comparison algorithms,

respectively. Among all algorithms, MBDPSO exhibited the best classification accuracy performance. On 9 out of 10 datasets, MBDPSO demonstrated higher or similar classification accuracy compared to BPSO, BBPSO, MIBBPSO, and HFS-C-P. Compared to ISBPSO, MBDPSO performed significantly better on 4 datasets and similarly on 4 datasets. Additionally, MBDPSO outperformed BBPSO-ACJ-BDFF on 6 datasets, achieving lower results on only 2 datasets. Moreover, bold numbers indicate the best classification accuracy among all algorithms in Table II. MBDPSO found 5 out of the 10 best results among all algorithms, the highest among all algorithms. The rank sum of algorithms on the 10 datasets was also calculated to demonstrate their overall performance. Across all datasets, MBDPSO had a rank sum of 24, significantly lower than other algorithms.

Table 6.2. Classification accuracy of MBDPSO and other comparative algorithms on 10 datasets (bold numbers indicate the best results).

Dataset	MDBPSO	BPSO	BBPSO	MIBBPSO	ISBPSO	HFS-C-P	BBPSO-ACJ-BDFF
Colon	**0.737 (1)**	0.695 (5, +)	0.689 (6, +)	0.674 (7, +)	0.705 (2, ≈)	0.700 (3, +)	0.697 (4, +)
WarpAR10P	0.540 (2)	0.401 (7, +)	0.416 (6, +)	0.458 (5, +)	0.510 (4, +)	0.515 (3, +)	**0.630 (1, –)**
GLIOMA	**0.753 (1)**	0.717 (3, +)	0.714 (4, +)	0.714 (4, +)	0.703 (7, +)	0.728 (2, ≈)	0.706 (6, +)
Leukemia_1	0.954 (3)	0.939 (5, ≈)	**0.965 (1, ≈)**	0.946 (4, ≈)	0.930 (6, ≈)	0.861 (7, +)	0.959 (2, ≈)
9_Tumor	0.391 (7)	0.430 (6, –)	0.445 (4, –)	0.448 (3, –)	0.452 (2, –)	0.439 (5, –)	**0.489 (1, –)**
TOX_171	**0.711 (1)**	0.627 (7, +)	0.698 (5, ≈)	0.707 (2, ≈)	0.705 (3, ≈)	0.705 (4, ≈)	0.674 (6, +)
Brain_Tumor_1	**0.791 (1)**	0.766 (6, +)	0.761 (7, +)	0.780 (3, ≈)	0.773 (4, +)	0.786 (2, ≈)	0.768 (5, +)
Nci9	**0.486 (1)**	0.370 (7, +)	0.380 (6, +)	0.389 (4, +)	0.420 (2, +)	0.418 (3, +)	0.384 (5, +)
Arcene	0.817 (2)	0.807 (3, ≈)	0.803 (4, ≈)	0.801 (5, ≈)	**0.829 (1, ≈)**	0.754 (7, +)	0.788 (6, +)
Orlraws10P	0.875 (5)	0.873 (6, ≈)	0.870 (7, ≈)	0.880 (3, ≈)	**0.897 (1, –)**	0.888 (2, ≈)	0.875 (4, ≈)
+/≈/-	NA	6/3/1	5/4/1	4/5/1	4/4/2	5/4/1	6/2/2
Rank Sum	24	55	50	40	32	38	40

In addition to classification accuracy, the number of selected features is also a crucial indicator in feature selection. Table III presents the average number of selected features in the best feature subset found by each algorithm, where symbols "+", "≈", and "-" indicate whether MBDPSO significantly reduced, was similar to, or significantly increased the number of selected features compared to the comparison algorithms, respectively. MBDPSO found feature subsets much smaller than BPSO, BBPSO, MIBBPSO, and ISBPSO on all datasets. Compared to HFS-C-P, MBDPSO

required fewer features on 7 datasets and a similar number on the Arcene dataset. On 9 out of 10 datasets, MBDPSO found feature subsets much smaller than BBPSO-ACJ-BDFF. While MBDPSO selected more features than BBPSO-ACJ-BDFF on the WarpAR10P dataset, it still selected significantly fewer features compared to other comparison algorithms. Overall, MBDPSO found the smallest feature subset on 6 out of 10 datasets and achieved a rank sum of 14, indicating its superior performance.

Table 6.3 Average number of selected features by algorithms on 10 datasets.

Dataset	MDBPSO	BPSO	BBPSO	MIBBPSO	ISBPSO	HFS-C-P	BBPSO-ACJ-BDFF
Colon	**152.7 (1)**	1206.9 (7, +)	996.2 (6, +)	624.7 (4, +)	292.5 (3, +)	243.3 (2, +)	671.3 (5, +)
WarpAR10P	96.7 (2)	1488.6 (7, +)	1191.7 (6, +)	506.4 (5, +)	343.8 (3, +)	374.2 (4, +)	**21.9 (1, –)**
GLIOMA	**538.4 (1)**	2700.8 (7, +)	2199.8 (6, +)	1943.4 (5, +)	1694.0 (4, +)	666.4 (2, +)	1321.7 (3, +)
Leukemia_1	590.4 (2)	3290.2 (7, +)	2651.8 (5, +)	2918.0 (6, +)	2143.2 (4, +)	**218.0 (1, –)**	1124.0 (3, +)
9_Tumor	**610.4 (1)**	3532.1 (6, +)	2859.3 (5, +)	2798.6 (4, +)	1142.9 (2, +)	4326.0 (7, +)	1198.7 (3, +)
TOX_171	448.0 (2)	3582.2 (7, +)	2872.0 (6, +)	1280.6 (4, +)	653.8 (3, +)	**204.1 (1, –)**	1375.4 (5, +)
Brain_Tumor_1	**454.6 (1)**	3638.7 (7, +)	2967.2 (6, +)	1500.9 (4, +)	1365.9 (3, +)	1242.2 (2, +)	1977.4 (5, +)
Nci9	**500.6 (1)**	5789.1 (7, +)	4857.4 (6, +)	3263.5 (4, +)	1412.5 (2, +)	2524.1 (3, +)	3705.9 (5, +)
Arcene	661.7 (2)	6176.4 (7, +)	4991.2 (6, +)	4276.7 (5, +)	2836.1 (4, +)	**616.5 (1, ≈)**	1301.4 (3, +)
Orlraws10P	**867.6 (1)**	5905.7 (7, +)	5133.7 (6, +)	3454.4 (3, +)	1613.8 (2, +)	3522.8 (4, +)	4125.7 (5, +)
+/≈/-	NA	10/0/0	10/0/0	10/0/0	10/0/0	7/1/2	9/0/1
Rank Sum	14	69	58	44	30	27	38

Table 6.4 Comparison of running time

Dataset	MDBPSO	BPSO	BBPSO	MIBBPSO	ISBPSO	HFS-C-P	BBPSO-ACJ-BDFF
Colon	0.26 (3)	0.45 (6, +)	0.36 (5, +)	0.46 (7, +)	**0.15 (1, –)**	0.15 (2, –)	0.29 (4, +)
WarpAR10P	0.91 (4)	1.57 (6, +)	1.26 (5, +)	14.84 (7, +)	0.72 (2, –)	0.74 (3, –)	**0.66 (1, –)**
GLIOMA	**0.76 (1)**	1.54 (6, +)	1.22 (5, +)	7.92 (7, +)	0.98 (4, +)	0.77 (2, ≈)	0.94 (3, +)
Leukemia_1	1.47 (3)	2.46 (6, +)	1.91 (5, +)	7.97 (7, +)	1.60 (4, +)	**0.81 (1, –)**	1.18 (2, –)
9_Tumor	1.68 (3)	2.65 (5, +)	2.05 (4, +)	19.07 (7, +)	**0.91 (1, –)**	3.75 (6, +)	1.46 (2, –)
TOX_171	3.25 (3)	6.07 (6, +)	4.61 (5, +)	148.27 (7, +)	1.58 (2, –)	**0.69 (1, –)**	3.32 (4, ≈)
Brain_Tumor_1	1.82 (3)	3.43 (6, +)	2.66 (5, +)	43.76 (7, +)	**1.39 (1, –)**	1.74 (2, ≈)	2.12 (4, +)
Nci9	3.15 (2)	6.98 (6, +)	5.43 (5, +)	8.97 (7, +)	**1.79 (1, –)**	4.34 (3, +)	4.41 (4, +)
Arcene	7.96 (3)	14.83 (6, +)	11.95 (5, +)	451.34 (7, +)	9.27 (4, +)	7.87 (2, ≈)	**7.04 (1, ≈)**
Orlraws10P	4.95 (2)	8.50 (6, +)	6.88 (4, +)	170.88 (7, +)	**2.86 (1, –)**	8.39 (5, +)	5.89 (3, +)
+/≈/-	NA	10/0/0	10/0/0	10/0/0	3/0/7	3/3/4	5/2/3
Rank Sum	27	59	48	70	21	27	28

The average running time of 20 runs was also recorded and the results are presented in Table IV. Symbols "+" and "_" denote significantly more or less time spent by MBDPSO compared to the respective algorithms, while "≈" indicates no significant difference between MBDPSO and the compared algorithms. MBDPSO required less time to complete the search on these 10 datasets compared to BPSO, BBPSO, and MIBBPSO. Compared to BBPSO-ACJ-BDFF, MBDPSO had shorter running times on 5 datasets and similar times on the other 2 datasets. Additionally, MBDPSO exhibited similar running times to HFS-C-P, with a rank sum of 27. Compared to ISBPSO, MBDPSO required less time to search for solutions. However, in most cases, MBDPSO achieved higher classification accuracy and fewer selected features.

6.4.4 Composition Analysis of MBDPSO

In this subsection, the two components of MBDPSO are further analyzed: the BDFF framework and the EMTO technique. Firstly, the MBDPSO-w/o-EMTO variant was designed, retaining only the second task of MBDPSO while eliminating KT between different tasks. Then, the BDFF component was removed from the particle update process, resulting in the MBDPSO-w/o-BDFF variant. The experimental results are presented in Table V.

Compared to MBDPSO-w/o-EMTO, MBDPSO exhibited an 8.22% improvement in classification accuracy and, on average, reduced the number of selected features by 76.66%. In 7 out of 10 datasets, MBDPSO significantly enhanced its classification accuracy through the integration of the EMTO technique. Across all datasets, MBDPSO consistently reduced the number of selected features relative to MBDPSO-w/o-EMTO. Hence, the EMTO component in MBDPSO generally aids particles in discovering smaller feature subsets and enhancing classification performance. Compared to MBDPSO-w/o-BDFF, MBDPSO required significantly fewer selected features in 8 datasets. Overall, MBDPSO achieved an average improvement of 1.30% in classification accuracy and reduced the number of selected features by an average of 26.14% across 10 datasets. Thus, the BDFF component used in MBDPSO can reduce the number of selected features while slightly improving classification performance.

Table 6.5. Experimental results of component analysis.

Dataset	MBDPSO		MBDPSP-w/o-EMTO		MBDPSP-w/o-BDFF	
	Accuracy	Features	Accuracy	Features	Accuracy	Features
Colon	**0.737**	**152.7**	0.666 (+)	797.7 (+)	0.726 (=)	197.4 (+)
WarpAR10P	**0.540**	96.7	0.411 (+)	431.4 (+)	0.525 (=)	232.6 (+)
GLIOMA	**0.753**	538.4	0.711 (+)	1919.1 (+)	0.733 (=)	702.5 (+)
Leukemia_1	0.954	590.4	0.935 (=)	1586.4 (+)	**0.959 (=)**	**535.6 (=)**
9_Tumor	0.391	610.4	**0.450 (–)**	2577.8 (+)	0.386 (=)	**580.5 (=)**
TOX_171	**0.711**	**448.0**	0.673 (+)	1815.9 (+)	0.700 (=)	913.3 (+)
Brain_Tumor_1	**0.791**	**454.6**	0.761 (+)	2303.5 (+)	0.789 (=)	592.7 (+)
Nci9	**0.486**	500.6	0.366 (+)	4351.9 (+)	0.468 (=)	964.5 (+)
Arcene	0.817	**661.7**	0.797 (+)	2376.5 (+)	**0.824 (=)**	996.7 (+)
Orlraws10P	**0.875**	**867.6**	0.873 (=)	4531.4 (+)	0.872 (=)	1037.1 (+)
+/≈/-			7/2/1	10/0/0	0/10/0	8/2/0

6.4.5 Impact of Parameter K

Table 6.6. Experimental results of different variants of MBDPSO with various *K* values

Dataset	Accuracy					Features				
	K = 0.1D	K = 0.2D	K = 0.3D	K = 0.4D	K = 0.5D	K = 0.1D	K = 0.2D	K = 0.3D	K = 0.4D	K = 0.5D
Colon	0.661 (5)	**0.737 (1)**	0.732 (2)	0.729 (3)	0.713 (4)	683.8 (5)	**152.7 (1)**	253.7 (2)	339.5 (3)	497.9 (4)
WarpAR10P	**0.594 (1)**	0.540 (3)	0.575 (2)	0.516 (5)	0.521 (4)	**36.4 (1)**	96.7 (3)	84.6 (2)	313.6 (5)	254.0 (4)
GLIOMA	0.753 (2)	**0.753 (1)**	0.708 (5)	0.708 (3)	0.708 (4)	**104.8 (1)**	538.4 (2)	1407.5 (4)	987.9 (3)	1497.9 (5)
Leukemia_1	0.946 (5)	0.954 (2)	**0.965 (1)**	0.952 (3)	0.948 (4)	688.9 (2)	**590.4 (1)**	889.6 (3)	1276.6 (4)	1350.9 (5)
9_Tumor	0.373 (5)	0.391 (4)	0.480 (2)	**0.500 (1)**	0.457 (3)	**291.3 (1)**	610.4 (2)	921.1 (3)	965.4 (4)	1328.2 (5)
TOX_171	0.708 (2)	**0.711 (1)**	0.670 (4)	0.638 (5)	0.678 (3)	**164.1 (1)**	448.0 (2)	878.2 (3)	1052.4 (4)	1268.9 (5)
Brain_Tumor_1	**0.796 (1)**	0.791 (2)	0.779 (4)	0.775 (5)	0.780 (3)	**223.9 (1)**	454.6 (2)	726.7 (3)	1087.6 (4)	1245.5 (5)
Nci9	**0.520 (1)**	0.486 (2)	0.441 (3)	0.395 (5)	0.395 (4)	**348.6 (1)**	500.6 (2)	991.8 (3)	1750.7 (5)	1577.3 (4)
Arcene	**0.847 (1)**	0.817 (2)	0.801 (4)	0.796 (5)	0.802 (3)	**403.1 (1)**	661.7 (2)	2163.1 (3)	2848.6 (5)	2199.1 (4)
Orlraws10P	0.877 (4)	0.875 (5)	0.885 (3)	**0.900 (1)**	0.900 (1)	1126.6 (2)	**867.6 (1)**	1145.7 (3)	1374.5 (4)	2407.8 (5)
Rank Sum	27	23	30	36	33	16	18	29	41	46

The number of hopeful features, *K*, is a crucial parameter in MBDPSO, determining the size of the hopeful feature subset in the first task of MBDPSO. On one hand, larger values of *K* consistently slow down the search for hopeful features. On the other hand, smaller values of *K* may weaken the diversity of the population. To investigate the impact of K, different variants of MBDPSO was compared with varying *K* values, and the experimental results are listed in Table VI. Overall, MBDPSO performs best in terms of classification accuracy and ranks when *K*=0.2D, with a ranking sum of 23. As the value of *K* decreases, the number of selected features also

decreases in most cases. Therefore, smaller values of K can effectively reduce the number of selected features discovered by MBDPSO. Taking into account both classification accuracy and the number of selected features, the value of K is suggested to be set to 0.2D.

6.5 Conclusion

The chapter introduces MBDPSO, which combines EMTO technology and the BDFF framework for high-dimensional feature selection. Initially, MBDPSO devises two correlated tasks to leverage prior knowledge from the BDFF framework for feature selection. Subsequently, MBDPSO proposes a KT strategy, effectively exchanging knowledge between the tasks, thereby enhancing its performance. Experimental results on 10 public classification datasets demonstrate that MBDPSO can find smaller feature subsets with higher classification accuracy compared to other algorithms [265].

However, despite the effectiveness of the KT strategy in MBDPSO, each task can only learn from the local optimum of the other task, limiting the scope of KT. For future work, advanced KT techniques such as orthogonal KT [236], meta-knowledge transfer [234], and bi-objective KT [235] could be considered to further enhance the performance of MBDPSO on complex feature selection problems. Additionally, comparing MBDPSO with more recent algorithms, especially non-PSO algorithms, could provide a more comprehensive validation of its performance.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusions

In conclusion, this thesis has done research in both theoretical research and application research in the field of optimization.

The theoretical contributions focus on integrating KT into EC for efficiently solving optimization problems, addressing challenges in DOP with the HIDE algorithm, and shifting the perspective in MTOP with the MCOP algorithm. These theoretical researches provide valuable frameworks for tackling complex optimization challenges.

On the application aspect, this thesis has presented effective solutions to real-world problems. In DOP applications, the KLRP-MA algorithm offers a robust solution to the dynamic nature of urban bike-sharing systems, ensuring swift re-optimization of planned routes and accelerating convergence through KT mechanisms. In MTOP applications, the integration of MTOP with the BDFF framework within the MBDPSO algorithm significantly enhances feature selection capabilities across multiple tasks, demonstrating improved performance in solving complex optimization problems.

7.2 Future Work

In the research area of DOP, future research endeavors can explore both theoretical and application directions. On the theoretical front, investigating strategies for subpopulations to independently explore partial search spaces within collaborative environments holds promise for enhancing algorithm efficiency. Furthermore, addressing challenges related to multi-modal, multi-objective, and expensive characterics could encourage more comprehensive solutions in dynamic optimization. Application-wise, integrating data-driven optimization techniques and multi-objective optimization strategies into DOP algorithms could facilitate their deployment in real-world scenarios, such as urban logistics and resource management.

Future research directions of MTOP also encompass theoretical advancements and

practical applications. Theoretical investigations may delve into advancing KT techniques within MTOP algorithms to enable more effective learning across multiple tasks. Moreover, exploring the integration of MTOP with engineering applications like logistics, supply chain management, and traffic optimization holds promise for addressing complex real-world challenges. Additionally, further research efforts could focus on extending the scope of MTOP algorithms to accommodate diverse optimization objectives and constraints in practical configurations.

In future research, techniques from dynamic optimization and multi-task optimization can be mutually beneficial to improve the efficiency and effectiveness of solving complex optimization problems. Although current searches in IEEE Xplore do not reveal papers combining dynamic and multi-task, this is undoubtedly a promising direction. From the problem perspective, dynamic optimization can be seen as a multi-task optimization problem that changes over time, while multi-task optimization can be viewed as dynamic optimization in the spatial dimension. Real-time decision-making methods, adaptability techniques, and the use of historical data in dynamic optimization can enhance the real-time response and adaptability of multi-task optimization systems. Conversely, multi-task optimization techniques such as multi-objective optimization algorithms, knowledge transfer, and co-evolution mechanisms can provide valuable solutions and insights for dynamic optimization. Additionally, common techniques like metaheuristic algorithms, reinforcement learning, and machine learning methods are crucial in handling complex data relationships and feature selection in dynamic and multi-task environments. For example, in intelligent transportation systems, the combination of these two approaches can improve real-time traffic signal adjustments and overall traffic flow balance. Therefore, further exploring and integrating these optimization techniques will provide strong support for addressing increasingly complex and dynamic optimization challenges in the real world.

BIBLIOGRAPHY

[1] A. Telikani, A. Tahmassebi, W. Banzhaf, and A. H. Gandomi, "Evolutionary machine learning: a survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1-35, Nov. 2022.

[2] X. Pang, Y. Ge, and K. Wang, "Genetic algorithm for patient assignment optimization in cloud healthcare system," in *Proc. International Conference on Health Information Science*, Oct. 2022, vol. 13705, pp. 1-12.

[3] S. Supriya, S. Siuly, H. Wang, and Y. Zhang, "EEG sleep stages analysis and classification based on weighed complex network features," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 2, pp. 236-246, Apr. 2021.

[4] Y. F. Ge, H. Wang, J. Cao, and Y. Zhang, "An information-driven genetic algorithm for privacy-preserving data publishing," in *Proc. International Conference on Health Information Science*, Nov. 2022, pp. 1-15.

[5] Y. F. Ge, E. Bertino, H. Wang, J. Cao, and Y. Zhang, "Distributed cooperative coevolution of data publishing privacy and transparency," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 1, pp. 1-23, Jan. 2024.

[6] Y. F. Ge, M. Orlowska, J. Cao, H. Wang, and Y. Zhang, "MDDE: multitasking distributed differential evolution for privacy-preserving database fragmentation," *The VLDB Journal*, vol. 31, pp. 957-975, Sep. 2022.

[7] J. Zhang, H. Li, X. Liu, Y. Luo, F. Chen, H. Wang, and L.Chang, "On efficient and robust anonymization for privacy protection on massive streaming categorical information," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 5, pp. 507-520, 1 Sept.-Oct. 2017.

[8] K. Cheng, L. Wang, Y. Shen, H. Wang, Y. Wang, X. Jiang, and H. Zhong, "Secure k-NN query on encrypted cloud data with multiple keys," *IEEE Transactions on Big Data*, vol. 7, no. 4, pp. 689-702, Oct. 2021.

[9] X. Sun, H. Wang, J. Li, and J. Pei, "Publishing anonymous survey rating data," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 379-406, Nov. 2011.

[10] M. You, Y. F. Ge, K. Wang, H. Wang, J. Cao, and G. Kambourakis, "TLEF: two-layer evolutionary framework for t-closeness anonymization," in *Proc. of Web Information Systems Engineering – WISE*, Oct. 2023, vol. 14306, pp. 401-423.

[11] E. Kabir, A. Mahmood, H. Wang, and A. Mustafa, “Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing,” *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 408-417, 1 April-June 2020.

[12] E. Kabir and H. Wang, “A role-involved purpose-based access control model,” *Information Systems Frontiers*, vol. 14, pp. 809-822, Jul. 2012.

[13] J. Yin, M. Tang, J. Cao, H. Wang, M. You, and Y. Lin, “Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning,” *World Wide Web*, vol. 25, pp. 401-423, Jan. 2022.

[14] J. Yin, G. Chen, W. Hong, H. Wang, J. Cao, and Y. Miao, “Empowering vulnerability prioritization: a heterogeneous graph-driven framework for exploitability prediction,” in *Proc. Web Information Systems Engineering – WISE*, Oct. 2023, pp. 401-423.

[15] H. Wang, Y. Zhang, and J. Cao, “Ubiquitous computing environments and its usage access control,” in *Proc. of of the 1st international conference on Scalable information systems*, New York, May 2006, pp. 47-54.

[16] Y. Qin, Q. Z. Sheng, N. J. G. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, “When things matter: a data-centric view of the internet of things,” *arXiv*, Jul. 2014. [Online]. Available: https://arxiv.org/abs/1407.2704.

[17] M. Foth, I. Anastasiu, M. Mann, and P. Mitchell, “From automation to autonomy: technological sovereignty for better data care in smart cities,” *Automating Cities*, pp. 319-343, Jan. 2021.

[18] J. Du, J. Rong, H. Wang, and Y. Zhang, “Neighbor-aware review helpfulness prediction,” *Decision Support Systems*, vol. 148, pp. 113581, Sep. 2021.

[19] H. Jiang, R. Zhou, L. Zhang, H. Wang, and Y. Zhang., “Sentence level topic models for associated topics extraction,” *World Wide Web*, vol. 22, no. 6, pp. 2545-2560, Nov. 2019.

[20] S. Vandehende, S. Georgoulis, W. V. G. M. Proesmans, D. Dai, and L. V. Gool, “Multi-Task learning for dense prediction tasks: a survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3614-3633, Jul. 2022.

[21] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586-5609, Dec. 2022.

[22] Z. Tan, L. Luo, and J. Zhong, “Knowledge transfer in evolutionary multi-task optimization: A survey,” *Applied Soft Computing*, vol. 138, Art. no. 110182, May 2023.

[23] M. Pelikan, M.W. Hauschild, and P.L. Lanzi, “Transfer learning, soft distance-based bias, and the hierarchical boa,” in *Proc. of the International Conference on Parallel Problem Solving from Nature*, Sep. 2012, pp. 173–183.

[24] N. T. Thanh, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, Oct. 2012.

[25] Z. H. Zhan, L. Shi, K. C. Tan, and J. Zhang, “A survey on evolutionary computation for complex continuous optimization,” *Artificial Intelligence Review*, vol. 55, pp. 59–110, Jan. 2022.

[26] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. of the IEEE Congress on Evolutionary Computation*, Jul. 1999, pp. 1875–1882.

[27] K. C. Tan, L. Feng, and M. Jiang, “Evolutionary transfer optimization—a new frontier in evolutionary computation research,” *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 22–33, Feb. 2021.

[28] M. Pelikan, M. W. Hauschild, and P. L. Lanzi, “Transfer learning, soft distance based bias, and the hierarchical boa,” in *Proc. of the International Conference on Parallel Problem Solving from Nature*, Sep. 2012, pp. 173–183.

[29] L. Feng, Y. S. Ong, A. H. Tan, and I. W. Tsang, “ Memes as building blocks: a case study on evolutionary optimization + transfer learning for routing problems,” *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.

[30] A. Gupta, Y. S. Ong, and L. Feng, “Multifactorial evolution: toward evolutionary multitasking,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[31] L. Feng, W. Zhou, L. Zhou, S. Jiang, J. Zhong, B. Da, Z. Zhu, and Y. Wang, “An empirical study of multifactorial PSO and multifactorial DE,” in *Proc. of 2017 IEEE Congress on Evolutionary Computation*, 2017, pp. 921–928.

[32] Y. Chen, J. Zhong, and M. Tan, “A fast memetic multi-objective differential evolution for multi-tasking optimization,” in *Proc. of 2018 IEEE Congress on Evolutionary Computation*, Jul. 2018, pp. 1–8.

[33] G. B. Dantzig and N. N. Thapa, *Linear Programming 1: Introduction*. Berlin, Germany: Springer-Verlag, 1997.

[34] Z. Dostl, *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer-Verlag, 2009.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[36] J. H. Holland, "Outline for a logical theory of adaptive system," *Journal of the Association for Computing Machinery*, vol. 3, pp. 297-314, 1962.

[37] D. B. Fogel, *Evolutionary computation: Toward a new philosophy of machine intelligence*, IEEE Press Series on Computational Intelligence, 1995.

[38] R. Salomon, "Evolutionary algorithms and gradient search: similarities and differences," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 2, pp. 45-55, 1998.

[39] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, pp. 476-482, 2015.

[40] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 1-16, Feb. 2011.

[41] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms," *California Inst. Technol. Technical Report Caltech Concurrent Comput. Prog. Rep.* 826, 1989.

[42] A. Gupta and Y. S. Ong, *Memetic Computation: The Mainspring of Knowledge Transfer in a Data-Driven Optimization Era*, Springer-Verlag, 2019.

[43] L. Feng, Y. Ong, M. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between CVRP and CARP," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 644-658, Oct. 2015.

[44] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, Dec. 1997.

[45] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[46] M. Bilal, M. Pant, H. Zaheer, L. G. Hernandez, and A. Abraham, "Differential

evolution: A review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, pp. 1-24, Apr. 2020.

[47] J. Ilonen, J. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, pp. 93–105, Feb. 2003.

[48] N. Karaboga, "Digital IIR filter design using differential evolution algorithm," *EURASIP Journal on Advanced Signal Processing*, vol. 2005, Art. no. 856824, 2005.

[49] S. Chakraborty, A. K. Saha, A. E. Ezugwu, J. O. Agushaka, R. A. Zitar, and L. Abualigah, "Differential evolution and its applications in image processing problems: a comprehensive review," *Archives of Computational Methods in Engineering*, vol. 30, pp. 985–1040, 2023.

[50] W. Gong, Z. Cai, and D. Liang, "Engineering optimization by means of an improved constrained differential evolution," *Computer Methods in Applied Mechanics and Engineering*, vol. 268, pp. 884-904, Jan. 2014.

[51] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, Nov. 1995, vol.4, pp. 1942-1948.

[52] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in MHS'95. *Proc. of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, Oct. 1995, pp. 39-43.

[53] R. C. Eberhart and Y. Shi, "Guest editorial special issue on particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 201-203, Jun. 2004.

[54] X. Yao, "Evolving artificial neural networks," in *Proc. of the IEEE*, Sep. 1999, vol. 87, no. 9, pp. 1423-1447.

[55] R. C. Eberhart and X. Hu, "Human tremor analysis using particle swarm optimization," in *Proc. of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, 1999, vol. 3, pp. 1927-1930.

[56] G. C. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization," *International Journal of Production Research*, vol. 42, no. 3, pp. 473–491, 2004.

[57] B. Liu, L. Wang, and Y. H. Jin, "An effective PSO-based memetic algorithm for

flow shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)*, vol. 37, no. 1, pp. 18-27, Feb. 2007.

[58] T. Tsukada, T. Tamura, S. Kitagawa, and Y. Fukuyama, "Optimal operational planning for cogeneration system using particle swarm optimization," in *Proc. of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, Indianapolis, IN, USA, 2003, pp. 138-143.

[59] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms," *California Institute of Technology, Technical Report Caltech Concurrent Computation Program Report 826*, 1989.

[60] R. Dawkins, *The selfish gene*, New York: Oxford University Press, 1976.

[61] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474-488, Oct. 2005.

[62] J. E. Smith, "Coevolving memetic algorithms: a review and progress report," *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)*, vol. 37, no. 1, pp. 6-17, Feb. 2007.

[63] P. Assiroj, H. L. H. S. Warnars, E. Abdurachman, A. I. Kistijantoro, and A. Doucet, "The implementation of memetic algorithm on image: a survey," *Journal of Mathematical and Computational Science*, vol. 11, pp. 6872-6896, 2021.

[64] R. Kumar and M. Memoria, "A review of memetic algorithm and its application in traveling salesman," *International Journal on Emerging Technologies*, vol. 11, no. 2, pp. 1110-1115, 2020.

[65] Y. F. Ge, Y. J. Gong, S. Kwong, H. Wang, and J. Zhang, "A niching memetic algorithm for multi-solution traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 508-522, Jun. 2020.

[66] P. Moscato and L. Mathieson, "Memetic algorithms for business analytics and data science: a brief survey," *Business and Consumer Analytics: New Ideas*, P. Moscato and N. de Vries, Eds. Springer, Cham, 2019, pp. 545-608.

[67] J. Yin, M. Tang, J. Cao, and H. Wang, "Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description," *Knowledge-Based Systems*, vol. 210, Art. no. 106529, Dec. 2020.

[68] N. Krasnogor and J. Smith, "Multimeme algorithms for the structure prediction

and structure comparison of proteins," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, 2002, pp. 42-44.

[69] A. Quintero and S. Pierre, "A memetic algorithm for assigning cells to switches in cellular mobile networks," *IEEE Communications Letters*, vol. 6, no. 11, pp. 484-486, Nov. 2002.

[70] X. Yao, *Knowledge Incorporation in Evolutionary Computation*. Berlin, Germany: Springer-Verlag, 2005.

[71] Y. F. Ge, W. J. Yu, J. Cao, H. Wang, Z. H. Zhan, Y. Zhang, and J. Zhang, "Distributed memetic algorithm for outsourced database fragmentation," *IEEE Transactions on Cybernetics*, vol. 51, no. 10, pp. 4808-4821, Oct. 2021.

[72] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.

[73] J. Yin, M. Tang, J. Cao, H. Wang, and M. Alazab, "Knowledge-driven cybersecurity intelligence: software vulnerability coexploitation behavior discovery," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, pp. 5593-5601, Apr. 2023.

[74] J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer learning for speech recognition on a budget," in *Proc. of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 168-177.

[75] C. Sferrazza and R. DAndrea, "Transfer learning for vision-based tactile sensing," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7961-7967.

[76] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019, pp. 15-18.

[77] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 9, May 2016.

[78] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Chapman & Hall, 2014.

[79] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[80] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A

survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019-1034, May 2015.

[81] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[82] M. Ansari Ardeh, Y. Mei, and M. Zhang, "Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 765-779, Aug. 2022.

[83] Y. Guo, G. Chen, M. Jiang, D. Gong, and J. Liang, "A knowledge guided transfer strategy for evolutionary dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1750-1764, Dec. 2023.

[84] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303-317, Jun. 2005.

[85] H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao, "What are dynamic optimization problems?" in *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1550-1557.

[86] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985-2999, Aug. 2008.

[87] A. Rakitianskaia and A. P. Engelbrecht, "Cooperative charged particle swarm optimiser," in *Proc. of IEEE Congress on Evolutionary Computation*, 2008, pp. 933–939.

[88] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, Feb. 2009.

[89] A. B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for dynamic environments," in *Proc. of International Symposium on Intelligence Computation and Applications*, 2009, pp. 422–433.

[90] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "Cellularde: A cellular based differential evolution for dynamic optimization problems," in *Proc. of International Conference on Adaptive and Natural Computing Algorithms*, 2011,

pp. 340–349.

[91] Y. Jin, H. Wang, and C. Sun, “Knowledge transfer in data-driven evolutionary optimization,” Data-Driven Evolutionary Optimization, vol. 975, pp. 273-294, Jun. 2021.

[92] A. Sharifi, V. Noroozi, M. Bashiri, A. B. Hashemi, and M. R. Meybodi, “Two phased cellular PSO: A new collaborative cellular algorithm for optimization in dynamic environments,” in *Proc. of IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[93] S. X. Yang and C. H. Li, “A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments,” *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 959–974, Dec. 2010.

[94] C. H. Li and S. X. Yang, “A general framework of multipopulation methods with clustering in undetectable dynamic environments,” *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 556–577, Aug. 2012.

[95] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, “A competitive clustering particle swarm optimizer for dynamic optimization problems,” *Swarm Intelligence*, vol. 6, pp. 177–206, Jun. 2012.

[96] U. Halder, S. Das, and D. Maity, “A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments,” *IEEE Transactions on Cybernetics*, vol. 43, pp. 881–897, Jun. 2013.

[97] C. H. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. X. Yang, “An adaptive multipopulation framework for locating and tracking multiple optima,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 590–605, Aug. 2016.

[98] X. Luo, Z. Wang, R. Guan, Z. Zhan, and Y. Gao, “A distributed multiple populations framework for evolutionary algorithm in solving dynamic optimization problems,” *IEEE Access*, vol. 7, pp. 44372–44390, Mar. 2019.

[99] W. W. Zhang, W. Z. Zhang, G. G. Yen, and H. L. Jing, “A cluster-based clonal selection algorithm for optimization in dynamic environment,” *Swarm and Evolutionary Computation*, vol. 50, Art. no. 100454, Nov. 2019.

[100] J. Branke, T. Kaussler, C. Smidt, and H. Schmeck, “A multi-population approach to dynamic optimization problems,” in *Proc. of Evolutionary Design and Manufacture*, 2000, pp. 299–307.

[101] C. H. Li and S. X. Yang, “Fast multi-swarm optimization for dynamic optimization problems,” in *Proc. of 4th International Conference on Natural Computation*, 2008, pp. 624–628.

[102] Z. H. Zhan, X. F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, and J. Zhang, “Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[103] Y. Li, Z. Zhan, H. Jin, and J. Zhang, “Cloudde-based distributed differential evolution for solving dynamic optimization problems,” in *Proc. of Tenth International Conference on Intelligent Control and Information Processing*, 2019, pp. 94–99.

[104] L. Cao, L. Xu, and E. D. Goodman, “A collaboration-based particle swarm optimizer with history-guided estimation for optimization in dynamic environments,” *Expert Systems with Applications*, vol. 120, pp. 1–13, Apr. 2019.

[105] W. Liu, Y. Zhou, B. Li, and K. Tang, “Cooperative co-evolution with soft grouping for large scale global optimization,” in *Proc. of IEEE Congress on Evolutionary Computation*, 2019, pp. 318–325.

[106] S. H. Wu, K. J. Du, Z. H. Zhan, H. Wang, and J. Zhang, “Historical information-based differential evolution for dynamic optimization problems,” in *Proc. of IEEE Congress on Evolutionary Computation*, 2021, pp. 1-8.

[107] Y. G. Woldesenbet and G. G. Yen, “Dynamic evolutionary algorithm with variable relocation,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 500–513, Jun. 2009.

[108] Z. H. Zhan, J. J. Li, and J. Zhang, “Adaptive particle swarm optimization with variable relocation for dynamic optimization problems,” in *Proc. of IEEE Congress on Evolutionary Computation*, IEEE, 2014, pp. 1565–1570.

[109] J. Wang, W. Zhang, and J. Zhang, “Cooperative differential evolution with multiple populations for multiobjective optimization,” *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2848–2861, Dec. 2016.

[110] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[111] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, “Orthogonal learning particle swarm

optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[112] X. F. Liu, Z. H. Zhan, and J. Zhang, "Neural network for change direction prediction in dynamic optimization," *IEEE Access*, vol. 6, pp. 72649–72662, Nov. 2018.

[113] G. W. F. Hegel, *The Science of Logic*, transl. A. V. Miller, Humanities Press, 1969.

[114] L. J. Wu, L. Shi, Z. H. Zhan, K. K. Lai, and J. Zhang, "A buffer-based ant colony system approach for dynamic cold chain logistics scheduling," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 6, pp. 1438-1452, Dec. 2022.

[115] X. Sui, Y. Tang, H. He, and J. Wen, "Energy-storage-based low-frequency oscillation damping control using particle swarm optimization and heuristic dynamic programming," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2539-2548, Sep. 2014.

[116] L. Rosenberg, N. Pescetelli, and G. Willcox, "Artificial swarm intelligence amplifies accuracy when predicting financial markets," in *Proc. of 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2017, pp. 58-62.

[117] L. Wang, Z. Luo, H. Tang, S. Guo, and X. Li, "A Novel model for dynamic manufacturing service collaboration on industrial internet," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 12, pp. 11788-11799, Dec. 2023.

[118] Z. Lin, J. Ma, J. Duan, S. E. Li, H. Ma, B. Cheng, and T. H. Lee, "Policy iteration based approximate dynamic programming toward autonomous driving in constrained dynamic environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5003-5013, May 2023.

[119] Z. Peng, J. Wang, D. Wang, and Q. L. Han, "An overview of recent advances in coordinated control of multiple autonomous surface vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 732-745, Feb. 2021.

[120] J. Guo, S. Tang, and Q. Xu, "An improved particle swarm optimization and its application in maneuvering control laws design of the unmanned aerial vehicle," in *Proc. of the 2012 8th International Conference on Natural Computation*, Chongqing, China, 2012, pp. 1107-1111.

[121] Y. Su, L. Xu, and D. Li, "Adaptive fuzzy control of a class of mimo nonlinear system with actuator saturation for greenhouse climate control problem," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 772-788, Apr. 2016.

[122] J. T. B. A. Kessels, M. W. T. Koot, P. P. J. van den Bosch, and D. B. Kok, "Online energy management for hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3428-3440, Nov. 2008.

[123] W. L. Liu, Y. J. Gong, W. N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094-5109, Dec. 2020.

[124] D. Wang, C. Yan, L. Wang, D. Lu, and L. Ma, "Optimization methods for joint capacity and appointment scheduling problem with walk-in patients," in *Proc. of the 2016 35th Chinese Control Conference (CCC)*, Chengdu, China, 2016, pp. 9600-9604.

[125] S. Choi et al., "A wide dynamic range multi-sensor ROIC for portable environmental monitoring systems with two-step self-optimization schemes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2432-2443, Jun. 2021.

[126] S. A. Fernandez, A. A. Juan, J. de Armas Adrián, D. G. e. Silva, and D. R. Terrén, "Metaheuristics in telecommunication systems: network design, routing, and allocation problems," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3948-3957, Dec. 2018.

[127] G. Rang, B. Xu, W. Li, Z. Fan, and Y. Su, "A long short-term memory prediction-based dynamic multi-objective evolutionary optimization algorithm," in *Proc. of the 2023 IEEE Congress on Evolutionary Computation (CEC)*, 2023, pp.1-8.

[128] L. Zhou, L. Feng, A. Gupta, Y. S. Ong, K. Liu, C. Chen, E. Sha, B. Yang, and B. W. Yan, "Solving dynamic vehicle routing problem via evolutionary search with learning capability," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 890-896.

[129] L. Yan, W. Qi, J. Liang, B. Qu, K. Yu, C. Yue, and X. Chai, "Interindividual correlation and dimension-based dual learning for dynamic multiobjective

optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1780-1793, Dec. 2023.

[130] X. Sun, R. Chai, S. Chai, B. Zhang, and A. Tsourdos, "Flexible final-time stochastic differential dynamic programming for autonomous vehicle trajectory optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 5, pp. 6658-6669, Oct. 2023.

[131] M. Jiang, Z. Wang, H. Hong, and G. G. Yen, "Knee point based imbalanced transfer learning for dynamic multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 117-129, Feb. 2020.

[132] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 501-514, Aug. 2018.

[133] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: A forward-looking approach," in *Proc. of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 1201-1208.

[134] Z. Liu and H. Wang, "Improved population prediction strategy for dynamic multi-objective optimization algorithms using transfer learning," *2021 IEEE Congress on Evolutionary Computation (CEC)*, Kraków, Poland, 2021, pp. 103-110.

[135] H. Ma, K. Wu, H. Wang and J. Liu, "Higher order knowledge transfer for dynamic community detection with great changes," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 90-104, Feb. 2024.

[136] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, and K. C. Tan, "A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning," *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3417-3428, Jul. 2021.

[137] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "When and how to transfer knowledge in dynamic multi-objective optimization," in *Proc. of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 2034-2041.

[138] J. Yi, J. Bai, H. He, W. Zhou, and L. Yao, "A multifactorial evolutionary algorithm for multitasking under interval uncertainties," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 908–922, Oct. 2020.

[139] E. Osaba, J. Del Ser, A. D. Martinez, and A. Hussain, “Evolutionary Multitask Optimization: a Methodological Overview, Challenges, and Future Research Directions,” *Cognitive Computation*, vol. 14, pp. 927-954, Apr. 2022.

[140] W. Shi, W. N. Chen, S. Kwong, J. Zhang, H. Wang, T. Gu, H. Yuan, and J. Zhang, “A coevolutionary estimation of distribution algorithm for group insurance portfolio,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 11, pp. 6714-6728, Nov. 2022.

[141] Q. Xu, N. Wang, L. Wang, W. Li, and Q. Sun, “Multi-task optimization and multi-task evolutionary computation in the past five years: A brief review,” *Mathematics*, vol. 9, no. 8, Arc. no. 864, Apr. 2021.

[142] Y. Chen, J. Zhong, L. Feng, and J. Zhang, “An adaptive archive-based evolutionary framework for many-task optimization,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 369–384, Jun. 2020.

[143] S. Huang, J. Zhong, and W. Yu, “Surrogate-assisted evolutionary framework with adaptive knowledge transfer for multi-task optimization,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 9, no. 4, pp. 1930–1944, 1 Oct.–Dec. 2021.

[144] D. Liu, S. Huang, and J. Zhong, “Surrogate-assisted multi-tasking memetic algorithm,” in *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8.

[145] T. Wei and J. Zhong, “A preliminary study of knowledge transfer in multi-classification using gene expression programming,” *Frontiers in Neuroscience*, vol. 13, Arc. no. 1396, Jan. 2020.

[146] Y. Chen, J. Zhong, and M. Tan, “A fast memetic multi-objective differential evolution for multi-tasking optimization,” in *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8.

[147] J. Ding, C. Yang, Y. Jin, and T. Chai, “Generalized multitasking for evolutionary optimization of expensive problems,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 44-58, Jan. 2019.

[148] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and Tan Puay Siew, “Linearized domain adaptation in evolutionary multitasking,” in *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1295-1302.

[149] J. Tang, Y. Chen, Z. Deng, Y. Xiang, and C. P. Joy, "A group-based approach to improve multifactorial evolutionary algorithm," in *Proc. of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018, pp. 3870-3876.

[150] A. Gupta, Y. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652-1665, Jul. 2017.

[151] R. Liaw and C. Ting, "Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems," in *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 2266-2273.

[152] J. Zhong, L. Feng, W. Cai, and Y. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4492-4505, Nov. 2020.

[153] H. Li, Y. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 733-747, Oct. 2019.

[154] Y. Jiang, Z. H. Zhan, K. C. Tan, S. Kwong, and J. Zhang, "Knowledge structure preserving-based evolutionary many-task optimization," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2024.3355781.

[155] Z. Wang, L. Cao, L. Feng, M. Jiang, and K. C. Tan, "Evolutionary multitask optimization with lower confidence bound-based solution selection strategy," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2023.3349250.

[156] W. Lin, Q. Lin, L. Feng, and K. C. Tan, "Ensemble of domain adaptation-based knowledge transfer for evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 2, pp. 388-402, Apr. 2024.

[157] X. Zhou, Z. Wang, L. Feng, S. Liu, K. C. Wong, and K. C. Tan, "Toward evolutionary multitask convolutional neural architecture search," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 3, pp. 682-695, Jun. 2024.

[158] Y. Feng, L. Feng, S. Liu, S. Kwong, and K. C. Tan, "Towards multi-objective high-dimensional feature selection via evolutionary multitasking," *arXiv*, vol. 2401.01563, 2024. doi: 10.48550/arXiv.2401.01563.

[159] K. K. Bali, Y. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 69-83, Feb. 2020.

[160] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. S. Ong, K. C. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457-3470, Sep. 2019.

[161] X. Ma, Q. Chen, Y. Yu, Y. Sun, L. Ma, and Z. Zhu, "A two-level transfer learning algorithm for evolutionary multitasking," *Frontiers in Neuroscience*, vol. 13, pp. 1-15, Jan. 2020.

[162] B. Da, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C. K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," *arXiv*, preprint arXiv:1706.03470, 2017.

[163] C. Yang, J. Ding, K. C. Tan, and Y. Jin, "Two-stage assortative mating for multi-objective multifactorial evolutionary optimization," in *Proc. of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 76–81.

[164] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization-a new frontier in evolutionary computation research," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 22–33, Feb. 2021.

[165] Z. J. Wang, Z. H. Zhan, Y. Lin, W. J. Yu, H. Wang, S. Kwong, and J. Zhang, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 114-128, Feb. 2020.

[166] T. Wei, S. Wang, J. Zhong, D. Liu, and J. Zhang, "A review on evolutionary multitask optimization: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 941-960, Oct. 2022.

[167] A. Gupta, L. Zhou, Y. S. Ong, Z. Chen, and Y. Hou, "Half a dozen real-world applications of evolutionary multitasking, and more," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 49–66, May 2022.

[168] Z. Kai, Y. J. Gong, and J. Zhang, "Real-time traffic signal control with dynamic evolutionary computation," in *Proc. of 2014 IIAI 3rd International Conference on Advanced Applied Informatics*, 2014, pp. 493-498.

[169] T. Huang, Y. J. Gong, W. N. Chen, H. Wang, and J. Zhang, "A probabilistic

niching evolutionary computation framework based on binary space partitioning," *IEEE Transactions on Cybernetics*, vol. 52, no. 1, pp. 51-64, Jan. 2022.

[170] Z. Yang, Y. Jin, and K. Hao, "A bio-inspired self-learning coevolutionary dynamic multiobjective optimization algorithm for Internet of things services," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 675-688, Aug. 2019.

[171] Y. Jia, W. N. Chen, T. Gu, H. Zhang, H. Yuan, Y. Lin, W. J. Yu, and J. Zhang, "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1607-1621, Sep. 2018.

[172] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646-657, Dec. 2006.

[173] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *Proc. of IEEE Congress on Evolutionary Computation*, 2009, pp. 415-422.

[174] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for CEC 2009 competition on dynamic optimization," *University of Leicester, University of Birmingham, Nanyang Technological University, Tech. Rep*., 2008.

[175] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *Proc. of IEEE Congress on Evolutionary Computation*, 2005, vol. 3, pp. 2808-2815.

[176] X. F. Liu, Z. H. Zhan, T. L. Gu, S. Kwong, Z. Lu, H. B. L. Duh, and J. Zhang, "Neural network-based information transfer for dynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1557-1570, May 2020.

[177] J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck, "A multipopulation approach to dynamic optimization problems," in *Proc. of Evolutionary Design and Manufacture*, I. C. Parmee, Ed. Springer, London, 2000, pp. 1-2.

[178] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a

dynamic environment using speciation," in *Proc. of the IEEE Congress on Evolutionary Computation*, 2004, vol. 1, pp. 98-103.

[179] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anticonvergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459-472, Aug. 2006.

[180] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *Proc. of the IEEE Congress on Evolutionary Computation*, 2007, pp. 564-567.

[181] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of ACM*, vol. 9, no. 3, pp. 297-314, Mar. 1962.

[182] Y. S. Ong and A. Gupta, "AIR5: five pillars of artificial intelligence research," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 5, pp. 411–415, Oct. 2019.

[183] A. Gupta and Y. S. Ong, "Insights on transfer optimization: because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, Mar. 2017.

[184] Y. S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, Apr. 2016.

[185] G. Li, Q. Lin, and W. Gao, "Multifactorial optimization via explicit multipopulation evolutionary framework," *Information Sciences*, vol. 512, pp. 1555–1570, Feb. 2020.

[186] L. Feng, L. Zhou, A. Gupta, J. Zhong, Z. Zhu, K. C. Tan, and K. Qin, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3171–3184, Jun. 2019.

[187] L. Feng, Y. Huang, I. W. Tsang, A. Gupta, K. Tang, K. C. Tan, and Y. S. Ong, "Towards faster vehicle routing by transferring knowledge from customer representation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 952-965, Feb. 2022.

[188] H. Wang, L. Feng, Y. Jin, and J. Doherty, "Surrogate-assisted evolutionary multitasking for expensive minimax optimization in multiple scenarios," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 34–48, Feb. 2021.

[189] J. Y. Li, Z. H. Zhan, and J. Zhang, “Evolutionary computation for expensive optimization: a survey,” *Machine Intelligence Research*, vol. 19, no. 1, pp. 3–23, Jan. 2022.

[190] M. Y. Cheng, A. Gupta, Y. S. Ong, and Z. W. Ni, “Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design,” *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 13–24, Sep. 2017.

[191] K. K. Bali, A. Gupta, Y. S. Ong, and P. S. Tan, “Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II,” *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1784–1796, Apr. 2021.

[192] Y. Zheng, Z. Zhu, Y. Qi, L. Wang, and X. Ma, “Multi-objective multifactorial evolutionary algorithm enhanced with the weighting helper-task,” in *Proc. of the 2nd International Conference on Industrial Artificial Intelligence (IAI)*, Jan. 2020, pp. 1-6.

[193] C. Yang, J. Ding, K. C. Tan, and Y. Jin, “Two-stage assortative mating for multi-objective multifactorial evolutionary optimization,” in *Proc. of the 2017 IEEE 56th Annual Conference on Decision and Control*, Jan. 2017, pp. 76-81.

[194] X. Zhang, Z. H. Zhan, W. Fang, P. Qian, and J. Zhang, “Multi population ant colony system with knowledge based local searches for multiobjective supply chain configuration,” *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 512-526, Jun. 2022.

[195] Z. H. Zhan, J. Li, J. Cao, J. Zhang, H. S. Chung, and Y. Shi, “Multiple populations for multiple objectives: a coevolutionary technique for solving multiobjective optimization problems,” *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[196] J. Y. Li, K. J. Du, Z. H. Zhan, H. Wang, and J. Zhang, “Multi-criteria differential evolution: treating multitask optimization as multi-criteria optimization,” in *Proc. of the Genetic and Evolutionary Computation Conference*, 2021, pp. 183-184.

[197] J. Y. Li, Z. H. Zhan, C. Wang, H. Jin, and J. Zhang, “Boosting data-driven evolutionary algorithm with localized data generation,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 923–937, Oct. 2020.

[198] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist

multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[199] W. Wang, X. Zhao, Z. Gong, Z. Gong, Z. Chen, N. Zhang, and W. Wei, "An attention-based deep learning framework for trip destination prediction of sharing bike," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 4601–4610, Nov. 2021.

[200] P. Jiménez, M. Nogal, B. Caulfield, and F. Pilla, "Perceptually important points of mobility patterns to characterise bike sharing systems: The Dublin case," *Journal of Transport Geography*, vol. 54, pp. 228–239, Nov. 2016.

[201] S. Yan, J. R. Lin, Y. C. Chen, and F. R. Xie, "Rental bike location and allocation under stochastic demands," *Computers & Industrial Engineering*, vol. 107, pp. 1–11, Apr. 2017.

[202] S. Maas, M. Attard, and M. A. Caruana, "Assessing spatial and social dimensions of shared bicycle use in a Southern European Island context: The case of Las Palmas de Gran Canaria," *Transportation Research Part A: Policy and Practice*, vol. 140, pp. 81–97, Dec. 2020.

[203] Y. Li and Y. Zheng, "Citywide bike usage prediction in a bike-sharing system," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, pp. 1079–1091, Jun. 2020.

[204] Y. Xing, K. Wang, and J. J. Lu, "Exploring travel patterns and trip purposes of dockless bike-sharing by analyzing massive bike-sharing data in Shanghai, China," *Journal of Transport Geography*, vol. 87, Arc. no. 102787, May 2020.

[205] Y. Liu, W. Y. Szeto, and S. C. Ho, "A static free-floating bike repositioning problem with multiple heterogeneous vehicles, multiple depots, and multiple visits," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 208–242, May 2018.

[206] H. Jia, H. Miao, G. Tian, M. Zhou, Y. Feng, Z. Li, and J. Li, "Multiobjective bike repositioning in bike-sharing systems via a modified artificial bee colony algorithm," *IEEE Transactions on Automation Science and Engineering*, vol. 17, pp. 909–920, Jul. 2020.

[207] R. Guo, Z. Jiang, J. Huang, J. Tao, C. Wang, J. Li, and L. Chen, "BikeNet: Accurate bike demand prediction using graph neural networks for station rebalancing," in *Proc. of the 2019 IEEE Smart World*, 2019, pp. 686–693.

[208] D. Liang, Z. H. Zhan, and J. Zhang, “An adaptive ant colony system for public bicycle scheduling problem,” in *Proc. of the 11th International Conference on Swarm Intelligence*, 2018, pp. 417-429.

[209] A. Liu, X. Ji, L. Xu, and H. Lu, “Research on the recycling of sharing bikes based on time dynamics series, individual regrets and group efficiency,” *Journal of Cleaner Production*, vol. 208, pp. 666–687, Feb. 2019.

[210] H. Lu, M. Zhang, S. Su, X. Gao, and C. Luo, “Broken bike recycling planning for sharing bikes system,” *IEEE Access*, vol. 7, pp. 177354–177361, Dec. 2019.

[211] D. Zhang, W. Xu, B. Ji, S. Li, and Y. L. Liu, “An adaptive tabu search algorithm embedded with iterated local search and route elimination for the bike repositioning and recycling problem,” *Computers & Operations Research*, vol. 123, pp. 1-17, Dec. 2020.

[212] P. Schimek, “Bike lanes next to on-street parallel parking,” *Accident Analysis & Prevention*, vol. 120, pp. 74–82, Nov. 2018.

[213] J. B. Cicchino, M. L. McCarthy, C. D. Newgard, S. P. Wall, C. J. DiMaggio, P. E. Kulie, B. N. Arnold, and D. S. Zuby, “Not all protected bike lanes are the same: Infrastructure and risk of cyclist collisions and falls leading to emergency department visits in three U.S. cities,” *Accident Analysis & Prevention*, vol. 141, Arc. no. 105490, Jun. 2020.

[214] T. He, J. Bao, S. Ruan, S. Ruan, R. Li, Y. Li, H. He, and Y. Zheng, “Interactive bike lane planning using sharing bikes’ trajectories,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1529–1542, Aug. 2020.

[215] Z. Li, J. Zhang, J. Gan, P. Lu, Z. Gao, W. Kong, “Large-scale trip planning for bike-sharing system,” *Pervasive and Mobile Computing*, vol. 54, pp. 16–28, Mar. 2019.

[216] Z. G. Chen, Z. H. Zhan, S. Kwong, J. Zhang, “Evolutionary computation for intelligent transportation in smart cities: A survey,” *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 83–102, May 2022.

[217] J. Zhang and P. S. Yu, “Trip route planning for bicycle-sharing systems,” in *Proc. of the 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC),* 2017, pp. 381–390.

[218] X. Chen, Y. Ong, M. Lim, and K. C. Tan, “A multi-facet survey on memetic computation,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5,

pp. 591–607, Oct. 2011.

[219] L. Lugo, C. Segura, and G. Miranda, “A diversity-aware memetic algorithm for the linear ordering problem,” *Memetic Computing*, vol. 14, pp. 395–409, Dec. 2022.

[220] A. V. Eremeev and Y. V. Kovalenko, “A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem,” *Memetic Computing*, vol. 12, pp. 23–36, Mar. 2020.

[221] J. Luo, D. Zhou, L. Jiang, and H. Ma, “A particle swarm optimization based multiobjective memetic algorithm for high-dimensional feature selection,” *Memetic Computing*, vol. 14, pp. 77–93, Mar. 2022.

[222] X. Shao, Y. J. Gong, Z. H. Zhan, and J. Zhang, “Bipartite cooperative coevolution for energy-aware coverage path planning of UAVs,” *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 1, pp. 29-42, Feb. 2022.

[223] Y. Guo, J.Y. Li, and Z. H. Zhan, “Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach,” *Cybernetics and Systems*, vol. 52, no. 1, pp. 36–57, Oct. 2020.

[224] S. C. Liu, Z. G. Chen, Z. H. Zhan, S. W. Jeon, S. Kwong, and J. Zhang, “Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach,” *IEEE Transactions on Cybernetics*, vol. 53, no. 3, pp. 1460-1474, Mar. 2023.

[225] Z. H. Zhan, J. Y. Li, and J. Zhang, “Evolutionary deep learning: A survey,” *Neurocomputing*, vol. 483, pp. 42–58, Apr. 2022.

[226] J. Y. Li, Z. H. Zhan, J. Xu, S. Kwong, and J. Zhang, “Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 5, pp. 2338-2352, May 2023.

[227] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, “Adaptive distributed differential evolution,” *IEEE Transactions on Cybernetics*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.

[228] J. Y. Li, K. J. Du, Z. H. Zhan, H. Wang and J. Zhang, “Distributed differential evolution with adaptive resource allocation,” *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 2791-2804, May 2023.

[229] Z. H. Zhan, J. Zhang, Y. Lin, J. Y. Li, T. Huang, X. Q. Guo, F. F. Wei, S. Kwong, X. Y. Zhang, and R. You, "Matrix-based evolutionary computation," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 315–328, Apr. 2022.

[230] J. Y. Li, Z. H. Zhan, R. D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation-level parallelism for evolutionary computation: a pipeline-based parallel particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 10, pp. 4848-4859, Oct. 2021.

[231] X. Zhang, Z. H. Zhan, W. Fang, P. Qian, and J. Zhang, "Multipopulation ant colony system with knowledge-based local searches for multiobjective supply chain configuration," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 512-526, Jun. 2022.

[232] X. F. Liu, Z. H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[233] S. C. Liu, Z. H. Zhan, K. C. Tan, and J. Zhang, "A multiobjective framework for many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13654-13668, Dec. 2022.

[234] J. Y. Li, Z. H. Zhan, K. C. Tan, and J. Zhang, "A meta-knowledge transfer-based differential evolution for multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 719–734, Aug. 2022.

[235] Y. Jiang, Z. H. Zhan, K. C. Tan, and J. Zhang, "A bi-objective knowledge transfer framework for evolutionary many-task optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1514-1528, Oct. 2023.

[236] S. H. Wu, Z. H. Zhan, K. C. Tan, and J. Zhang, "Orthogonal transfer for multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 185-200, Feb. 2023.

[237] K. J. Du, J. Y. Li, H. Wang, and J. Zhang, "Multi-objective multi-criteria evolutionary algorithm for multi-objective multi-task optimization," *Complex Intelligent Systems*, vol. 9, pp. 1211–1228, Apr. 2023.

[238] H. Zhao, Z. H. Zhan, Y. Lin, X. Chen, X. N. Luo, J. Zhang, S. Kwong, and J.

Zhang, “Local binary pattern-based adaptive differential evolution for multimodal optimization problems,” *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3343–3357, Jul. 2020.

[239] Z. G. Chen, Z. H. Zhan, H. Wang, and J. Zhang, “Distributed individuals for multiple peaks: a novel differential evolution for multimodal optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 708-719, Aug. 2020.

[240] Y. Jiang, Z. H. Zhan, K. C. Tan, and J. Zhang, “Optimizing niche center for multimodal optimization problems,” *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2544-2557, Apr. 2023.

[241] Z. J. Wang, Z. H. Zhan, Y. Lin, W. J. Yu, H. Q. Yuan, T. L. Gu, S. Kwong, and J. Zhang, “Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[242] S. H. Wu, Z. H. Zhan, and J. Zhang, “SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 478–491, Jun. 2021.

[243] X. F. Liu, Z. H. Zhan, and J. Zhang, “Resource-aware distributed differential evolution for training expensive neural-network-based controller in power electronic circuit,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6286-6296, Nov. 2022.

[244] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, Aug 2021.

[245] L. Shi, Z. H. Zhan, D. Liang, and J. Zhang, “Memory-based ant colony system approach for multi-source data associated dynamic electric vehicle dispatch optimization,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17491–17505, Oct. 2022.

[246] C. He, Y. Zhang, D. Gong, X. Song, and X. Sun, “A multi-task bee colony band selection algorithm with variable-size clustering for hyperspectral images,” *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1566–1580, Dec. 2022.

[247] X. F. Song, Y. Zhang, Y. N. Guo, X. Y. Sun, and Y. Li. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 882–895, Oct. 2020.

[248] Data Open Platform of Shenzhen Government. https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_00403627.

[249] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, Jun. 2019.

[250] J. Y. Li, Z. H. Zhan, H. Wang, and J. Zhang, "Data-driven evolutionary algorithm with perturbation-based ensemble surrogates," *IEEE Transactions on Cybernetics*, vol. 51, no. 8, pp. 3925–3937, Aug. 2021.

[251] C. A. C. Coello, S. G. Brambila, J. F. Gamboa, and M. G. C. Tapia, "Multi-objective evolutionary algorithms: Past, present, and future," *Springer Optimization and Its Applications*, vol. 170, pp. 137–162, Jan. 2021.

[252] J. Y. Li, X. Y. Deng, Z. H. Zhan, L. Yu, K. C. Tan, K. K. Lai, and J. Zhang "A multipopulation multiobjective ant colony system considering travel and prevention costs for vehicle routing in covid-19-like epidemics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25062-25076, Dec. 2022

[253] J. R. Jian, Z. G. Chen, Z. H. Zhan and J. Zhang, "Region encoding helps evolutionary computation evolve faster: a new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 779-793, Aug. 2021.

[254] J. Y. Li, Z. H. Zhan, K. C. Tan, and J. Zhang, "Dual differential grouping: a more general decomposition method for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 53, no. 6, pp. 3624-3638, Jun. 2023.

[255] X. Zhang, B. W. Ding, X. X. Xu, J. Y. Li, Z. H. Zhan, P. Qian, W. Fang, K. K. Lai, and J. Zhang, "Graph-based deep decomposition for overlapping large-scale optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2374-2386, Apr. 2023.

[256] J. Q. Yang, C. H. Chen, J. Y. Li, D. Liu, T. Li, and Z. H. Zhan, "Compressed-encoding particle swarm optimization with fuzzy learning for large-scale feature

selection," *Symmetry*, vol. 14, no. 6, pp. 1142, Jun. 2022.

[257] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Recent advances and emerging challenges of feature selection in the context of big data," *Knowledge-Based Systems*, vol. 86, pp. 33–45, Sep. 2015.

[258] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16-28, Jan. 2014.

[259] Y. Li, T. Li, and H. Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 551–577, Dec. 2017.

[260] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606-626, Aug. 2016.

[261] Z. H. Zhan, J. Y. Li, S. Kwong, and J. Zhang, "Learning-aided evolution for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1794-1808, Dec. 2023.

[262] Q. Yang, W. N. Chen, T. Gu, H. Jin, W. Mao, and J. Zhang, "An adaptive stochastic dominant learning swarm optimizer for high dimensional optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 3, pp. 1960–1976, Mar. 2022.

[263] B. Tran, B. Xue, and M. Zhang, "A new representation in PSO for discretization-based feature selection," *IEEE Transactions on Cybernetics*, vol. 48, no. 6, pp. 1733–1746, Jun. 2018.

[264] Y. Xue, B. Xue, and M. Zhang, "Self-adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 5, pp. 1-27, Sep. 2019.

[265] Y. Xue, T. Tang, W. Pang, and A. X. Liu, "Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers," *Applied Soft Computing*, vol. 88, pp. 1-12, Mar. 2020.

[266] X. Song, Y. Zhang, D. Gong, and X. Sun, "Feature selection using bare-bones particle swarm optimization with mutual information," *Pattern Recognition*, vol. 112, pp. 1-17, Apr. 2021.

[267] A. D. Li, B. Xue, and M. Zhang, "Improved binary particle swarm optimization for feature selection with new initialization and search space reduction

strategies," *Applied Soft Computing*, vol. 106, arc. no. 107302, Jul. 2021.

[268] X. F. Song, Y. Zhang, D. W. Gong, and X. Z. Gao, "A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9573–9586, Sep. 2022.

[269] J. Q. Yang et al., "Bi-directional feature fixation-based particle swarm optimization for large-scale feature selection," *IEEE Transactions on Big Data*, vol. 9, no. 3, pp. 1004-1017, Jun. 2023.

[270] K. Chen, B. Xue, M. Zhang, and F. Zhou, "Evolutionary multitasking for feature selection in high-dimensional classification via particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 446–460, Jun. 2022.

[271] K. Chen, B. Xue, M. Zhang, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 7172-7186, Jul. 2022.

[272] J. Kennedy, "Bare bones particle swarms," in *Proc. of the 2003 IEEE Swarm Intelligence Symposium*, Apr. 2003, pp. 80–87.

[273] X. Zheng, A. K. Qin, M. Gong, and D. Zhou, "Self-regulated evolutionary multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 16–28, Feb. 2020.

[274] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Transactions on Cybernetics*, vol. 51, no. 5, pp. 2563–2576, May 2021.

[275] S. Yao, Z. Dong, X. Wang, and L. Ren, "A multiobjective multifactorial optimization algorithm based on decomposition and dynamic resource allocation strategy," *Information Sciences*, vol. 511, pp. 18–35, Feb. 2020.

[276] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1-45, Dec. 2017.

[277] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, arc.no. 106839, Mar. 2020.

[278] B. Singh, N. Kushwaha, and O. P. Vyas, "A feature subset selection technique

for high dimensional data using symmetric uncertainty," *Journal of Data Analysis and Information Processing*, vol. 2, Art. no. 4, pp. 95-105, Nov. 2014.

[279] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 1997, vol. 5, pp. 4104–4108.

[280] F. Wilcoxon, "Individual Comparisons by Ranking Methods," in Breakthroughs in Statistics, S. Kotz and N.L. Johnson (eds.), Springer Series in Statistics, Springer, New York, NY, 1992, doi: 10.1007/978-1-4612-4380-9_16.

[281] P. Somol, P. Vácha, S. Mikeš, J. Hora, P. Pudil, and P. Zid, "Introduction to feature selection toolbox 3–the c++ library for subset search, data modeling and classification," *Academy of Sciences of the Czech Republic Institute of Information Theory and Automation (UTIA)*, Technical Report No. 2287, Oct. 2010.