Parallel memetic algorithms for solving large scale combinatorial optimization problems

Tang, Jing

2008

Tang, J. (2008). Parallel memetic algorithms for solving large scale combinatorial optimization problems. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3496>

<https://doi.org/10.32657/10356/3496>

Nanyang Technological University

Downloaded on 26 Apr 2023 10:50:28 SGT

Parallel Memetic Algorithms for Solving Large Scale Combinatorial Optimization Problems

Tang Jing

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

To my family, for all your love.

Acknowledgements

I sincerely thank my supervisor, Associate Prof. Lim Meng Hiot, for his inspiring constant support, invaluable guidance, encouragements, and easy-going communication throughout my Ph.D. study. I really appreciate his style of supervision and his patience to his students. His concerns not only on the research work, but also on my life, which make the relationship more like friends. His supervision, inside and outside of academics, is worth learning in my whole life.

I would also like to express my gratitude to Assistant Prof. Ong Yew Soon for his encouragements, guidance, concerns and much help in numerous ways, acting as a co-supervisor professionally and a friend.

I also thank the Associate Prof. Er Meng Joo, Mr. Leong Yew Kong and other staffs of Intelligent Systems Centre (IntelliSys) for their support and help. And I also wish to convey my appreciation to Mr. Sia Liang Poo, Mr. William Seow Yang Hing, and Mr. Jimmy Goh Mia Yong and other EEE staffs for their kind and friendly assistance.

I would like to thank my seniors and other friends during my Ph.D. study for their unhesitating help and long-standing friendship.

Most importantly, special thanks to my parents and my husband for their invaluable support and constant love on me all the time.

I dedicate this thesis to all of you.

Abstract

Although significant development of heuristics for various combinatorial optimization problems has been achieved, the extremely expensive computational overhead to tackle large scale problems is still a challenge in its own right. The solution of these problems requires both improvement in mathematical programming algorithms and the utilization of powerful computational platforms. The studies in this dissertation concentrated on memetic algorithms (MAs), a combination of efficient neighborhood search strategies and evolutionary algorithms. Nowadays, distributed computing is an up-and-coming computing technology which has caught much attention of researchers. Based on the intrinsic parallelism of evolutionary algorithms (EAs), in this dissertation, an island model parallel memetic algorithm (PMA) was developed which combines evolutionary algorithms, local search, parallel and distributed computing technologies for large scale combinatorial optimization, such as quadratic assignment problems (QAPs). The experimental results show that substantial speedup can be achieved, especially for the larger size QAPs, in particular for $n \geq 100$. The impact of certain migration parameters, such as migration topology, on the performance of PMA is also analyzed.

In order to strike an optimum balance between the level of exploration provided by the EA, against the level of exploitation posed by the local search procedure throughout the

memetic search, two diversity-adaptive strategies, i.e., 1) diversity-based static adaptive strategy (PMA-SLS) and 2) diversity-based dynamic adaptive strategy (PMA-DLS) are proposed for controlling the local search frequency during the PMA search. Empirical study on large scale QAPs shows that the diversity-adaptive PMA converges to competitive solutions at significantly lower computational cost when compared to the canonical MA and PMA. Furthermore, it is found that the diversity-based dynamic adaptation strategy displays better robustness in terms of solution quality across the class of QAP problems considered. Static adaptation strategy on the other hand requires extra effort in selecting suitable parameters to suit the problems in hand.

To deal with the heterogeneity of the computing resources, a hierarchical model PMA (hPMA-DLS) is proposed to provide the speedup regardless of the heterogeneity in the distributed environment while preserving the standard behavior of the PMA. The empirical study on several large scale QAPs shows that hPMA-DLS can enhance the efficiency of the island model PMA-DLS search without deterioration in the solution quality.

Table of Contents

Abstract.....	i
Table of Contents	iii
List of Acronyms	viii
List of Tables	x
List of Figures.....	xii
Chapter 1 Introduction.....	1
1.1 Combinatorial Optimization Problems	1
1.2 Memetic Algorithm.....	3
1.3 Parallel Algorithm Methodology	4
1.4 Computational Platform.....	6
1.5 Motivation.....	7
1.6 Objectives	11
1.7 Contributions	13

1.8	Organization of the Thesis	15
Chapter 2	Combinatorial Optimization and Metaheuristics.....	17
2.1	Combinatorial Optimization	17
2.1.1	Definitions	17
2.1.2	Computational Complexity.....	18
2.1.3	Solution Methods.....	20
2.1.4	Quadratic Assignment Problems.....	21
2.2	Metaheuristics.....	25
2.2.1	Trajectory Methods.....	26
2.2.1.1	Basic Local Search.....	27
2.2.1.2	Simulated Annealing	28
2.2.1.3	Tabu Search.....	29
2.2.1.4	Greedy Randomized Adaptive Search Procedures.....	30
2.2.1.5	Variable Neighborhood Search	32
2.2.1.6	Guided Local Search.....	32
2.2.1.7	Iterated Local Search	33
2.2.2	Population-based Methods.....	33
2.2.2.1	Ant Colony Optimization.....	34
2.2.2.2	Evolutionary Computation	35
2.2.2.3	Parallel Evolutionary Algorithm	38
2.2.3	Towards Memetic Algorithms	41
2.2.3.1	Design Issues for Memetic Algorithm	43

2.2.3.2	Memetic Algorithm and Multi-Objective Optimization	51
2.2.3.3	Parallel Paradigm of Memetic Algorithms.....	53
2.2.3.4	Theoretical Study	54
2.2.3.5	Applications of MAs	55
2.2.3.6	Other Research on Memetic Algorithms.....	56
2.3	Summary.....	57
Chapter 3 Island Model Parallel Memetic Algorithm (PMA) with Fixed Local Search.....		59
3.1	Introduction.....	59
3.2	Sequential Memetic Algorithm for QAPs.....	61
3.3	Island Model Parallel Memetic Algorithm	64
3.3.1	Operators for Single Island of PMA	65
3.3.1.1	Fitness Evaluation, Scaling and Selection	65
3.3.1.2	Crossover, Mutation and Elitism.....	67
3.3.1.3	k -gene Exchange Local Search	69
3.3.2	Inter-islands relationships of multi-island PMA.....	73
3.3.3	Parallel Implementation.....	75
3.4	Experimental Results of PMA-FLS for QAPs.....	77
3.4.1	Simulation Setup and Performance Measure Criteria	78
3.4.2	Comparison of SMA-FLS and Two-island PMA-FLS	81
3.4.3	Analysis on Solution Quality.....	87
3.4.4	Analysis on Computational Cost	88

3.4.5	Scalability Study of Multi-island PMA-FLS	90
3.5	Study of Migration Topology of PMA-FLS	97
3.5.1	Migration Topology	97
3.5.2	Empirical Study	100
3.5.2.1	Analysis on Empirical Results	102
3.5.2.2	Analysis on Scalability.....	103
3.6	Summary.....	105
Chapter 4	Diversity-based Static Adaptive Strategy	107
4.1	Introduction.....	107
4.2	Canonical Island Model PMA	110
4.3	Diversity-based Static Adaptive Strategy (PMA-SLS).....	113
4.4	Empirical Study	115
4.4.1	Results Comparison — PMA-SLS vs. PMA	117
4.4.2	Analysis of Results	121
4.5	Summary.....	127
Chapter 5	Diversity-based Dynamic Adaptive Strategy	128
5.1	Introduction.....	128
5.2	PMA-DLS — Parallel Memetic Algorithm with Diversity-based Dynamic Adaptive Strategy	130
5.3	Empirical Study	132
5.3.1	Results Comparison — PMA-DLS vs. PMA-SLS	133
5.3.2	Results Comparison — PMA-DLS, PMA-SLS and PMA	135

5.3.3	Overall Comparison of Results and Analysis	137
5.3.4	Comparison with Other Results.....	145
5.3.5	Discussion on the Advantages of PMA-DLS	147
5.4	Summary.....	148
Chapter 6 Hierarchical Model PMA in Heterogeneous Computing Environment....		
		150
6.1	Introduction.....	150
6.2	Hierarchical Model (hPMA-DLS) in Heterogeneous Computing Environment....	
		152
6.3	Empirical Study	160
6.4	Summary.....	163
Chapter 7 Conclusions and Future Work.....		165
7.1	Conclusions.....	165
7.2	Future Work	169
Chapter 8 Author's Publications.....		172
8.1	Journals.....	172
8.2	Conferences	172
Bibliography		175

List of Acronyms

ACO – Ant colony optimization

CO – Combinatorial Optimization

EA – Evolutionary Algorithm

EDA – Estimation of distribution algorithms

GAs – Genetic algorithms

GLS – Guided local search

GRASP – The greedy randomized adaptive search procedures

hPMA-DLS – Hierarchical model of PMA-DLS

ILS – Iterated local search

LS – Local search

MA – Memetic algorithm

PEA – Parallel evolutionary algorithm

PMA – Parallel memetic algorithm with complete local search strategy

- PMA-DLS – Island model parallel memetic algorithm with diversity-based dynamic adaptive strategy
- PMA-DLSa – Island model parallel memetic algorithm with diversity-based dynamic adaptive strategy running in heterogeneous computing environment
- PMA-DLSb – Island model parallel memetic algorithm with diversity-based dynamic adaptive strategy running in homogeneous computing environment
- PMA-FLS – Island model parallel memetic algorithm with fixed local search strategy
- PMA-FLS-rand – PMA-FLS with random topology
- PMA-FLS-ring – PMA-FLS with one-way ring topology
- PMA-SLS – Island model parallel memetic algorithm with diversity-based static adaptive strategy
- QAP – Quadratic assignment problem
- SA – Simulated annealing
- SMA – Sequential memetic algorithm
- SMA-FLS – Sequential memetic algorithm with fixed local search strategy
- TS – Tabu search
- TSP – Traveling salesman problem
- VNS – Variable neighborhood search

List of Tables

Table 3.1 Parameters setting for SMA-FLS and two-island PMA-FLS	81
Table 3.2 Testing results of <i>sko</i> benchmarks	84
Table 3.3 Testing results of <i>lipa</i> benchmarks	85
Table 3.4 Testing results of <i>tai</i> benchmarks	86
Table 3.5 Testing results of <i>will100</i> and <i>tho150</i> benchmarks	86
Table 3.6 Comparison of PMA-FLS with MF1997	88
Table 3.7 PMA parameters setting	100
Table 3.8 Testing results of <i>sko100b</i> benchmark	101
Table 3.9 Testing results of <i>tai100b</i> benchmark	101
Table 3.10 Testing results of <i>tho150</i> benchmark	102
Table 4.1 Comparison between PMA-SLS and PMA	117
Table 4.2 Results of testing on <i>sko100b</i> benchmark	119
Table 4.3 Results of testing on <i>tai100b</i> benchmark	119

Table 4.4 Results of testing on <i>sko100*</i> benchmarks	120
Table 4.5 Results of testing on <i>tai100a</i> benchmark.....	120
Table 4.6 Results of testing on <i>wil100</i> benchmark	120
Table 4.7 Results of testing on <i>tho150</i> benchmark.....	121
Table 4.8 The mean and the one tail difference of <i>t</i> -test for statistical significance of <i>sko100b</i> between PMA-SLS and PMA	123
Table 4.9 Comparison of PMA-SLS with MF1999b and MF2000b	126
Table 5.1 Comparison of PMA-DLS and PMA-SLS with γ_1 , γ_2 and γ_3	134
Table 5.2 Comparison among PMA-SLS, PMA-DLS and PMA	135
Table 5.3 Results of testing on <i>sko100b</i> benchmark.....	137
Table 5.4 Results of testing on <i>tai100b</i> benchmark.....	138
Table 5.5 Results of testing on <i>sko100*</i> benchmarks	138
Table 5.6 Results of testing on <i>tai100a</i> benchmark.....	139
Table 5.7 Results of testing on <i>wil100</i> benchmark	140
Table 5.8 Results of testing on <i>tho150</i> benchmark.....	140
Table 5.9 Mean and the one tail difference of <i>t</i> -test between PMA-DLS and PMA	143
Table 5.10 Comparison of PMA-DLS, PMA-SLS with MF1999b, MF2000b, ATO2000, and ZSTF2004	147
Table 6.1 Configurations of heterogeneous computing nodes.....	153
Table 6.2 Comparison between PMA-DLSa and PMA-DLSb.....	155
Table 6.3 Comparison between hPMA-DLS and PMA-DLSa.....	161

List of Figures

Figure 2.1 Local Search.....	27
Figure 2.2 Basic structure of EC algorithm.....	37
Figure 2.3 Canonical Memetic Algorithm.....	44
Figure 3.1 Flowchart of sequential memetic algorithm.....	62
Figure 3.2 Crossover implemented in SMA-FLS.....	68
Figure 3.3 2-gene exchange procedure.....	70
Figure 3.4 Pseudo codes of local search procedure using 2-gene exchange operation	71
Figure 3.5 Algorithm flow chart for each subpopulation of multi-island model.....	65
Figure 3.6 Two-island PMA Model	74
Figure 3.7 Inter-islands relationships of multi-island PMA model	75
Figure 3.8 Execution mechanism for PMA implementation on NetSolve infrastructure.	77
Figure 3.9 Ratio of communication to computing time and percentage of communication overhead for different <i>sko</i> benchmarks in two-island PMA-FLS model.....	90

Figure 3.10 <i>sko100b</i> benchmark.....	92
Figure 3.11 <i>tai100b</i> benchmark.....	92
Figure 3.12 <i>tho150</i> benchmark.....	93
Figure 3.13 Speedup of <i>sko100b</i> benchmark.....	95
Figure 3.14 Speedup of <i>tai100b</i> benchmark.....	96
Figure 3.15 Speedup of <i>tho150</i> benchmark.....	96
Figure 3.16 One-way ring topology.....	99
Figure 3.17 Total CPU time of <i>sko100b</i> benchmark	104
Figure 3.18 Total CPU time of <i>tai100b</i> benchmark.....	104
Figure 3.19 Total CPU time of <i>tho150</i> benchmark.....	104
Figure 4.1 Pseudo-code of the canonical island model parallel memetic algorithm.....	110
Figure 4.2 Entropy measure for PMA and PGA on the <i>sko100b</i> QAP problem.....	113
Figure 4.3 Pseudo-code of PMA-SLS	114
Figure 4.4 Comparison of entropy between PMA-SLS and PMA	117
Figure 4.5 Comparison between PMA-SLS, PMA and PMA-FLS on <i>sko100b</i> benchmark	122
Figure 4.6 Comparison between PMA-SLS and PMA-FLS on <i>tai100b</i> benchmark	124
Figure 5.1 Pseudo-code of PMA-DLS.....	131
Figure 5.2 Application of Gaussian functions to determine number of chromosomes selected for local search according to PMA-SLS	134
Figure 5.3 Comparison of diversity among PMA-SLS, PMA-DLS and PMA	136
Figure 5.4 Comparison among PMA-SLS, PMA-DLS, PMA and PMA-FLS on <i>sko100b</i>	

benchmark.....	141
Figure 5.5 Two-dimensional plot of MA, PMA, PMA-DLS, PMA-SLS and PMA-FLS on <i>sko100b</i> benchmark	143
Figure 5.6 Comparison among PMA-DLS, PMA-SLS and PMA-FLS on <i>tai100b</i> benchmark.....	145
Figure 6.1 Comparison between PMA-DLSa and PMA-DLSb on <i>sko100b</i> benchmark	156
Figure 6.2 The hierarchical model (hPMA-DLS).....	157
Figure 6.3 The size of ensembles of individuals for hPMA-DLS on <i>sko100b</i> benchmark	162
Figure 6.4 Comparison between hPMA-DLS and PMA-DLSa on <i>sko100b</i> benchmark	163

Introduction

1.1 Combinatorial Optimization Problems

Combinatorial optimization (Lai 1995) is an important area of research and development. Unlike continuous variable optimization, combinatorial optimization deals with finding a good or possibly the best solution out of a large but finite number of possibilities, provided that a clear notion of solution quality exists. Finding the factory production schedule with the highest throughput, designing the most reliable telecommunications network, or finding the structure of a protein molecule in a three dimensional space that minimizes potential energy are just few examples of combinatorial optimization problems. Many others have been reported in the fields of management science, industrial engineering, computer science, biology, and physics. Many of these problems are very complex and thus hard to solve; general mathematical methods are usually not available. Combinatorial optimization problems constitute a class of problems with practical significance and finding a solution is usually not trivial, from a computational point of view.

In contrast to continuous optimization problems, combinatorial problems have a finite but large number of candidate solutions. Therefore, an obvious way to solve these problems is to enumerate all candidate solutions by comparing them against each other. Unfortunately, for most interesting combinatorial optimization problems, this approach proves to be impractical since the number of candidate solutions typically grows exponentially with the problem size. However, in recent years great progress has been made in solving these problems with *exact* algorithms. But the issue of computational complexity still remains. For most cases, the alternative is to tackle the problems using *heuristic* search, with no guarantee of finding the optimum solution. Consequently, there has been significant effort in developing heuristics aimed at finding high quality solutions in short time. In the 1970s, a new approximate algorithm emerged, combining basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. These methods are commonly referred to as *metaheuristics*. Metaheuristics are widely acknowledged as essential tools to address difficult problems in numerous and diverse fields. In fact, quite often, metaheuristics offer the only practical approach to solve complex problems of realistic scale.

In the field of combinatorial optimization, the quadratic assignment problem (QAP) (Koopmans and Beckmann 1957) is among the hardest. Since it can be applied to many interesting practical applications, the QAP has been a subject of research focus and has been investigated for years. Because QAPs are NP-hard, only implicit enumeration approaches are known to solve them to optimality. However, the computational time cost

increases exponentially with respect to the size of the problem and thus become unmanageable. Generally, it is considered unsolvable for QAP with size 20 or more by exhaustive methods. But in practice, many applications are significantly larger than 20. Alternatively, heuristic approaches have played an important role in algorithms capable of providing good solutions in reasonable time. Although significant development of heuristics for QAP has been achieved, the extremely expensive computational overhead to tackle large scale QAPs is still a challenge in its own right. The solution of these problems requires both improvement in mathematical programming algorithms and the utilization of powerful computational platforms.

1.2 Memetic Algorithm

Evolutionary algorithms (EAs) are a class of search and optimization techniques that work on a principle inspired by nature — Darwinian Evolution. The concept of natural selection is captured in EAs. Specifically, solutions to a given problem are coded as so-called chromosomes. The evolution of chromosomes due to the action of crossover, mutation, and natural selection is simulated through procedures defined as computer codes. Nevertheless, canonical EAs generally suffer from excessively slow convergence in locating a precise enough solution because of their failure to exploit local information. This often limits the practicality of EAs on many large-scale real world problems where computational time is a crucial consideration. Hence, it is now well established that pure

EAs are not well suited to fine tuning search in complex combinatorial spaces and that hybridization with other techniques can greatly improve the efficiency of search. The combination of EAs with local search (LS) was named “memetic algorithms” (MAs) in (Moscato 1989). MAs are extensions of EAs that apply separate processes to refine individuals, for example, improving their fitness by hillclimbing. These methods are inspired by models of adaptation in natural systems that combine the evolutionary adaptation of a population with individual learning within the lifetime of its members.

From an optimization point of view, MAs have been shown to be both more efficient (i.e., requiring orders of magnitude fewer evaluations to find optima) and more effective (i.e., identifying higher quality solutions) than traditional EAs for some problem domains. As a result, MAs are gaining wide acceptance, in particular, in well-known combinatorial optimization problems where large instances have been solved to optimality and where other metaheuristics have failed to produce comparable results (Merz and Freisleben 1999b).

1.3 Parallel Algorithm Methodology

Although the use of metaheuristics allows for significant reduction in the temporal complexity of the search process, the latter remains time-consuming for large scale combinatorial optimization problems. Therefore, parallelism is necessary to not only

reduce the resolution time but also improve the quality of the provided solutions (Resende *et al.* 2001). Parallel metaheuristics aim to address both issues. Of course, the first goal is to solve larger problem instances in reasonable computing time. In appropriate modeling and settings, parallel meta-heuristics also prove to produce solutions of much higher quality than sequential versions. Fortunately, many classes of metaheuristics are inherently parallelizable and this leads researchers to develop parallelization techniques and efficient implementations.

Parallel EAs are naturally prone to parallelism since most variation operators can be easily undertaken in parallel (Alba and Tomassini 2002). Much research has been done on parallel EA models (Cantu-Paz 1998, Alba and Tomassini 2002). Till now, there are four main types of parallel EAs (PEAs) (1) Master-slave, (2) Fine-grained, (3) Multi-population, and (4) Hierarchical.

It is important that the intrinsic parallelism of EAs is retained when designing any MAs. Best of all, parallel EAs possess diversity preservation capabilities that alleviate the effect of premature convergence. Parallel Memetic Algorithms (PMAs) are a class of modern parallel meta-heuristics that combine evolutionary algorithms, local search, parallel and distributed computing technologies for global optimization.

1.4 Computational Platform

It is noted that computing technology will allow for further advancement in heuristic algorithms by making available computational horsepower unseen before in the past. The performance of parallel algorithms is desirable, but they have often been implemented on high-performance computers. Although traditional parallel processing (super) computers continue to become more and more powerful, these resources have inherent limitations. They are expensive and usually not readily available. An alternative platform for massively distributed computation is based on the notion of grid computing (Foster and Kesselman 1999), also referred to as metacomputing (Catlett and Smarr 1992). Nowadays, grid computing is an up-and-coming computing technology which has caught much attention of researchers with its novel approach of organizing and accessing computational resources. It can also be perceived as a novel concept rather than a technology. Many applications have been explored to capitalize on this new wave of computing paradigm (Foster and Kesselman 1999).

Grid computing can be differentiated from almost all distributed computing paradigms by its defining characteristic: The essence of grid computing lies in the efficient and optimal utilization of a wide range of heterogeneous, loosely coupled resources in an organization tied to sophisticated workload management capabilities or information virtualization. It is defined as a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. A

computational grid consists of a potentially large number of geographically dispersed CPUs linked by a communication medium such as the Internet. The main advantage of such a platform compared to a traditional multiprocessor machine is that a large number of CPUs may be assembled very inexpensively.

1.5 Motivation

Various metaheuristics have been designed to solve large scale combinatorial optimization problems effectively and efficiently. In this thesis, we focus on memetic algorithm — a combination of evolutionary algorithms and local search strategies. It has stimulated great interest in recent years for its flexibility and effectiveness.

Evolutionary algorithms take inspiration from biological evolution. They involve a search from a “population” of solutions, making use of competitive selection, recombination and mutation operators to generate new solutions which are biased towards better regions of the search space.

On the other hand, local search performs iterative search for optimum solution in the neighborhood of a candidate solution. Starting from a single candidate solution, local search methods use “local” information to decide the next neighbor they should move to. As the name suggests, all local search methods suffer from the same drawback as they tend to get stuck in local optima.

In order to locate the global optimum, there are two competing goals, exploration and exploitation. Exploration is important to ensure global reliability, i.e., every part of the domain is searched enough to provide a reliable estimate of the global optimum. Exploitation is also important since it concentrates the search effort around the best solutions found so far by searching their neighborhoods to produce better solutions. MAs achieve these two goals using a combination of global searches and local searches. When correctly implemented, MA would do no worse than evolutionary algorithm or local search alone (Davis 1991).

Parallelism is considered a desired feature of any framework for optimization of computationally expensive problems. Considering the expensive computational cost incurred by many combinatorial optimization problems, especially the large scale instances of certain combinatorial optimization problems, one spontaneous outcome is to develop corresponding parallel algorithms to improve the efficiency and effectiveness of the primary serial algorithm. On the other hand, with the development of advanced computing technology such as distributed computing, there is great potential to develop powerful solution methodology for many applications. Due to the intrinsic parallelism of EA, there is great motivation to develop our novel parallel MA model in a distributed computing framework. The approach of our model is based on the island model (Cantu-Paz 1998, Cantu-Paz 2000a, Alba and Tomassini 2002) whereby islands of subpopulations are farmed to different processing nodes for execution. Furthermore, the proposed model will be efficiently parallelized on distributed computing architecture

because of the parallel characteristic offered by the island model. Embedded in a distributed environment, the proposed new model is expected to treat the complex combinational optimization problem more efficiently and effectively.

In the island model parallel MA, four control parameters pertaining to the migration operation, which have a significant effect on performance are migration interval, migration rate, migration policy and migration topology. The decision on the appropriate parameters has significant impact on the performance of corresponding algorithms. In this work, the impact of migration topology of the island model on the performance is empirically investigated when solving large scale QAPs. Two different topologies are compared, ring topology and random topology in terms of both solution quality and computational time.

It is worth noting that a crucial aspect of MAs or PMAs is to strike an optimum balance between the level of exploration provided by the GA, against the level of exploitation posed by the local search procedure throughout the memetic search. However, in canonical MAs or PMAs, it is common practice for the local search procedure to be applied on every individual/chromosome in the GA population(s). Typically, the extent of the application of local searches in canonical memetic algorithm is based on the principle of “more is better”. In the same spirit, the island model parallel memetic algorithm (PMA) is an important extension of the canonical memetic algorithm which applies local searches to every transitional solutions being considered. This is a very computationally intensive and inefficient search process. At the same time, exhaustive

local search may lead to ineffective search due to premature fall in diversity during the PMA search.

To control the local search frequency during a PMA search, we proposed two diversity-adaptive strategies, i.e., PMA-SLS and PMA-DLS. In contrast to canonical MAs and PMAs, the diversity-based adaptive approaches control the number of individuals undergoing the local search procedure throughout the PMA evolutionary search process. PMA-SLS uses a static adaptation strategy, maintaining population diversity throughout the PMA search by using a pre-defined Gaussian distribution to adjust the local search frequency. PMA-DLS's adaptation is based on online monitoring of population diversity during the PMA search for controlling the local search frequency. The proposed adaptive strategies on the local search frequency are expected to manage a more desirable diversity profile as the search progresses. The proposed adaptive approaches are expected to avoid premature convergence resulting from fast decreasing population diversity, as well as to reduce the computational effort for solving large scale combinatorial optimization problems.

Furthermore, to deal with the heterogeneity of the computing resources, a simple solution to maintain the benefit of parallelization of our proposed island model PMA is to configure the subpopulation size according to the computational capabilities of each processing node. Nevertheless, such an approach might alter the standard behavior of a synchronous island PMA. Since the search behavior could be unpredictable in such approach, it is advisable to be more conservative by maintaining uniform subpopulation

sizes. A preferable solution should still provide the speedup regardless of the heterogeneity in the distributed environment while preserving the standard behavior of the PMA. To achieve such a solution, based on the island model PMA, we proposed a hierarchical model PMA to suit the heterogeneous computing environment. The core idea is to proportionally distribute the most time-consuming task, local search, to computing processors according to their processing capability on the low level. And the high level is sequential implementation of island model PMA. Consequently, we can maintain the uniformity of the subpopulation size at the master side and allocate non-uniform ensembles of chromosomes for local search on the computing processors.

1.6 Objectives

Based on the motivation mentioned above, the primary goal is to develop an effective and efficient parallel memetic algorithmic model to solve large scale combinatorial optimization problems implemented on distributed computing infrastructure. The following objectives are our milestones in achieving this goal.

Firstly, an island model parallel memetic algorithm (PMA) is proposed to improve the effectiveness and efficiency to solve quadratic assignment problems as one case of combinatorial optimization problem, especially the large scale instances of QAP. The proposed parallel memetic algorithm (PMA) combines evolutionary algorithms, local

search, parallel and distributed computing technologies in one entity. The approach of our PMA is based on the island model whereby islands of subpopulations are farmed to different processing nodes for execution. Furthermore, the proposed model will be efficiently parallelized on distributed computing architecture because of the parallel characteristic offered by the island model. The PMA was applied to solve the quadratic assignment problems (QAP) to demonstrate the potential effectiveness of the model. In particular, we will focus on QAP benchmarks of high complexity for n ranging from 60 to 256.

Secondly, we will focus on some different control parameters and strategies used in island model PMA, especially the various migration topologies. Empirical studies were deployed to discover the impact of different migration topologies on the performance of PMA.

Thirdly, to strike an optimum balance between the level of exploration provided by the GA, against the level of exploitation posed by the local search procedure throughout the memetic search, the diversity-based static adaptive strategy (PMA-SLS) is proposed for controlling the local search frequency during the PMA search. PMA-SLS uses a static adaptation strategy, maintaining population diversity throughout the PMA search by using a pre-defined Gaussian distribution to adjust the local search frequency.

Fourthly, based on the online entropy measure of PMA, a diversity-based dynamic adaptive strategy (PMA-DLS) is proposed. PMA-DLS's adaptation is based on online monitoring of population diversity during the PMA search for adaptively controlling the

local search frequency.

Finally, to deal with the heterogeneity of the computing resources, a hierarchical model PMA is proposed to provide the speedup regardless of the heterogeneity in the distributed environment while preserving the standard behavior of the PMA.

1.7 Contributions

The outcome of the work has resulted in several significant contributions.

Firstly, the proposed island model parallel memetic algorithm (PMA) combines evolutionary algorithms, local search, parallel and distributed computing technologies for global optimization which is an important extension of evolutionary algorithm and memetic algorithm. Embedded in the distributed computing framework, a large number of CPUs may be assembled very inexpensively to implement the proposed PMA.

Secondly, investigation on certain control parameters of island model PMA, in particular, the migration topology, sheds light on the impact of various migration topologies on the performance of island model PMA.

Thirdly, the proposed diversity-based static adaptive strategy (PMA-SLS) controls the local search frequency during the island model PMA search, striking an optimum balance between the level of exploration provided by the GA, against the level of exploitation

posed by the local search procedure throughout the memetic search. We demonstrate PMA-SLS's utility in solving complex combinatorial optimization problems, in particular large-scale quadratic assignment problems (QAPs). Based on the empirical results, the PMA-SLS compared to the PMA with complete local search strategy, can reduce the computational time significantly with little or no loss of solution quality. This is due mainly to the ability of the PMA-SLS to manage a more desirable diversity profile as the search progresses.

Fourthly, the proposed diversity-based dynamic adaptive local search strategy (PMA-DLS) is employed, such that the local search frequency is adaptively adjusted based on the online fluctuation of population diversity. This diversity-adaptive approach avoids premature convergence resulting from fast decreasing population diversity, as well as reduces the computational effort. Empirical study on large-scale quadratic assignment problems (QAPs) shows that the diversity-adaptive PMA converges to competitive solutions at significantly lower computational cost when compared to the canonical MA and PMA. Furthermore, it is found that the diversity-based dynamic adaptation strategy displays better robustness in terms of solution quality across the class of QAP problems considered.

Finally, the proposed hierarchical model PMA (hPMA-DLS) provides the speedup regardless of the heterogeneity in the distributed computing environment while preserving the standard behavior of the PMA. The empirical study on several large scale QAPs shows that hPMA-DLS can enhance the efficiency of the island model PMA-DLS

search without deterioration in the solution quality.

It is emphasized that a central theme of the work is to develop a novel island model parallel memetic algorithm for large scale combinatorial optimization problems embedded in a distributed computing platform, as well as addressing certain important issues related to the island model and the balance of exploration and exploitation of memetic algorithms. It is perceived as a major contribution in the field of parallel memetic algorithm and combinatorial optimization research.

1.8 Organization of the Thesis

This chapter has given a concise introduction to some basic concepts in combinatorial optimization problems, memetic algorithm, parallel algorithm methodology, and distributed computational platform which are the major work related to the topics addressed in this dissertation. The motivation, objectives and contributions of the work have also been presented.

The rest of this thesis is organized as follows.

In Chapter 2, a brief overview of combinatorial optimization problems is provided, followed by description of modern metaheuristic techniques with emphasis on evolutionary computations and extended parallel paradigm, neighbourhood search as well as their combination, memetic algorithms.

Chapter 3 focuses on the proposed island model parallel memetic algorithm (PMA) embedded on a distributed computing platform in detail. Empirical studies of PMA utilized for large scale quadratic assignment problems are presented. The study on the migration topology of the island model is also presented in this chapter.

Chapter 4 describes the proposed diversity-based static adaptive strategy for controlling the local search frequency of island model PMA (PMA-SLS). Empirical studies of PMA-SLS on large scale quadratic assignment problems are presented and compared with other algorithms.

Chapter 5 proposes diversity-based dynamic adaptive strategy for controlling the local search frequency of island model PMA (PMA-DLS). Empirical studies of PMA-DLS on large scale quadratic assignment problems are presented and compared with PMA-SLS and other algorithms.

Chapter 6 focuses on the proposed hierarchical model of PMA to suit the heterogeneous characteristics of distributed computing environment.

Chapter 7 summarizes this dissertation by discussing the contributions and limitations of the whole work. It also suggests some possible directions for future research.

Combinatorial Optimization and Metaheuristics

In this chapter, we give a short review of basic concepts on combinatorial optimization, especially quadratic assignment problems (QAPs), computational complexity, and the solution techniques used to solve hard problems, with emphasis on metaheuristics.

1.1 Combinatorial Optimization

Definitions

Many problems with important practical applications are concerned with the selection of a “best” configuration or a set of parameters to achieve some objective criteria. Such problems are generally referred to as optimization problems. If the entities to be optimized are discrete, the number of feasible solutions is finite. We call such problems *combinatorial optimization problems*.

According to (Garey and Johnson 1979), a combinatorial optimization problem P is either a minimization problem or a maximization problem, and it consists of

- (i) a set D_p of instances,
- (ii) a finite set $S_p(I)$ of candidate solutions for each instance $I \in D_p$, and
- (iii) a function M_p that assigns a positive rational number $M_p(I, x)$ called the solution value for x to each instance $I \in D_p$ and each candidate solution $x \in S_p(I)$.

Thus, an optimal solution for an instance $I \in D_p$ is a candidate solution $x^* \in S_p(I)$ such that, for all $x \in S_p(I)$, $M_p(I, x^*) \leq M_p(I, x)$ if P is a minimization problem, and $M_p(I, x^*) \geq M_p(I, x)$ if P is a maximization problem.

Based on the fact that the set of candidate solutions is finite, *exhaustive search* can simply evaluate and compares $M_p(I, x)$ for all $x \in S_p(I)$. However, the search space of many combinatorial problems grows exponentially with the problem size, i.e., the number of components in a solution vector x . Thus, this complete enumeration scheme becomes impractical. For a large class of combinatorial optimization problems no alternative algorithms running in polynomial time are known. This phenomenon has led to the development of complexity theory (Garey and Johnson 1979), and in particular, to the theory of *NP*-completeness.

Computational Complexity

The theory of *NP*-completeness is focused on decision problems such as:

Is there a feasible solution $x \in S_p(I)$ such that $M_p(I, x) \leq L$ ($M_p(I, x) \geq L$)?

Two basic classes of decision problems are distinguished: the class P of decision problems that can be solved by a polynomial-time algorithm, and the class NP of decision problems that can be solved by a non-deterministic polynomial-time algorithm. The latter consists of two stages. In the first stage, a solution to a given instance I is guessed. In the second stage, this solution is checked by a deterministic polynomial verification algorithm (Garey and Johnson 1979).

Given these two classes, NP -complete problems can be defined (Garey and Johnson 1979, Johnson and Papadimitriou 1985):

Def. 2.1 *A decision problem p is NP -complete, if (a) $p \in NP$, and (b) all problems in NP can be reduced to p by a polynomial-time algorithm.*

Probably the most important open question in computer science is whether $P = NP$. From the above definition, it follows immediately that if for one problem in NP a polynomial time algorithm can be found, all problems in NP can be solved in polynomial time and thus $P = NP$. However, it is commonly believed that $P \neq NP$, but no proof has been found yet.

Optimization problems cannot be NP -complete, since they are not decision problems, even though for each optimization problem a decision problem can be defined which is equivalent in complexity. However, the notion of NP -hard problems is less restricted than the definition of NP -completeness (Johnson and Papadimitriou 1985):

Def. 2.2 *A problem (decision or otherwise) is NP -hard if all problems in NP are*

polynomially reducible to it.

This definition includes decision problems that are not contained in *NP* as well as problems which are not decision problems.

Solution Methods

A wide range algorithmic strategies exist to deal with combinatorial optimization problems. We can roughly classify these approaches as follows.

Exact techniques: These methods are usually based on some enumeration scheme like branch-and-bound or dynamic programming and yield provably optimal solutions. In particular, techniques utilizing linear programming work remarkably well in many cases. Nevertheless, the applicability of exact methods is often restricted to small instances due to the problem's complexity.

Heuristics: Heuristic algorithms, contrary to exact algorithms, can not provide any guarantee of optimality for the best solution obtained. The reason for the current research on suboptimal solution methods is the fact that many of them can provide good solutions within reasonable time constraints, which are often necessary in real-world application environments. Heuristic methods include the following categories: *constructive*, *enumeration* and *improvement* methods. Another category of methods worthy of mention are *approximate algorithms*, which are heuristics providing quality guarantees for their solutions.

Metaheuristics: Metaheuristics as the name suggests, are heuristic algorithms too. They

usually can be easily adapted to a wide range of different problems; this is in general not possible for traditional heuristics. However, as the main focus of this thesis lies in metaheuristics, we address them extensively in the next section.

Quadratic Assignment Problems

Quadratic Assignment Problems (QAP) belongs to a class of combinatorial optimization problems with many interesting practical applications. It was formulated by Koopmans and Beckmann (1957) for location planning of economic activities. To formulate a QAP mathematically, consider n facilities to be assigned to n locations with minimum cost. The QAP can be described by two $n \times n$ matrices $A=[a_{ij}]$ and $B=[b_{ij}]$. The goal is to find a permutation π of the set $M=\{1,2,3,\dots,n\}$, which minimizes the objective function $C(\pi)$ as in Eq.(2.1).

$$C(\pi) = \sum_{l=1}^n \sum_{t=1}^n a_{lt} b_{\pi(l)\pi(t)} . \quad (2.1)$$

In the above equation, matrix A can be interpreted as a distance matrix, i.e. a_{ij} denotes the distance between location i and location j , and B is referred to as the flow matrix, i.e. b_{ij} represents the flow of materials from facility i to facility j . We represent an assignment by the vector π . $\pi(i)$ is the location to which facility i is assigned. The quadratic objective function of the distances between the locations and the flows between the facilities gives rise to what is commonly known a quadratic assignment. Also the two famous combinatorial optimization problems, TSP and graph bipartitioning problem, are the special cases of QAP.

The QAP has been a subject of frequent study on combinatorial optimization. A recent survey article in (Burkard *et al.* 1998, Loiola *et al.* 2007) provides a fairly comprehensive outline of the development in this field. The applications as well as the techniques for solving QAP are also quite diverse. The edited volumes by Pardalos and Wolkowicz (1994), Cela (1998), and Pardalos and Pitsoulis (2000) should give a profound indication of such diversity in QAP algorithms and applications. Over the years, researchers have continued to study and develop effective algorithms for solving QAP. This is due primarily to the fact that many real-life allocation, assignment and layout problems can be formulated as QAP. Some examples include backboard wiring on electronic circuits (Steinberg 1961), campus planning (Dickey and Hopkins 1972), ranking of archaeological data (Krarup and Pruzan 1978), hospital layout planning (Elshafei 1977), scheduling parallel production lines (Geoffrion and Graves 1976), machine scheduling (Back *et al.* 1997), facility layout problem (Mavridou and Pardalos 1997), statistical data analysis (Hubert 1987), the design of typewriter keyboards and control panels (McCormick 1970) and so on. Besides other domains of engineering and design (Gero *et al.* 1997), a new application of the QAP in biology has recently been discovered in the context of indirect gradient analysis (reconstruction of the intensity of some latent environmental factors from species' responses) (Cejchan 1998). Its availability has elicited much research focused on the optimization algorithm.

Since the QAP is NP-hard (Sahni and Gonzalez 1976), only implicit enumeration approaches are known to solve them optimally. However, the computational time cost

will increase exponentially with respect to n and thus become unmanageable. Generally, it is considered unsolvable for QAP with size larger than 20 through exhaustive methods. But in practice, many applications are significantly larger than 20.

From the literature survey, heuristic approaches have played an important role in algorithms capable of providing good solutions in reasonable time. In the last decade, much research has been devoted into the development of such approaches as randomized greedy search (GRASP) (Li *et al.* 1994), simulated annealing (Burkard and Rendl 1984, Connolly 1990, Laursen 1993), neural networks (Chakrapani and Skorin-Kapov 1992, Ishii and Sato 1998), tabu search (Skorin-Kapov 1994, Skorin-Kapov 1990, Taillard 1991, Taillard 1995, Battiti and Tecchiolli 1994, Bachelet *et al.* 1996), genetic algorithms (GA) (Tate and Smith 1995), ant colonies (Maniezzo *et al.* 1994, Gambardella *et al.* 1999, Stutzle and Hoos 1999, Stutzle and Dorigo 1999, Middendorf *et al.* 2002), evolution strategies (Nissen 1994), memetic algorithms (Fleurent and Ferland 1994, Lim *et al.* 2000, Lim *et al.* 2002, Xu *et al.* 2003, Merz and Freisleben 1997, Merz and Freisleben 2000b, Brown *et al.* 1989, Carrizo and Moscato 1992, Drezner 2003, Muhlenbein 1989), threshold accepting (Nissen and Paul 1995), scatter search (Cung *et al.* 1997), iterated local search (Stutzle 2006), and tree search (Maniezzo 1998). Earlier, parallel GA has been utilized to solve this problem with significant achievements (Brown *et al.* 1989, Muhlenbein 1989). According to the *no free lunch theorems for optimization* (Wolpert and Macready 1997), it must be qualified that there is no single algorithm that has managed to perform consistently well, based on a set of standard benchmarks tested

(Burkard *et al.* 1997), which differ in size and complexity. The robustness of an algorithm when running on these benchmarks can be gauged based on its performance in terms of solution quality and computational time.

There are various kinds of benchmarks of QAPs. In this work, the large scale benchmarks are the investigation focus. The benchmark problems considered in the work are classes of synthetic problems randomly generated or created to study the robustness of algorithms for solving QAPs (Burkard *et al.* 1997). The characteristics of these benchmark problems are summarized below:

sko — This group of benchmarks was proposed by Skorin-Kapov (1990). The distance matrices of these problems are rectangular and the entries of the flow matrices are pseudo-random numbers.

lipa — These instances come from problem generators described in (Li and Pardalos 1992), they are asymmetric problems with known optimal solutions.

tai — The instances *tai-a* are uniformly generated and were proposed in (Taillard 1991), while the instances *tai-b* were introduced in (Taillard 1995). Problems of *tai-b* group are asymmetric and randomly generated.

wil — This group of benchmarks was proposed by Wilhelm and Ward (1987), the distance matrices of these problems are rectangular.

tho — This group of benchmarks was proposed by Thonemann and Bolte (1994), the distance matrices of these instances are rectangular.

1.2 Metaheuristics

During the last few decades, a new kind of heuristic algorithms has emerged which tried to use lower-level heuristic approaches to build higher-level frameworks targeted at efficiently and effectively exploring a search space. The name metaheuristic, first introduced in Glover (1986), stems from the composition of two Greek words. *Heuristic* derives from the verb *heuriskein* which means “to find” and the prefix *meta* means “beyond, in an upper level”. Glover and Kochenberger (2003) and Blum and Roli (2003) provide a survey on metaheuristics and related topics and current state of the art in the area. Here, we give an overview of some of the most widely known metaheuristic methods often applied to combinatorial optimization problems. The main features of metaheuristics are intensification (thorough investigation of promising regions of the search space, which might lead to short term improvement of the current solution) and diversification (exploration of parts of the search space not yet covered, which might require short term worsening of the current solution). These features are complementary and are therefore required for an effective metaheuristic search. The effectiveness of a metaheuristic will depend on how these two features are balanced. We distinguish between trajectory (or point-based) metaheuristics and population-based ones. Whereas in the former category only one solution is maintained at a time, the latter maintains a population of solutions. We present below an overview of some of the most popular metaheuristic methods. A special emphasis on memetic algorithm is given as it

represents a class of successful hybridization of single-point search algorithms and population-based one for solving complex combinatorial optimization problems.

Trajectory Methods

In this section we outline metaheuristics referred to as *trajectory methods*. The term trajectory methods is used because the search process performed by these methods is characterized by a trajectory in the search space. This class of methods is *neighborhood search* based algorithms. Therefore, some definitions need to be clarified (Alba 2005).

Def 2.3 A **neighborhood structure** is a function $N: S \rightarrow 2^S$ that assigns to every $s \in S$ a set of neighbors $N(s) \subseteq S$. $N(s)$ is called the neighborhood of s . Often, neighborhood structures are implicitly defined by specifying the changes that must be applied to a solution s in order to generate all its neighbors. The application of such an operator that produces a neighbor $s' \in N(s)$ of a solution s is commonly called a **move**.

A neighborhood structure together with a problem instance defines the topology of a so-called search (or fitness) landscape. A solution $s^* \in S$ is called a globally minimal solution (or global minimum) if for all $s \in S$ it holds that $f(s^*) \leq f(s)$. The set of all globally minimal solution is henceforth denoted by S^* . The introduction of a neighborhood structure enables us to define the concept of *locally* minimal solutions.

Def 2.4 A **locally minimal solution (or local minimum)** with respect to a neighborhood structure N is a solution $\hat{s}$ such that $\forall s \in N(\hat{s}): f(\hat{s}) \leq f(s)$. We call $\hat{s}$ a *strict*

locally minimal solution if $\forall s \in N(\hat{s}): f(\hat{s}) < f(s)$.

1.2.2.1 Basic Local Search

Local search algorithms (LSs) have played an important role for solving a variety of combinatorial optimizations. For example, the first local search algorithm for the traveling salesman problem was proposed in 1956/58 (Croes 1958, Flood 1956), and a local search for the facilities location problem was developed in 1963 (Armour and Buffa 1963). Figure 2.1 shows the general local search algorithm for a minimization problem; it begins with a feasible solution to the problem, a new solution with a lower objective f is searched in its neighborhood. If such a solution is found, the new solution is accepted and its neighborhood is searched for a better solution, and so on. The algorithm terminates when a local optimum is reached, i.e. when there is no solution in the neighborhood of the current best solution with a lower objective value.

```

Procedure Local-Search ( $s \in S$ ):  $S$ ;
  begin
    repeat
      Generate neighboring solution  $s' \in N(s)$ ;

      if ( $f(s') < f(s)$ ) then  $s := s'$ ;

    until  $\forall s' \in N(s): f(s') \geq f(s)$ ;

    return  $s$ ;
  end;

```

Figure 2.1 Local Search

Local search (LS) has some similarity to simple hill-climbing except that (a) the neighborhood of the current solution is searched systematically instead of randomly, and

(b) the neighborhood search is repeated until a locally optimum solution is found. There are different ways to conduct local search. For example, *best improvement* performs in a greedy way, i.e., the current solution is always replaced with the best solution in the whole neighborhood. On the other hand, *first improvement* accepts a better solution whenever it is found.

The time complexity of a certain local improvement procedure strongly depends on the size of the neighborhood and on the complexity of a single move. The effectiveness of the heuristic depends highly on the choice of an appropriate neighborhood N . One advantage of LS over other heuristics is that the configuration space can be searched very efficiently. Instead of calculating the objective value of $s' \in N(s)$, it is sufficient to calculate the difference $\Delta f = f(s) - f(s')$ (by utilizing problem-specific properties to avoid the explicit computation of $f(s)$ and $f(s')$ and to test whether Δf is less than zero.

A disadvantage of LS is that the obtained solutions are only local optima. Once a local optimum solution has been reached, the algorithm terminates. To address this weakness, many advanced local search methods were proposed.

1.2.2.2 Simulated Annealing

Simulated annealing (SA) was first introduced as a search strategy by Kirkpatrick *et al.* in 1983 (Kirkpatrick *et al.* 1983). The idea originated from the physical annealing process of metals published by Metropolis in 1953 (Metropolis *et al.* 1953). The idea of simulated annealing search is to always accept the solution under consideration if it is

better than the current solution (intensification), otherwise a worse solution may be accepted with a certain probability (diversification). Over the course of the search the probability of accepting a worse solution gradually decreases. This ensures that as the search progresses, the algorithm focuses on areas of the search space which are likely to contain good (local or global) optima. Aarts *et al.* (Aarts *et al.* 1997) have given proofs that the SA algorithm converges to the global optimum with probability 1 provided that the sequence of trials (or Markov chains) approximate a stationary distribution.

Simulated annealing has been applied to many combinatorial optimization problems, such as the quadratic assignment problem (QAP) (Connolly 1990) and the job shop scheduling (JSS) problems (Laarhoven *et al.* 1992) and large range of applications, including routing, graph partitioning, the traveling salesman problems, VLSI design, and etc (Aarts *et al.* 1997). Nowadays SA is used more as a component in metaheuristics, rather than applied as a stand-alone search algorithm. Variants of SA called Threshold Accepting and the Great Deluge Algorithm were presented in (Deuck and Scheuer 1990) and (Deuck 1993), respectively.

1.2.2.3 Tabu Search

Tabu search (TS) is one of the most successful metaheuristics for the application to combinatorial optimization problems. The basic ideas of TS were introduced in (Glover 1986). A description of the method and its concepts can be found in (Glover and Laguna 1997). The basic idea of TS is the explicit use of search history, both to escape from local minima and to implement an explorative strategy. At each iteration of a TS, the *best*

solution is chosen from a *subset* of the neighbors of the current solution considered. The subset of neighboring solutions considered at each step is made up of all the solutions in the neighborhood except the *tabu* solutions. The tabu solutions (tabu list) are usually solutions or moves that would bring the search back to a solution that has already (recently or frequently) been visited. Thus the tabu list inhibits the search from going into cycles. The tabu list is a form of short term memory that guides the search away from areas that have already been visited, but it can also be overridden if a solution that is tabu has some property that makes it particularly appealing (eg. it has the best evaluation yet encountered in the search). TS also allows for the evaluation function, used to choose the best solution in the reduced neighborhood under consideration, to be changed so that the search can be guided towards or away from certain areas. Specifically, the search can be guided *towards* solutions that are in some way similar to the current solution (*intensification*), or *away* from solutions similar to the current solution (*diversification*).

A large variety of applications of TS is summarized in Glover and Laguna (1997). Hertz *et al.* (1997) also describe the application of TS to problems including the quadratic assignment problem, graph coloring, vehicle routing and course scheduling. By incorporating domain knowledge and heuristics, TS can be a flexible metaheuristic technique with high search efficiency.

1.2.2.4 Greedy Randomized Adaptive Search Procedures

The greedy randomized adaptive search procedures (GRASP) (Feo and Resende 1995, Pitsoulis and Resende 2002) is a multi-start metaheuristic technique for combinatorial

optimization problems, in which each iteration consists of a constructive phase followed by an improvement phase. In the constructive phase, a constructive algorithm is used to create a solution from scratch. The constructive heuristic can be applied with different random seeds to create different starting solutions. The constructive heuristic is a greedy approach which tends to select the best candidate elements taken from a restricted candidate list in order to build a complete solution. It is also adaptive in that it updates the evaluation of remaining candidate elements each time a candidate element has been selected. The solutions thus obtained by the constructive heuristic are not necessarily optimal, even with respect to simple neighborhoods. The second phase of GRASP is therefore invoked in order to improve on the starting solution produced at the end of the first phase. This second phase uses a local search approach. It is therefore necessary to define a neighborhood structure and a search space. Usually the second phase utilizes simple neighborhood structures. Overall the GRASP algorithm will repeatedly apply Phase 1 and Phase 2 using different seeds at phase 1 (so as to produce different starting solutions). The best solutions found are reported at the end of the algorithm. A major advantage of GRASP algorithms is their ease of implementation. Indeed the method requires few parameters and the neighborhood structures used are quite simple. There are a variety of applications of GRASP, such as the job shop scheduling problem (Binato *et al.* 2001), the graph planarization problem (Resende and Ribeiro 1997), and assignment problems (Prais and Ribeiro 2000). A detailed and annotated bibliography references many more applications (Festa and Resende 2002).

1.2.1.5 Variable Neighborhood Search

Variable neighborhood search (VNS) is metaheuristic proposed in Hansen and Mladenovic (2001), which explicitly applies strategies for swapping between different neighborhood structures from a predefined finite set. At the initialization of the algorithm, a set of neighborhood structures has to be defined. These neighborhood structures can be arbitrarily chosen. Then, an initial solution is generated, the neighborhood index is initialized, and the algorithm iterates until a termination condition is met. Each iteration consists of three phases: shaking, local search, and move. The objective of the shaking phase is to select a solution from some neighborhood of the current local minimum that is a good starting point for the local search. The starting point should enable local search to reach a different local minimum than the current one, but should not be “too far”, otherwise the algorithm would degenerate into a simple multi-start local search with random starting solutions. Several combinatorial problems were solved using variable neighborhood search, e.g., TSP (Hansen and Mladenovic 2001), facilities location problems (Hansen and Mladenovic 1998) and its continuous version (Brimberg *et al.* 2000). Other applications are introduced in (Hansen and Mladenovic 2001).

1.2.2.6 Guided Local Search

Guided Local Search was developed by (Voudouris and Tsang 1999). The basic principle in GLS consists of dynamically changing the objective function value when local optimum has been reached. The aim is to gradually make the current local optimum “less desirable” over time so that other areas of the search space can be explored. GLS has

been successfully applied to the weighted MAXSAT (Mills and Tsang 2000), the vehicle routing problem (Kilby *et al.* 1999), the TSP, and the QAP (Voudouris and Tsang 1999).

1.2.2.7 Iterated Local Search

Iterated Local Search is a metaheuristic that is based on a simple but powerful concept. The essential idea in Iterated Local Search lies in focusing the search not on the whole solution space but on a smaller subset of the solution space which contains solutions that are locally optimal with regards to the local search method. The key in Iterated Local Search is in the sampling of the reduced set of local optima. Iterated Local Search starts with an initial solution to which the local search method is applied. Then, from that local optimum, Iterated Local Search will repeatedly perform a *perturbation* of the local optimum and apply local search to that perturbed local optimum and so forth. The local search method may be viewed as a black box which is repeatedly applied to the perturbed local optima. When a local optimum to the perturbed local optimum is obtained, an acceptance criterion is applied to decide which of the new and previous local optimum is accepted. How effective Iterated Local search is will depend on the local search method, the perturbations carried out and the acceptance criteria. Lourenco *et al.* (2003) give a thorough investigation of Iterated Local Search methods.

Population-based Methods

Population-based methods deal with a set (i.e., a population) of solutions rather than with a single solution in every iteration of the algorithm. In this way, population-based algorithms provide a natural, intrinsic way for the exploration of the search space. Yet,

the final performance strongly depends on the way the population is manipulated. The most studied population-based methods in combinatorial optimization are ant colony optimization (ACO) and evolutionary computation (EC). In ACO, a colony of artificial ants is used to construct solutions guided by the pheromone trails and heuristics information, and in EC algorithm, a population of individuals is modified by recombination and mutation operators.

1.2.2.8 Ant Colony Optimization

Ant colony optimization (ACO) is a population-based technique for optimization problems. The method was developed in analogy to the biological organization of real ants searching for food in nature. Thus, ant colony optimization is based on the indirect communication of a colony of artificial agents (the ants) mediated by artificial pheromone trails. Pheromone trails in ACO are expressed as numerical values which the ants use to probabilistically construct solutions to the problem at hand. Pheromone trails are adapted during the search to reflect search experience gained by the ants. By design, ant colony optimization techniques are constructive heuristics. An ant colony optimization starts with a population of empty initial solutions. The algorithm proceeds by iteratively adding elements to the existing partial solutions in order to form complete feasible solutions for the problem being solved. The problem is usually represented by a graph with vertices and edges, where the vertices represent states of the problem and the edges the possible connections between states. A colony of ants concurrently and asynchronously builds solutions by moving through adjacent vertices of the problem on

the construction graph thus building paths. At each state of the problem (represented by a vertex in the construction graph) the ant must decide which vertex to visit next, which corresponds to which element to add to the current partial solution. Once an ant has built a complete solution or while the ant is building the solution, the ant evaluates the current solution (which may or may not be partial) and deposits a certain amount of pheromone on the connections (or edges) it has used. This pheromone will direct the search of future ants. To prevent early convergence, ant colony optimization algorithms maintain a certain level of pheromone evaporation, that is, the pheromone deposited by previous ants decreases over time. It is also possible for ant colony optimization algorithms to perform what is known as ‘daemon’ actions, which are actions that cannot be carried out by individual ants but are instead centralized decisions. The key in a successful implementation of an ant colony optimization algorithm is how pheromone is updated, which has a direct impact on the balance between intensification and diversification of the search. Ant systems, which are the early form of ant colony optimization algorithms, were introduced by Dorigo *et al.* (Dorigo *et al.* 1991, Dorigo *et al.* 1996). Further publications can be found on <http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.htm> maintained by Marco Dorigo.

1.2.2.9 Evolutionary Computation

Evolutionary Computation (EC) algorithms are inspired by nature’s capability to evolve living beings well adapted to their environments. EC algorithms can be characterized as computational models of evolutionary processes. At each iteration a number of operators

are applied to the individuals of the current population to generate the individuals of the population of the next generation (iteration). Usually, EC algorithms use different *variation operators* to produce new individuals. For example, one operator called *recombination* or *crossover* recombines two or more individuals to produce new individuals. Another operator called *mutation* or *modification* causes a self-adaptation of individuals. The driving force in evolutionary algorithms is the *selection* of individuals based on their *fitness* (which can be based on the objective function, the result of a simulation experiment, or some other kind of quality measure). Individuals with a higher fitness have a higher probability to be chosen as members of the population of the next iteration (or as parents for the generation of new individuals). This corresponds to the principle of *survival of the fittest* in natural evolution. It is the capability of nature to adapt itself to a changing environment, which gave the inspiration for EC algorithms.

The majority of current implementations of evolutionary algorithms descend from three strongly related but independently developed approaches: genetic algorithms (GAs) initiated by Holland in (Holland 1975), evolutionary programming (EP) introduced by Fogel (Fogel 1962), and evolution strategies (ES) proposed by Rechenberg in (Rechenberg 1973).

Figure 2.2 describes the basic structure of EC algorithms. In this algorithm, P denotes the population of individuals. At each iteration a set of offspring individuals P' is generated by the application of the **Variation operators**, either recombination and/or mutation. The individuals for the next population are then selected in function **Select** (P, P') from the

union of the old population P and the set of offspring individuals P' . In the context of genetic algorithms, individuals are called *genotypes*, whereas the solutions that are encoded by individuals are called *phenotypes*. This is to differentiate between the representation of solutions and solution themselves. The choice of an appropriate representation is crucial for the success for an EC algorithm.

```
Procedure Evolutionary Computation (EC)
   $P \leftarrow \text{GenerateInitialPopulation}()$ 
  Evaluate( $P$ )
  while termination conditions not met do
     $P' \leftarrow \text{Variation operators } (P)$ 
    Evaluate( $P'$ )
     $P \leftarrow \text{Select}(P, P')$ 
  end while
output: best solution found
```

Figure 2.2 Basic structure of EC algorithm

EC algorithms have been applied to most combinatorial optimization problems and optimization problems in general (e.g. TSP in Braun (1990)). Recent successes were obtained in the rapidly growing bioinformatics area (e.g., Fogel *et al.* 2002), but also in multiobjective optimization (Coello 2000) and other fields. A detailed reference to EC application can be referred to (Back *et al.* 1997, Yao 1999). The other two population-based methods which are sometimes also considered as EC algorithms, i.e., Scatter Search and Path Relinking (Glover *et al.* 2000), and Estimation of Distribution Algorithms (EDAs) (Muhlenbein and Paab 1996).

1.2.2.10 Parallel Evolutionary Algorithm

Parallelism is considered a desired feature of any framework for optimization of computationally expensive problems. Hence, we briefly overview the diverse forms of parallel evolutionary algorithms in literature. Parallel evolutionary algorithm (PEA) represents an extension of the canonical EA. The basic concept of PEA is based on principle of tasks division of a classical EA across multiple processing nodes. The other advantage of PEA is that it facilitates speciation, a process by which different subpopulations evolve in diverse directions simultaneously. They have been shown to speed up the search process, attaining higher quality solutions on complex design problems. For this reason PEAs are not only an extension of the traditional EA sequential model, but they represent a new class of algorithms in that they manage the solution space in a different manner. Several types of PEAs (Cantu-Paz 1998, Cantu-Paz 2000a, Alba and Tomassini 2002) are briefly discussed.

Master-slave PEA — In master-slave PEAs, it is assumed that there is only a single panmictic population, i.e., a canonical EA. Like the canonical EA, each individual competes and reproduces with others. However, unlike the canonical EA, evaluations of individuals are distributed by scheduling fractions of the population among the processing slave nodes. In addition, master-slave PEA uses parallel computing to speed up the operation of the simple EA without changing the basic operations of the sequential EA. Such a model has the advantage of easy implementation since it does not alter the protocol of canonical EA search, i.e., the existing theory of simple EA still applies.

Furthermore, it serves as an efficient method of parallelization when the fitness evaluation is computationally expensive.

Fine-grained PEA — Fine-grained parallel EA consists of a single population pool, which is spatially structured. It is designed to run on closely-link massively parallel processing system, i.e., a computing system consisting of large number of processing elements and connected in a specific high-speed topology. For instance, the population of individuals in a fine-grained PEA may be organized as a two-dimensional grid, since many massively parallel computers have processing elements that are connected using this topology. Consequently, selection and mating in a fine-grained parallel EA are restricted to small groups. Nevertheless, groups overlap to permit some interactions among all the individuals so that good solutions may disseminate across the entire populations. Sometimes, fine-grained parallel EA is also termed as the cellular model (Cantu-Paz 1999a).

Multi-population PEA — Multiple population (or deme) EA may be more sophisticated, as it consists of several subpopulations that exchange individuals occasionally. This exchange of individuals is called migration and it is controlled by several parameters. Multi-population PEAs are characterized by the use of multiple subpopulations and migration operation. Multi-population PEAs are also known by various names. Since they resemble the “island model” in population genetics that considers relatively isolated demes, it is often known as “Island EA”.

In multi-population PEAs, the arrival of individuals from other populations can trigger

evolutionary changes. It is also noticed that there was relatively little change between migrations, but new solutions were found shortly after individuals were exchanged. Multi-population PEAs, although popular, is also a class of PEAs that is probably the most difficult to understand. This is because the effects of migration are not fully understood. Migration is controlled by several parameters (Cantu-Paz 2000b):

- Migration rate determines how many individuals migrate from a population.
- Migration frequency (migration interval) determines how often migration occurs.
- Migration topology determines the destination of the migrants.
- Migration policy determines which individuals migrate and which are replaced at the receiving deme.

Hierarchical PEA — Various PEA models may also be hybridized to produce other new hierarchical PEA (HPEA) models. One may form a hierarchical PEA that combines a multi-population PEA (at the upper level) and a fine-grained PEA or master-slave PEA (at the lower level). On the other hand, multi-population PEA may also be designed with multiple levels in a manner such that migration rate is faster at the lower level and possessing a communication topology which is much denser than the upper level. In general, hierarchical PEA is considered to be more effective in generating significant speed up than standalone PEA models.

Towards Memetic Algorithms

Evolutionary algorithms (EAs) are global search techniques derived from Darwin's theory of evolution by natural selection. As introduced in Section 2.2.2.2, although these algorithms have been widely applied in global optimization problems, it is now well known that it is hard for an EA to fine tune the search in complex spaces. The main reason is that EAs update a population of potential solutions (called individuals) in a global search style so that there is no local search or learning during the life span of an individual. Although EAs are very good at rapidly identifying good areas of the search space (exploration), they are often less good at refining near-optimal solution (exploitation). Hence, hybridization with other techniques can greatly improve the efficiency of search (Davis 1991, Wolpert and Macready 1997, Culberson 1998, Goldberg and Voessner 1999). On the other hand, as introduced in Section 2.2.1.1, local search performs iterative search for optimum solution in the neighbor of a candidate. Starting from a single candidate solution, local search methods use "local" information to decide which neighbor they should move to next. In general, all local search methods suffer from the same drawback as they tend to get stuck in local optima.

In order to locate the global optimum, there are two competing goals, exploration and exploitation. While exploration is important to ensure global reliability, i.e., every part of the domain is searched enough to provide a reliable estimate of the global optimum, exploitation is also important since it concentrates the search effort around the best solutions found so far by searching their neighborhoods to produce better solutions.

To solve the above problem, there have been growing interests in MAs (Moscato and Norman 1992, Steenbeek *et al.* 1998) that combine the global search ability of EAs with some kind of heuristic local search methods. The term memetic algorithm was firstly introduced by Pablo Moscato in 1989 (Moscato 1989) for local search and problem-specific knowledge enhanced EAs. MAs are population-based metaheuristic search methods inspired by Darwinian's principles of natural evolution and Dawkins' notion of a meme defined as a unit of cultural evolution that is capable of local refinements (Dawkins 1976). In the case of MAs, a "meme" refers to the strategies (e.g. local refinement, perturbation or constructive methods, etc) individuals (also called "agents") employ to improve themselves. Hence, a memetic algorithmic model of adaptation exhibits the plasticity of individuals that a strictly genetic model fails to capture. In MAs, learning occurs both in an evolutionary time-scale and in an individual's life-span time-scale. The characterization of a meme suggests that in culture evolution processes, information is not simply transmitting unaltered between individuals. In contrast, it is processed and enhanced by the communicating parts. This characteristic is accomplished in MAs by incorporating heuristics, approximate algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. Basically, most MAs can be regarded as a search strategy in which a population of optimizing agents cooperates and competes with each other (Moscato 2001).

In a more diverse context, MAs are also commonly known as Hybrid GAs (He and Mort 2000, Vazquez and Whitley 2000, Fleurent and Ferland 1994, Steenbeek *et al.* 1998),

Baldwinian GAs (Ku and Mak 1998), Lamarckian GAs (Morris *et al.* 1998), and Genetic Local Searchers (Ulder *et al.* 1991, Aarts *et al.* 1994, Kolen and Pesch 1994, Yamada and Nakano 1996, Freisleben and Merz 1996, Merz 2000, Ishibuchi *et al.* 1997, Jaszewicz 2002). Other researchers simply call their memetic approaches hybrid evolutionary algorithms without explicitly distinguishing them from other methods of hybridization (Gorges-Schleuter 1989, Brown *et al.* 1989, Bui and Moon 1994, Laszewski and Muhlenbein 1991, Burke and Smith 2000, Preux and Talbi 1999, Talbi 2002). Some of the published memetic approaches employ tabu search (Fleurent and Ferland 1994) or simulated annealing (Brown *et al.* 1989, Moscato and Norman 1992) instead of local search.

EAs and MAs have been applied in a number of different areas (Krasnogor and Smith 2005), for example, operational research and optimization, automatic programming, and machine and robot learning, the study and optimizing of models of economies, immune systems, ecologies, population genetics, the interaction between evolution and learning, and of social systems, to name but a few.

1.2.2.11 Design Issues for Memetic Algorithm

The most common use of memetic algorithms is the combination of EAs with local improvement or local search, since this symbiosis has been shown to be very successful (Merz 2001, Krasnogor and Smith 2000, Holstein and Moscato 1999). That way, the exploration abilities of the evolutionary algorithm are complemented with the exploitation capabilities of local search procedures. In early stage, a complete local

search is performed on every individual in the MA's population after each global search. This strategy is referred as Canonical Memetic Algorithm. The pseudo code of a Canonical Memetic Algorithm is shown in Figure 2.3.

Begin

Initialize: Generate an initial GA population;

While (Stopping conditions are not satisfied)

Evaluate all individuals in the population

For each individual in the population

Perform local search on it

Proceed with Lamarckian learning or Baldwinian learning operation

End For

Apply standard genetic algorithm operators to create a new population

End while

End

Figure 2.3 Canonical Memetic Algorithm

In the canonical memetic algorithm, the GA population is initially generated. Individuals in the populations will then undergo the local search procedure in the spirit of Lamarckian learning (Ong and Keane 2004, Hart 1994, Houck *et al.* 1996, Krasnogor 2002) or Baldwinian learning (Baldwin 1896). The genotypes and/or phenotypes in the original population are replaced with the improved solution depending on the learning mechanism, i.e., Lamarckian or Baldwinian learning. Subsequently new population is created through standard genetic algorithm operators, e.g., selection, mutation and crossover. Lamarckian learning means that a genetic modification (by means of local search) that is beneficial for the individual is acquired in its life span and hence subject to

propagation and selection into future generations. This is achieved by coding back into the genome any improvement found by the learning mechanism. Baldwinian learning (Baldwin 1896) in MAs can be regarded as the application of local search with the sole purpose of fitness evaluation or development. If local search directly modifies the genotypes that the EA is working with, then it is a Lamarckian MA regardless of which stage the local search was applied. If local search is only used to bias the search of the EA then it is a Baldwinian MA. It is difficult to decide a priori which method is the best, and probably no one is better in all cases. Lamarckian learning tends to substantially accelerate the evolutionary process with the caveat that it often results in premature convergence. On the other hand, Baldwinian learning is more unlikely to bring a diversity crisis within the population but it tends to be much slower than Lamarckian learning.

Memetic algorithm may be regarded as a marriage between a population-based global search and the local improvement made by each of the individuals. This has the potential to exploit the complementary advantages of EAs (generality, robustness, global search efficiency), and problem-specific local search (exploiting application-specific problem structure, rapid convergence toward local minima). In recent years, a number of independent researchers have addressed several issues relating to the trade-off between exploration and exploitation in MAs. On one hand, different local searchers can be applied to different individuals, and on the other hand, those local searchers can be used with different probability and intensities. Some of the typical issues considered in

literature are as follow:

- 1) How often should local learning be applied for, i.e., local search frequency?
- 2) On which solutions should the local learning be applied?
- 3) How long should the local learning be run, i.e., local search intensity?
- 4) Which local learning procedure or local search or meme to use?

The first issue pertinent to memetic algorithm design is to consider how often the local search should be applied for, i.e., local search frequency. Although the majority of memetic algorithms in the literature apply local search to every individual in every generation of the evolutionary algorithm, this is not mandatory. Early work by Goldberg and Voessner (1999) presented a theoretical framework for “choosing proper balance between global and local solvers to help find solutions quickly, reliably, and accurately”. They provided a theoretical alternative for efficient global-local hybrid search and characterize the optimum local search time that maximizes the probability of achieving a solution of a specified accuracy. Hart (1994) investigated the effect of local search frequency on MA search performance for continuous optimization. He studied various configurations of the local search frequency at different stages of the MA search based on the population statistics. And he also described how the optimal frequency is related to the type of problems being optimized. Land (1998) extended his work to combinatorial optimization problems and introduced the concept of “sniff” for balancing genetic and local search, also known as the local/global ratio.

The second issue pertinent to memetic algorithm design is to consider on which solutions

the local learning should be applied. Hart (1994) also studied the issue on how to best select the individuals among the EA population that should undergo local search. In his work, fitness-based and distribution-based strategies were studied for adapting the probability of applying local search on the population of chromosomes in continuous parametric search problems. Conversely, it was shown in Ku *et al.* (2000) that it may be worthwhile to applying local search on every individual if the computational complexity of the local search is relatively low. Lozano *et al.* (2004) proposed a fitness-based method to adaptively assign different local search probabilities to individuals. It was observed that the algorithm adjusts the global/local search balance according to the particularities of each problem instance. Furthermore, Molina *et al.* (2005) proposed a fitness-based adaptive method in real-coded memetic algorithm. They divided the individuals of the population into three different categories based on fitness and assigned different values of the parameters on local search probability (when local search will be applied) and local search depth (how much effort should be applied) to the individual according to the category to which that individual belongs.

As for the third issue, it is considered that how long should the local learning be run, i.e., local search intensity. Land (1998) used the concept of “sniffs” to try to gauge not only which individuals should go through a local search phase, but also the intensity. In particular, he addressed the problem of how to best integrate the local search operators with genetic operators. He proposed to use fine grain schedulers, both for mutation and crossover, that “sniffs” the basin of attraction represented by a solution. That is, instead

of performing a complete local search in every solution generated by the evolutionary operators, a partial local search is applied; only those solutions that are in a promising basin of attraction will be assigned later (by the coarse grain scheduler) an extended computational budget for local search. Incorporating with other knowledge, Krasnogor *et al.* proposed the mechanisms for tuning the local search intensity self-adaptively based on an adaptive Boltzmann method (Krasnogor and Smith 2000, Krasnogor 2004a) for local search acceptance criteria and fuzzy sets and systems method (Krasnogor and Pelta 2002, Pelta and Krasnogor 2004) within the decision rule in the local search stage according to the convergence state of the evolutionary search. Recently, Bambha *et al.* (2004) introduced a simulated heating technique for systematically integrating parameterized local search into evolutionary algorithms to achieve maximum solution quality under a fixed computational time budget. A careful balancing between global and local search has essentially been studied in the continuous optimization domain, often implemented as truncated local searches (Rudolph 1996).

Regarding the fourth issue, this essentially asks how to design an effective memetic algorithm for a specific problem. What are the local searches with some characteristics to be found more suitable for that problem and how to measure the suitability? In order to find the answer, the study may either base on the mathematical relationships between the search method and the problem or to do it adaptively during the run of the algorithm. It is worth noting that the performance of MA search is also greatly affected by the choice of neighborhood structures. Fitness landscape analysis (Merz 2000, Merz and Freisleben

1998, Merz and Freisleben 1999a, Merz and Freisleben 2000a, Merz and Freisleben 2000b, Merz 2004a, Merz 2004b) provided a way for identifying the structure of a given problem and thus a selection of local search algorithms. Although the correlation measures discussed in those papers can provide very valuable indications on the likely performance of memetic algorithm, they can sometimes be misleading. In particular, as a fitness distance correlation is measured based on one operator for a particular move (i.e. local searcher), if any of the schedulers has access to more than one local searcher, then different fitness landscapes will need to be considered. This fact was recognized by Jones in his “One Operator, One Landscape” axiom in (Jones 1995). Kallel *et al.* (2001) also showed through some statistical analysis of fitness landscapes that the choice of move operator can have a significant impact on the efficiency and effectiveness of the local search, and hence of the overall performance of the memetic algorithm. Moreover, Krasnogor (2002) has formally shown that, in order to reduce the worst-case run time of evolutionary algorithms hybridized with local search, it is necessary to choose a local search method whose move operator is different from the ones that the recombination and mutation operators induce, so called “crossover-aware” and “mutation-aware” local search. Krasnogor (2002) also investigated how to change the size and the type of neighbourhood structures dynamically in the framework of multi-meme memetic algorithms where each meme had a different neighbourhood structure, a different acceptance rule and different local search intensity. Multiple local searchers, where each one induces a different search space with distinct local optima thus avoiding local traps have been used in (Krasnogor and Smith 2000, Krasnogor and Smith 2001, Krasnogor

and Smith 2002, Caar *et al.* 2002, Krasnogor *et al.* 2002, Krasnogor 2004b, Krasnogor and Gustafson 2002, Krasnogor and Gustafson 2003, Krasnogor and Gustafson 2004), the authors resort to this technique to improve the robustness of the memetic algorithms, they have shown that the choice of which single local search operator has the best results when incorporated in an MA was entirely instance-specific. Furthermore, studies of the multimeme MAs have shown that the choice of which local search operator yielded the biggest improvement was also time-dependent. The use of a set of possible local search strategies is analogous to Dawkins' memes (Dawkins 1976). Moreover, Ozcan (2006) proposed a multimeme memetic algorithm capable of determining where to apply a hill climber and which hill climber to apply, self adaptively for solving a real-world nurse rostering problem. In addition, Smith (2002, 2003a, 2003b, 2004) proposed a co-evolutionary framework of memetic algorithms for protein structure prediction which uses a memetic algorithm as its underlying search method. Instance-specific knowledge can be learned, stored and applied by the system in the form of a population of rule. These rules determine the neighborhoods used by the local search process, which is applied to each member of the co-evolving population of candidate solutions. Similarly, in continuous optimization, Shahidi *et al.* (2004) proposed an adaptive conjugate gradient approach to co-evolve the key parameters of the conjugate gradient alongside the solution and adjusted based on regional or generational conditions of individuals in the evolution process. Magyar *et al.* (2000) proposed a mechanism that the local search method with higher fitness improvement is rewarded with greater chances of being selected. The choice of multiple local learning procedure or memes during a memetic

algorithm search in the spirit of Lamarckian learning, otherwise, known as meta-Lamarckian learning, on continuous optimization problems was also considered in Ong and Keane (2004). Ning *et al.* (2003) also concluded from their experiments that the meme choice in an MA influences the performance significantly. For a detailed taxonomy and comparative study on adaptive choice of memes in memetic algorithms, the reader may refer to (Ong *et al.* 2006).

1.2.2.12 Memetic Algorithm and Multi-Objective Optimization

In recent years, there are increasing development in the context of multi-objective MA. Ishibuchi and Murata introduced a “multiobjective genetic local search” (MOGLS) (Ishibuchi and Murata 1996) algorithm in 1996. Jaszkievicz proposed an algorithm initially called RD-MOGLS for “random directions” MOGLS (Jaszkievicz 1998), and a slight variant called the Pareto memetic algorithm (PMA) (Jaszkievicz 2003). And Knowles and Corne developed an algorithm called M-PAES (Knowles and Corne 2000, Knowles 2002).

In addition, the importance of striking a balance between genetic and local search is also emphasized in the multi-objective optimization. In some studies, local search was applied to individuals only in the final generation intending to decrease (Deb and Goel 2001) or increase (Talbi *et al.* 2001) the diversity of the final solutions. Ishibuchi *et al.* (Ishibuchi *et al.* 1997, Ishibuchi and Murata 1998) implemented a first version of a memetic algorithm for the flow-shop scheduling problem (named genetic local search in that paper) where they proposed not to examine the whole neighborhood but only a fraction

of it (i.e. best of k instead of best of all) and stop the search when no better neighbor is found after a small number of iterations. Later they also proposed to apply local search to only good offspring to improve the search ability of their genetic local search approach (Ishibuchi *et al.* 2002). In more recent study, Ishibuchi *et al.* proposed several strategies (Ishibuchi *et al.* 2003) to address the balance of global search and local search in multi-objective memetic algorithm. It has been observed that applying the local search for a limited number of iterations enables better results in the long run as reported by Burke and Smith (Burke and Smith 1999) for the maintenance scheduling problem. The use of archives of solution (Knowles and Corne 2000) can also enhance the balance of genetic and local search. Tan *et al.* (2001) also incorporated the concept of fuzzy boundary local perturbation (FBLP) with interactive local fine tuning to facilitate broader neighborhood exploration in the context of multi-objective optimization. Furthermore, measuring the characteristics of the fitness landscape as the search progresses can help in the design of a dynamic method to balance the genetic and the local search. This has been investigated by Knowles and Corne (Knowles and Corne 2002) in the context of the multi-objective quadratic assignment problem.

Furthermore, with respect to the genetic operators, it has been proposed to apply them only to parent solutions that have a certain distance between them in the genotypical space (Murata *et al.* 2000). Ishibuchi and Murata have also made a number of interesting studies on investigating restricted mating and other innovations, and have tested it on several problems (Ishibuchi and Murata 1998, Ishibuchi and Shibata 2003, Ishibuchi and

Yoshida 2002, Ishibuchi *et al.* 2002, Murata and Ishibuchi 1998, Murata *et al.* 2000, Murata *et al.* 2001, Murata *et al.* 2003). It is also worth mentioning that Abbass (2001) proposed a memetic pareto evolutionary approach, which introduced a hybrid of differential evolution (Storn and Price 1995) and backpropagation learning in order to evolve both the architecture and weights of an artificial neural network with apparently good results. An excellent exposition on the design on multi-objective MAs can be found in (Knowles and Corne 2004).

1.2.2.13 Parallel Paradigm of Memetic Algorithms

A variety of parallel memetic algorithm (PMA) models which are extensions of canonical PGA have also been studied recently. These include the blackboard parallel asynchronous memetic algorithm (Bradwell and Brown 1999), master/slave PMA (Digalakis and Margaritis 2004, Garcia *et al.* 2006, Mendes *et al.* 2005), cellular PMA (Alba *et al.* 2005), global PMA (Digalakis and Margaritis 2001) and the island model PMA (Cotta *et al.* 2003, Mendes and Linhares 2004). Digalakis and Margaritis (2001) adopted a global parallelization approach that preserves the properties, behavior, and fundamental of the sequential memetic algorithm to solve multimodal and non-linear optimization problems. Alba *et al.* (2005) also studied two different ways of embedding local search: (1) a computationally light local search step applied to every individual (called light), or (2) an in depth exploitation local search step applied with a low probability (called intensive). They found that the intensive case always derives better results. However, Mendes *et al.* (2005) apply the local search on the entire population,

but only after its convergence. The issue on local search intensity and which individuals should local learning be applied was also recently considered in the context of island model parallel memetic algorithm (Cotta *et al.* 2003).

1.2.2.14 Theoretical Study

Although theoretical analysis of memetic algorithms is difficult, there have been some advances. Sharpe (2000) empirically demonstrates that searching a landscape with a hybrid approach which mixes hillclimbing, recommendation and mutation performs moderately well on each of three different landscapes which are respectively designed to be easy for hillclimbing, easy for recommendation, and easy for mutation. The hybrid does not perform as efficiently as the appropriate pure strategy but its effectiveness is high, suggesting that a hybrid merely processing the appropriate operator(s) will be effective on a given problem. Sharpe's results also demonstrate the converse, that is, that a hybrid algorithm is only as good as its operators are appropriate. Hybrids without mutation, on a mutation-easy problem do not fare well, for example.

As discussed earlier, Merz and Freisleben (1999a) show that by analyzing properties of fitness landscapes, appropriate representations and operators can be designed. Hart (2001) shows some convergence analysis for a special type of memetic algorithms which work based on evolutionary pattern search. Krasnogor and Smith (2005) proposed a syntactic model of memetic algorithms.

The most fundamental theoretical issue related to MAs as well as EAs is their convergence property, i.e., under what condition a MA or EA will converge, and how fast

it converges. Xu and He (2005) have presented a theoretical model as well as the convergence analysis of a class of gradient-based MAs. By modeling the local search of gradient methods as an abstract strong evolution operator, the theoretical framework for abstract memetic evolutionary algorithm was derived. Moreover, the global convergence theorems and the convergence rate estimations of gradient-based MAs were also established. Later, Sudholt (2006) built a rigorous theory (theory not based on assumptions) on memetic algorithms. He analyzed a simple memetic algorithm, the (1+1) memetic algorithm ((1+1) MA), working with a population size of 1 and no crossover. He compared it with the well known (1+1) EA and randomized local search and showed that these algorithms can outperform each other drastically. He also investigated the (1+1) MA with a fixed maximal local search duration and defined a class of fitness functions where a small variation of the local search duration has a large impact on the performance of the (1+1) MA. Moreover, Ong *et al.* (2006) analyzed the global convergence properties of adaptive MAs according to their level of adaptation using the theory of Markov chain.

Furthermore, Krasnogor (2002) proposed a grammar to formulate a wide range of memetic algorithms. He also expressed that the grammar can help to envisage many more different implementations of memetic algorithms that have not been investigated.

1.2.2.15 Applications of MAs

Memetic algorithm has been applied to a vast number of optimization problems with considerable success (see (Moscato 2002) for some example references).

We can be sure that MAs constitute one of the most successful approaches for combinatorial optimization in general, and for the approximate solution of NP Optimization problems in particular such as TSP (Traveling Salesman Problem) (Moscato and Norman 1992), the Quadratic Assignment Problem (Merz and Freisleben 2000b, Misevicius 2006), the graph partitioning problem (Merz and Freisleben 2000a), etc. As for other combinatorial optimization problems, they can include network design (Ljubic and Raidle 2003, Ljubic 2004, Quintero and Pierre 2002), scheduling problems (Cheng and Gen 1997, Burke and Landa Silva 2004, Garcia *et al.* 2006, Ozcan 2006), etc. Recently, MAs have been wildly applied in other areas, such as bioinformatics (Krasnogor 2004a, Cotta *et al.* 2003), design of multidrug therapies (Neri *et al.* 2006), knowledge discovery (Nel 2004), etc. Moscato and Cotta (2002) gave a detailed overview of different applications of memetic algorithms.

1.2.2.16 Other Research on Memetic Algorithms

There are some other research trends of memetic algorithms recently. For example, Guimaraes *et al.* (2006) proposed the local learning of the objective and constraint functions prior to the local search phase of memetic algorithms, based on the samples gathered by the population through the evolutionary process. Then the local search operator is applied over this approximated model. Sorensen and Sevaux (2006) introduced a memetic algorithm with population management (MA|PM), which is capable of dynamically controlling the diversity of a small population of high-quality individuals based on certain distance measure, thereby avoiding slow or premature

convergence, and achieve excellent performance on hard combinatorial optimization problems. Memetic algorithm can even hybrid with other techniques, such as Puchinger *et al.* (2005) studied a hybrid system in which a memetic algorithm (MA) and a general purpose integer linear programming (ILP) solver based on branch-and-cut (B&C) are executed in parallel and continuously exchange information in a bidirectional, asynchronous way. In order to avoid the premature convergence, Wiering (2004) described (steady-state) memory-based memetic algorithms, which search more efficiently by increasing the diversity of the population. Each time a new individual is created, it is brought to its local optima using local search, and then the algorithm checks whether the individual was found before. If that is the case, the lowest possible fitness value is assigned, so that the individual will be replaced during the next iteration.

1.3 Summary

Many researchers have proposed different methodologies to better solve the combinatorial optimization problems, and developed various kinds of approaches for improving the effectiveness and efficiency of different search algorithms.

This chapter gave a concise review of a variety of topics related to combinatorial optimization and metaheuristics. Some basic concepts of quadratic assignment problems (QAPs) are introduced as a testbed for the development of algorithms. An introduction to

parallel evolutionary algorithms (PEAs) is provided, giving a brief overview of the different forms of parallelism used in EAs and of the multiple-population model in particular. Furthermore, a detailed review of memetic algorithm is provided since it is the focus of this dissertation. Some terms and concepts have been introduced for a better understanding of the discussion in the subsequent chapters of this dissertation.

Island Model Parallel Memetic Algorithm (PMA) with Fixed Local Search

1.4 Introduction

Among the many classes of combinatorial optimization problems, the quadratic assignment problems (QAPs) are among the hardest. Recent research has focused intensively on developing new innovative and effective meta-heuristics to handle QAPs. However, tackling large scale QAPs remains a big challenge due to the extreme computational overhead involved. The solution of large scale QAPs requires both improvement in mathematical programming algorithms and the utilization of powerful computational platforms. In this chapter, an island model PMA is proposed for large scale QAPs on a distributed computing architecture. Maintenance of the diversity of the EA population is an essential requirement. It is noted that Mori *et al.* (1998) applied a feedback thermodynamical GA to regulate the level of the diversity of the population measured by entropy. In this Chapter, using multiple subpopulations, a PMA is made to

run on each subpopulation with occasional migration of fit individuals between the subpopulations. Hence, it can facilitate a greater bias towards population diversity. Embedded in a distributed computing infrastructure, the island model PMA is expected to deliver the necessary computational horsepower to solve large instances of QAPs. Performance analysis of the island model is measured both in terms of the solution quality and the improvement in execution time, as well as the scalability of the model. Empirical studies on a diverse range of standard large scale QAP benchmarks show that the PMA achieves significantly better performance than the sequential counterpart.

The sequential memetic algorithm for QAP will be described in Section 3.2. In Section 3.3, the canonical island model PEA is used to model the island model parallel memetic algorithm (PMA) implemented by a grid-enabled solver — NetSolve. Furthermore, in Section 3.4, the empirical study will be given on the large scale instances of QAPs using the proposed island model PMA. Experimental results are shown to compare the sequential MA with island model PMA on several large scale QAPs. Finally, the impact of certain migration parameters, such as migration interval and migration topology, on the performance of PMA will be analyzed in Section 3.5. We summarize this chapter in Section 3.6.

1.5 Sequential Memetic Algorithm for QAPs

Based on the canonical memetic algorithm shown in Figure 2.3 and our previous work (Lim *et al.* 2000, Lim *et al.* 2002), a memetic algorithm for QAP is configured as shown in Figure 3.1. For the convenience of comparison later, the proposed sequential memetic algorithm shown in Figure 3.1 is abbreviated as SMA-FLS (sequential memetic algorithm with fixed local search strategy), whereby local search is applied only on the initial individuals and those have undergone modification by the evolutionary operators. In implementation, local search is applied to those new individuals generated by the mutation procedure. These individuals are considered the representation of the new search region that mutation uncovered. We intend to use local search to seek the local minimum for improving the solution quality. Therefore, the local search frequency of SMA-FLS is fixed by mutation probability P_m .

Firstly, SMA-FLS starts with an initial population of randomly generated chromosomes. To obtain a good initial population in solution quality, which affects the global performance of MA greatly, for each initial chromosome, a local search is performed.

Then, an initial generation of individuals is generated. Before the termination condition is met, the reproduction procedure is repeated as follows:

Figure 3.1 Flowchart of sequential memetic algorithm

- 1) Select two parent chromosomes based on the biased roulette wheel selection mechanism, and then perform crossover procedure with a certain probability (crossover rate) to generate a new child.
- 2) The child derived from crossover, with a certain probability (mutation rate), is subjected to a mutation procedure and local search is performed on it.
- 3) All these newly derived offspring are added into the next generation of population. Meanwhile the best few chromosomes in the current generation are also duplicated into the next generation. Thus, a new generation is derived and it replaces the current population.
- 4) Once the termination condition is met, stop the algorithm.

Finally, return the best chromosome in the population as the final solution.

Figure 3.1 shows the basic flowchart of SMA-FLS. For the QAP, genetic coding is relatively straightforward. Each chromosome can be represented as a permutation string of integers. Initially, a population of chromosomes is randomly generated. Subsequently, the algorithm will go through a number of generations until the population converges or a pre-specified maximum number of generations is reached. The mechanisms involved for each generation will be described later in Section 3.3.

1.6 Island Model Parallel Memetic Algorithm

Considering the expensive computational cost incurred by many combinatorial optimization problems, especially the large scale instances of certain combinatorial optimization problems, such as large scale QAPs, one spontaneous outcome is to develop corresponding parallel algorithms to improve the efficiency and effectiveness of the primary serial algorithm. In this section, the canonical island model PEA is used to model the island model parallel memetic algorithm (PMA). The island model parallel memetic algorithm with fixed local search strategy (PMA-FLS) for solving combinatorial optimization are introduced and described.

The main characteristic of the multi-island model is to partition the whole EA population into several subpopulations, each of which is randomly initialized and evolved through independent sequential MA simultaneously, with each island resident on a single computing node. Some selected individuals are exchanged via migration periodically. One of the islands is designated the Root Island (RI). It is tasked with coordinating communications between the multiple islands. Cooperation between subpopulations is achieved through migration of individuals.

The algorithm flow chart for each subpopulation is illustrated in Figure 3.2. Each island pursues its own MA operations independently and periodically carries out exchange with

other subpopulations through migration. For each migration interval, the migration protocol is enforced and administrated by the RI.

Figure 3.2 Algorithm flow chart for each subpopulation of multi-island model

Operators for Single Island of PMA

1.6.1.1 Fitness Evaluation, Scaling and Selection

A feasible solution of QAP of size n can be genetically coded as a permutation string of n integers, which are evaluated based on the objective function described by Eq.(2.1). For each permutation string denoted as π , the objective value obtained is normalized to

obtain a fitness value f as follows:

$$f = \begin{cases} 0 & \text{if } C(\pi) > F_z \\ 1 - C(\pi) / F_z & \text{otherwise} \end{cases} \quad (3.1)$$

The parameter F_z is a threshold value specified for each problem and it is akin to the upper bound value used in many deterministic search algorithms. Since we require that $f \geq 0$, Eq. (3.1) is clipped at $f = 0$ for $C(\pi) \geq F_z$. For this reason, we refer to F_z as a *zerofit* threshold. In our implementation of the algorithm, the threshold is defined as $F_z = K_z \times \Omega$; K_z being an integer constant while Ω refers to the known optimum or lower bound of the problem. Subsequently, the fitness values of the general population are scaled in order to avoid any unintentional bias. The approach for scaling is based on the commonly used linear scaling model (Goldberg 1989) as follows:

$$F = k_1 f + k_2, \quad (3.2)$$

where f and F denote the fitness values before and after scaling respectively. The coefficients k_1 and k_2 are chosen such that $F_{ave} = f_{ave}$ and $F_{max} = S_f \times F_{min}$. S_f refers to the scaling factor, f_{ave} the average fitness before scaling while F_{ave} , F_{max} and F_{min} are respectively the average, maximum and minimum fitness values after scaling.

Based on these conditions, the coefficients are determined as follows:

$$k_1 = \frac{(1 - S_f) \times f_{ave}}{(f_{min} \times S_f - f_{max}) + (1 - S_f) \times f_{ave}}, \quad (3.3)$$

$$k_2 = (1 - k_1) \times f_{ave}, \quad (3.4)$$

where $f_{\min}$ is the minimum fitness and $f_{\max}$ the maximum fitness before scaling.

The scaled fitness values of the chromosomes serve as a basis for selecting parent chromosomes for reproduction. In our implementation, we adopt the biased roulette wheel selection criterion, which assigns to a chromosome i in a population of m individuals a selection probability P_i proportional to its fitness value as follows:

$$P_i = \frac{F_i}{\sum_{j=1}^m F_j}. \quad (3.5)$$

Using Eq. (3.5), a chromosome with higher fitness will be awarded a proportionately higher chance of being selected for reproduction.

1.6.1.2 Crossover, Mutation and Elitism

Crossover serves as the main genetic search operator in an EA. The method implemented is a sort of position-oriented crossover (Lim *et al.* 2000) as shown in Figure 3.3. It involves two selected parent chromosomes to construct a new child chromosome by randomly choosing a crossing site. The portion from the beginning up to the crossing site of the child is replicated directly from the first parent, while the portion after the crossing site of the child is constructed by rearranging the alleles of the genes based on the order specified by the second parent. As a result, a child chromosome inherits partial information and characteristics from each of its parents.

```

Randomly pick a crossover site  $x \in \{1, 2, \dots, n-1\}$ ;
for  $i:=1$  to  $x$ 
    begin  $child[i] \leftarrow parent1[i]$ ;
    endfor( $i$ );
for  $i:=x+1$  to  $n$ 
    begin  $p \leftarrow parent2[i]$ ;
    do begin
         $flag \leftarrow 0$ ;
        for  $j:=1$  to  $x$ 
            begin if  $\{p = child[j]\}$  then
                begin  $p \leftarrow parent2[j]$ ;
                 $flag \leftarrow 1$ ;
                break;
            endif
        endfor( $j$ )
    enddo while( $flag=1$ )
     $child[i] \leftarrow p$ ;
endfor( $i$ )

```

Figure 3.3 Crossover implemented in SMA-FLS

Mutation serves as the secondary search operator for exploring new search region by altering the allele of a gene randomly. In our implementation, 2 genes are randomly picked up and their alleles swapped accordingly. In this way, the permutation constraint of the string is maintained.

Besides the operators described above, we adopt elitism in addition to the basic reproduction mechanisms. Using this strategy, a certain number of chromosomes in the current population are replicated directly into the new population. To implement this, we choose a certain number of best chromosomes from the population for each generation and duplicate them into the succeeding population. In this way, the survival of the best

members in the population is guaranteed. Such a scheme appears to be effective in improving the overall performance of the GA (Davis 1991).

1.6.1.3 *k*-gene Exchange Local Search

The standard genetic algorithm could be viewed as the global search engine, which is capable and suitable for exploring the entire search space. It explores regions of the solution landscape through the evolutionary mechanisms described earlier. Local search on the other hand is more focused in finding a good solution within a small region of the search space.

When a local search procedure is incorporated into the genetic algorithm, satisfactory performance can be obtained. Whenever genetic algorithm explores a certain region in the search space, it is rational and natural to apply some forms of heuristics method to find the “local optimum” within this region. The *k*-gene exchange local search is the procedure in SMA-FLS to seek the local optimum. Most heuristic approaches employed in local searches are driven by the primary mechanism of accepting alternative solutions with incremental improvement.

The local search procedure implemented is a form of *k*-gene exchange. The basic premise of this exchange is to carry out swapping of *k* number of genes in the chromosome, so that a better solution can be derived. *k* is a parameter that can be varied from 2 to $n/2$ to achieve the optimal performance. Although not necessarily destructive, a larger value of *k* has a tendency of introducing more drastic variations and much more computational time during the local search. The effect of *k* on the performance of the local search was

studied in Lim *et al.* (2000). It was shown that $k=2$ is reasonable from the viewpoints of solution quality and computational time. For some instances of benchmarks tested, larger values of k did produce better results. In general, choosing pair wise exchange is appropriate, which is the choice for our implementation of the SMA-FLS.

The main role of a local search procedure is to carry out search within the neighbourhood solution set of an individual being considered. For a 2-gene exchange procedure, the neighbourhood solution set refers to potential solutions that are reachable by pairwise swapping of genes. Figure 3.4 illustrates the 2-gene exchange procedure. Consider local search to be carried out on chromosome A . Two genes are selected (2nd and 5th genes) according to a certain predetermined selection scheme. As a result of the swap, chromosome A' is derived, which differs from A at the two genes specified. Based on the procedure of swapping two genes, a chromosome can be considered as a representation of a neighbourhood space of $n(n-1)/2$ potential solutions.

Figure 3.4 2-gene exchange procedure

Earlier work has relied on a GA embedded with a 2-gene exchange local search procedure for QAP (Lim *et al.* 2000, Lim *et al.* 2002). Compared with genetic algorithm,

it consistently achieved significantly better performance in terms of solution quality. To illustrate the local search procedure, consider π as the permutation chosen to undergo local search and π^* as the new permutation derived. Let δ be the *improvement factor* denoting the difference between $C(\pi)$ and $C(\pi^*)$, i.e. $\delta = C(\pi) - C(\pi^*)$. The replacement of π by π^* is performed only if δ indicates that the modified permutation π^* is better than π . We illustrate the procedure by the pseudo-code as in Figure 3.5.

```

for  $i$ : =1 to  $n$ 
  begin
    for  $p$ : =1 to  $n$ 
       $\pi^* \leftarrow \pi$ ;
      begin
        if  $\pi^*(i) \neq p$  then
          begin
             $\pi^*(i) \leftarrow p$ ;
            find  $j$  such that  $\pi(j) = p$ ;
             $\pi^*(j) \leftarrow \pi(i)$ ;
            calculate  $\delta$ ;
            if  $\delta > 0$  then  $\pi \leftarrow \pi^*$ ;
          endif
        endfor( $p$ )
      endfor( $i$ )

```

Figure 3.5 Pseudo codes of local search procedure using 2-gene exchange operation

The basic mechanism of local search involves making small adjustments to the permutation π . In principle, a modified permutation denoted as π^* will only be accepted if it is of better quality than π . This can only be determined if the objective value of π^* is known. As described earlier, the objective value is computed based on Eq.(2.1). The computational cost is significant. For this reason, it is desirable to incorporate a more

efficient approach to determine the *improvement factor* δ , which is computationally more efficient.

For a general case of k -gene exchange, let the k number of genes that are to undergo exchange be denoted by $i_1, i_2, \dots, i_k$, and the set $M = \{1, 2, \dots, n\}$ be divided into two subsets $M_1 = \{i_1, i_2, \dots, i_k\}$ and $M_2 = M - M_1$. M_1 is the set of the genes in π whose alleles differ from that of π^* . Then the idea of k -gene exchange procedure can be described as follow:

$$\begin{aligned}
 \pi^*(l) &= \pi(l), & \text{for all } l \in M_2. \\
 \pi^*(i_1) &= \pi(i_2), \\
 \pi^*(i_2) &= \pi(i_3), \\
 &\vdots \\
 \pi^*(i_k) &= \pi(i_1).
 \end{aligned} \tag{3.6}$$

By further denoting $\pi(i_{k+1}) = \pi(i_1)$, Eq. (3.6) can be written in a general form as follows:

$$\begin{aligned}
 \pi^*(l) &= \pi(l), & \text{for all } l \in M_2. \\
 \pi^*(i_r) &= \pi(i_{r+1}), & \text{for all } i_r \in M_1.
 \end{aligned} \tag{3.7}$$

The objective function of Eq.(2.1) can be rewritten in the form of Eq.(3.8) as follows:

$$C(\pi) = \sum_{l \in M_2} \sum_{t \in M_2} a_{lt} b_{\pi(l)\pi(t)} + \sum_{l \in M_1} \sum_{t \in M_2} a_{lt} b_{\pi(l)\pi(t)} + \sum_{l \in M_2} \sum_{t \in M_1} a_{lt} b_{\pi(l)\pi(t)} + \sum_{l \in M_1} \sum_{t \in M_1} a_{lt} b_{\pi(l)\pi(t)}. \tag{3.8}$$

Based on Eq.(3.7), it is clear that in Eq.(3.8), $C(\pi)$ and $C(\pi^*)$ are only different in the last three terms. Thus, the *improvement factor* δ can be computed as follows (Burkard and Derigs 1980):

$$\begin{aligned}
 \delta &= C(\pi) - C(\pi^*) \\
 &= \sum_{i_l \in M_1} \sum_{t \in M_2} [a_{i_l t} (b_{\pi(i_l)\pi(t)} - b_{\pi(i_{l+1})\pi(t)})] + \sum_{l \in M_2} \sum_{i_t \in M_1} [a_{l i_t} (b_{\pi(l)\pi(i_t)} - b_{\pi(l)\pi(i_{t+1})})] \\
 &\quad + \sum_{i_l \in M_1} \sum_{i_t \in M_1} a_{i_l i_t} (b_{\pi(i_l)\pi(i_t)} - b_{\pi(i_{l+1})\pi(i_{t+1})}) \\
 &= \sum_{p \in M_2} \sum_{r=1}^k [a_{i_r p} (b_{\pi(i_r)\pi(p)} - b_{\pi(i_{r+1})\pi(p)}) + a_{p i_r} (b_{\pi(p)\pi(i_r)} - b_{\pi(p)\pi(i_{r+1})})] \\
 &\quad + \sum_{l=1}^k \sum_{t=1}^k a_{i_l i_t} (b_{\pi(i_l)\pi(i_t)} - b_{\pi(i_{l+1})\pi(i_{t+1})}). \tag{3.9}
 \end{aligned}$$

If δ is calculated directly based on Eq. (2.1), the computation complexity of a single k -gene exchange operation is $O(n^2)$. While basing on Eq. (3.9), the complexity involved is significantly reduced to $O(kn)$. From Figure 3.4, each local search execution of the k -gene exchange operation requires n^2 iterations. This means that the total time required for the local search based on Eq. (3.9) is in the order of $O(kn \times n^2) = O(kn^3)$, instead of $O(n^2 \times n^2) = O(n^4)$ based on Eq. (2.1). In the case of $k=2$, the computational complexity reduces significantly from $O(n^4)$ to $O(2n^3)$.

Inter-islands relationships of multi-island PMA

Figure 3.6 illustrates specifically a two-island PMA model. It shows the specific

independent sequential MA operations running on each subpopulation as well as the periodic migration between the two subpopulations. The migration characteristic is controlled by several migration parameters.

Figure 3.6 Two-island PMA Model

A general multi-island PMA showing the inter-islands relationship is depicted in Figure 3.7. In the general multi-island model, all islands are identical except for the root island which holds the additional administrative duties to facilitate the migration of individuals across the different subpopulations. In Figure 3.7, the dotted line represents the migration of fit individuals between islands. The continuous line indicates that migration of individuals is achieved through the RI, which then forwards these individuals to the respective subpopulations. By doing this, we attempt to facilitate a greater bias towards population diversity. This offers opportunities to explore a wider scope of the solution landscape, reducing the tendency for local minima attraction. With this framework, it is also easy to develop larger scale applications by incorporating more islands.

Figure 3.7 Inter-islands relationships of multi-island PMA model

The migration of strings between subpopulations is a key feature of the island model. First, it allows the distribution and sharing of above-average schemata via the strings that migrate. This increases the overall selective pressure since additional reproductive trials are allocated to those strings that are fit enough to migrate (Whitley 1993). At the same time, the introduction of one or more migrant strings into the local population helps to maintain genetic diversity, since the migrant strings arrive from a different subpopulation which has evolved independently. In essence, comparing it to the conventional EA, the island model parallel EA besides facilitating parallel operations, promotes healthy solutions diversity. In general, it is often a difficult task to select the appropriate control parameters for migration between independent subpopulations in advance for any given optimization problems.

Parallel Implementation

Grid computing (Foster and Kesselman 1999) raises challenging issues in many areas of computer science, especially in the area of distributed computing. It is defined as a

hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. Many applications have been explored to capitalize on this new wave of computing paradigm (Foster and Kesselman 1999). Recently, computational grid is widely deployed to study a variety of exact (Brixius 2000, Anstreicher *et al.* 2002) and heuristic (Melab *et al.* 2006a, Melab *et al.* 2006b) optimization algorithms for complex problems.

In the present implementation of the proposed PMA-FLS, NetSolve (Casanova and Dongarra 1998) is employed. NetSolve is a computational platform which facilitates grid-based computing in a transparent and efficient manner. It provides remote access to computational resources, both hardware and software and serves as a grid middleware. For the proposed multi-island PMA framework, the root island is first created on one processor. The other islands are also created by providing them with the address of the root island. The idea is to keep the islands in coordination where the root island performs these duties. Parallelism is achieved by wrapping the subpopulation routine on a NetSolve server. Hence, using the farming client application programming interface, subpopulation routine for each island in the multi-island PMA can be readily executed in parallel on remote servers. Figure 3.8 presents the execution mechanism for the PMA implementation on the NetSolve infrastructure.

Figure 3.8 Execution mechanism for PMA implementation on NetSolve infrastructure

As shown in Figure 3.8, at the server level, each registered computing node acts as one NetSolve server executing its own subpopulation routine of independent MA operations. At the agent level, the Root Island administers the migration between different subpopulations. For each migration, the RI is required to exchange elites between islands, and redistribute the subpopulations back to the processing nodes. This process is repeated till the termination conditions of the algorithm are satisfied.

1.7 Experimental Results of PMA-FLS for QAPs

In this section, we present the empirical study of the SMA-FLS and PMA-FLS on large scale QAP benchmarks (n up to 256). In particular, we evaluate the SMA-FLS and island

model PMA-FLS both in terms of computation time and solution quality. Firstly, we evaluate the performance of two-island model PMA-FLS in comparison to the SMA-FLS. We then extend the two-island model to a multi-island model to further investigate the scalability of the island model PMA-FLS on large scale QAP instances. The effect of varying the multi-island's migration interval on the convergence speed of the PMA-FLS is also presented. The experimental results have been published in our papers (Tang *et al.* 2003a, Tang *et al.* 2003b).

Simulation Setup and Performance Measure Criteria

The algorithms were coded in C programming language and simulations were carried out on a cluster of Pentium IV 1.9 GHz workstations. Each computing node is configured with 256MB of RAM, running on Linux Redhat 7.0 operating system. For each QAP benchmark problem, we carried out ten optimization runs and the algorithms were evaluated based on their average performance.

In solving QAP, two issues are of primary concern. One is the solution quality which depends on the algorithm effectiveness, the other being the computational cost which depends on the algorithm efficiency. Many algorithms generate good solutions while incurring huge computational cost. On the other hand, those that converge to solutions quickly tend to produce poor results. In particular, we evaluate the performance of different algorithms both in terms of computation time and solution quality. Several criteria defined to measure the performance are listed as follows:

CPU time — Average computation time in seconds upon termination of the algorithm;

Generation — Average number of generations elapsed before the occurrence of the best solution;

TG — Average number of generations elapsed before the algorithm terminates;

Average — Average objective value of the solutions obtained for all the simulation runs;

Average gap — Difference between the *Average* and the best-known value of the objective function;

Best — Best solution obtained among all the simulation runs;

Gap — Difference between the best-found value and the best-known value of a benchmark problem;

Success rate — Number of times the algorithm finds the best-known solution out of all the simulation runs.

Among these criteria, *CPU time* is used to measure the computational cost of the algorithms in wall-clock time. *Generation* and *TG* provide a measure on the convergence rate of the algorithms in terms of the number of iterations rather than the wall-clock time.

Average, *Average gap*, *Best*, *Gap* and *Success rate* serve as the criteria for measuring the solution quality of the algorithm.

In particular, to compare the PMA-FLS with respect to the SMA-FLS directly, in terms of solution quality and the computational time, we define four other metrics; R_1 , R_2 , R_3 and R_4 . The definitions of these metrics are summarized as follows.

R_1 — Percentage improvement in solution quality of the PMA-FLS with respect to the SMA-FLS.

$$R_1 = (Average_{SMA-FLS} - Average_{PMA-FLS}) / Average_{SMA-FLS}$$

where $Average_{SMA-FLS}$ and $Average_{PMA-FLS}$ are the average objective value of the solution obtained for all the SMA-FLS and the PMA-FLS runs, respectively.

R_2 — Percentage reduction in *Average gap* of the PMA-FLS with respect to the SMA-FLS.

$$R_2 = (Average_{SMA-FLS} - Average_{PMA-FLS}) / (Average_{SMA-FLS} - bf)$$

where bf is the best-found value of the objective function.

R_3 — Percentage improvement in CPU time for the PMA-FLS with respect to the SMA-FLS.

$$R_3 = (Time_{SMA-FLS} - Time_{PMA-FLS}) / Time_{SMA-FLS}$$

where $Time_{SMA-FLS}$ and $Time_{PMA-FLS}$ are the average computation time in seconds of the SMA-FLS and the PMA-FLS, respectively.

R_4 — Percentage improvement in CPU time for each generation of the PMA-FLS with respect to the SMA-FLS.

$$R_4 = (Time_{SMA-FLS} / TG_{SMA-FLS} - Time_{PMA-FLS} / TG_{PMA-FLS}) / (Time_{SMA-FLS} / TG_{SMA-FLS})$$

where $TG_{SMA-FLS}$ and $TG_{PMA-FLS}$ are the average number of generations elapsed before the SMA-FLS and the PMA-FLS terminates, respectively.

The first two metrics R_1 and R_2 evaluate the PMA-FLS in terms of solution quality with respect to the SMA-FLS. The latter two metrics R_3 and R_4 compare the algorithms in terms of computational cost.

Comparison of SMA-FLS and Two-island PMA-FLS

For our simulation, the algorithm control parameters setting is summarized in Table 3.1. In addition, we consider the configuration of the migration parameters. In (Cantu-Paz 1999a), it was shown that the total amount of work expended in the parallel GA to reach the same solution quality level is less than the sequential GA. Through migration, the selection pressure is increased and this aids the parallel GA to converge faster than would be expected from dividing a large population into smaller demes. However, it should be noted that the convergence rate depends on the GA parameters and migration interval which have to be chosen appropriately for optimum performance. When migration occurs too frequently, the method degenerates to a global population model. The subpopulations tend to converge very quickly, and rarely do they produce satisfactory results. When migration is too infrequent, the injection of new genetic material could become meaningless. This is because in the settled equilibrium state, the material introduced thro-

Table 3.1 Parameters setting for SMA-FLS and two-island PMA-FLS

Parameters	SMA-FLS	Two-island PMA-FLS
Population size	200	200
Subpopulation size	NA	100
Maximum number of generations	500	500
Fitness scaling factor S_f	3	3
Crossover probability P_c	0.8	0.8
Mutation probability P_m	0.05	0.05
Zerofit threshold constant K_z	5	5
Elites size	6	NA
Elites size in each subpopulation	NA	3

NA: not applicable

ough migration may not be sufficient to make the chromosomes in a subpopulation to express similar characteristics. After a few generations, the new characteristics represented by the injected chromosomes would be lost, hence leading to premature convergence in the subpopulations. Considering all the above issues, the parameters pertaining to migration are difficult to ascertain on general problems. Through a series of empirical study, the following control parameters of migration operation have been adopted in the two-island PMA-FLS.

Migration Interval – Two-stage progressive increasing. Migration occurs every 10 generations during the initial 100 generations succeeded with every 5 generations;

Migration Rate – One chromosome per migration phase;

Migration Policy – The simple elitist strategy, where the best individual in one subpopulation replaces the worst in the other;

Migration Topology – Migration involves exchanges between the two machines.

The search stops or terminates when either one of the following criteria is satisfied:

- i. Solution stalls for more than 70 successive generations,

- ii. Maximum number of generations has been reached.

The empirical results on the benchmarks are summarized in Tables 3.2 to 3.5. The best-known value corresponding to each instance of QAP is shown in the first column (Burkard *et al.* 1997). The benchmarks where the PMA-FLS outperforms the SMA-FLS, both in terms of solution quality and CPU time are highlighted in bold.

Tables 3.2 to 3.4 present the search results of the PMA-FLS and the SMA-FLS on *sko*, *lipa* and *tai* types benchmark problems, respectively. Table 3.5 shows the simulation results on the other two large QAPs, namely, *will100* and *tho 150*. The sizes of all these problems considered in the study are relatively large and the empirical results are presented in an ascending order of problem size.

Table 3.2 Testing results of *sko* benchmarks

		CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate	R1	R2	R3	R4
sko64	PMA-FLS	90.00	157.20	227.20	48613.60	0.24%	48508	0.02%	0.00%	0.146%	38.05%	-67.91%	-120.01%
48498	SMA-FLS	53.60	227.40	297.70	48684.60	0.39%	48550	0.11%	0.00%				
sko72	PMA-FLS	94.30	154.40	224.40	66469.40	0.32%	66306	0.08%	0.00%	0.134%	29.48%	-38.80%	-65.77%
66256	SMA-FLS	67.94	198.00	268.00	66558.60	0.46%	66330	0.11%	0.00%				
sko81	PMA-FLS	107.20	154.50	224.50	91209.60	0.23%	91072	0.08%	0.00%	0.172%	42.56%	35.54%	18.63%
90998	SMA-FLS	166.30	213.40	283.40	91366.40	0.40%	91118	0.13%	0.00%				
sko90	PMA-FLS	164.20	216.10	286.10	115928.20	0.34%	115640	0.09%	0.00%	0.181%	34.76%	37.45%	47.31%
115534	SMA-FLS	262.50	171.00	241.00	116138.20	0.52%	115884	0.30%	0.00%				
sko100a	PMA-FLS	194.00	203.40	273.40	152322.80	0.21%	152122	0.08%	0.00%	0.234%	52.74%	52.10%	55.71%
152002	SMA-FLS	405.03	183.80	252.80	152680.80	0.45%	152104	0.07%	0.00%				
sko100b	PMA-FLS	183.60	171.80	241.80	154215.00	0.21%	153924	0.02%	0.00%	0.224%	51.58%	43.29%	40.19%
153890	SMA-FLS	323.73	185.00	255.00	154561.20	0.44%	154122	0.15%	0.00%				
sko100c	PMA-FLS	184.40	205.80	275.80	148140.40	0.18%	148050	0.13%	0.00%	0.095%	33.65%	69.77%	63.29%
147862	SMA-FLS	610.09	265.00	335.00	148281.60	0.28%	148054	0.13%	0.00%				
sko100d	PMA-FLS	232.10	259.90	327.40	150036.80	0.31%	149732	0.10%	0.00%	0.228%	42.72%	48.39%	60.61%
149576	SMA-FLS	449.70	179.90	249.90	150380.40	0.54%	150142	0.38%	0.00%				
sko100e	PMA-FLS	235.50	252.90	322.90	149642.20	0.33%	149198	0.03%	0.00%	0.226%	40.80%	28.99%	49.03%
149150	SMA-FLS	331.66	161.80	231.80	149981.40	0.56%	149400	0.17%	0.00%				
sko100f	PMA-FLS	206.50	214.80	284.80	149496.60	0.31%	149228	0.13%	0.00%	0.270%	46.74%	40.69%	44.37%
149036	SMA-FLS	348.16	197.10	267.10	149900.80	0.58%	149570	0.36%	0.00%				

Table 3.3 Testing results of *lipa* benchmarks

		CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate	R1	R2	R3	R4
lipa60a	PMA-FLS	79.30	138.80	208.80	108071.00	0.80%	108025	0.75%	0.00%	0.042%	5.02%	-46.31%	-79.38%
107218	SMA-FLS	54.20	186.00	256.00	108116.10	0.84%	107983	0.71%	0.00%				
lipa70a	PMA-FLS	82.10	132.80	202.80	170960.60	0.71%	170886	0.67%	0.00%	0.041%	5.45%	2.53%	-0.84%
169755	SMA-FLS	84.23	139.80	209.80	171030.10	0.75%	170893	0.67%	0.00%				
lipa80a	PMA-FLS	145.20	230.80	300.80	254771.30	0.62%	254663	0.58%	0.00%	0.061%	8.98%	-97.36%	-34.83%
253195	SMA-FLS	73.57	135.50	205.50	254926.80	0.68%	254805	0.64%	0.00%				
lipa90a	PMA-FLS	170.20	232.80	302.80	362707.40	0.58%	362622	0.55%	0.00%	0.038%	6.20%	-14.82%	18.70%
360630	SMA-FLS	148.23	144.40	214.40	362844.80	0.61%	362682	0.57%	0.00%				
lipa60b	PMA-FLS	72.64	118.60	188.60	2843870.40	12.85%	2520135	0.00%	30.00%	4.923%	31.26%	-159.43%	-136.18%
2520135	SMA-FLS	28.00	101.70	171.70	2991112.70	18.69%	2978995	18.21%	0.00%				
lipa70b	PMA-FLS	75.07	119.50	189.50	5305700.60	15.26%	4603200	0.00%	20.00%	1.883%	12.66%	29.25%	24.73%
4603200	SMA-FLS	106.10	131.60	201.60	5407543.60	17.47%	4603200	0.00%	10.00%				
lipa80b	PMA-FLS	78.60	102.90	172.90	9021037.10	16.19%	7763962	0.00%	20.00%	3.388%	20.11%	18.04%	-15.14%
7763962	SMA-FLS	95.90	179.80	242.90	9337411.30	20.27%	9305497	19.86%	0.00%				
lipa90b	PMA-FLS	136.40	134.70	204.70	14533756.70	16.36%	12490441	0.00%	20.00%	3.619%	21.08%	27.83%	35.06%
12490441	SMA-FLS	189.00	114.20	184.20	15079519.10	20.73%	15017837	20.23%	0.00%				

Table 3.4 Testing results of *tai* benchmarks

		CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate	R1	R2	R3	R4
tai60a	PMA-FLS	70.80	121.70	191.70	7362678.60	2.14%	7317376	1.51%	0.00%	0.493%	19.13%	-152.77%	-135.78%
7208572	SMA-FLS	28.01	108.82	178.82	7399125.45	2.64%	7361202	2.12%	0.00%				
tai80a	PMA-FLS	132.00	209.40	279.40	13793878.40	1.74%	13703252	1.07%	0.00%	0.509%	23.02%	-70.21%	-31.03%
13557864	SMA-FLS	77.55	145.09	215.09	13864464.36	2.26%	13756708	1.47%	0.00%				
tai60b	PMA-FLS	97.80	178.60	248.60	608700699.20	0.08%	608215054	0.00%	40.00%	0.128%	61.63%	-156.02%	-131.51%
608215054	SMA-FLS	38.20	154.80	224.80	609480646.60	0.21%	608352112	0.02%	0.00%				
tai80b	PMA-FLS	118.70	180.10	250.10	822631266.20	0.52%	818720669	0.04%	0.00%	0.484%	48.68%	-27.22%	-21.58%
818415043	SMA-FLS	93.30	169.00	239.00	826629986.30	1.00%	818629320	0.03%	0.00%				
tai64c	PMA-FLS	21.50	0.00	70.00	1855928.00	0.00%	1855928	0.00%	100.00%	0.000%	0.00%	-11.51%	-11.51%
1855928	SMA-FLS	19.28	0.00	70.00	1855928.00	0.00%	1855928	0.00%	100.00%				
tai100a	PMA-FLS	222.80	238.50	308.50	21464686.20	1.61%	21335594	1.00%	0.00%	0.271%	14.68%	33.73%	47.69%
21125314	SMA-FLS	336.21	166.00	243.50	21523065.20	1.88%	21407522	1.34%	0.00%				
tai100b	PMA-FLS	186.90	175.30	245.30	1188882832.20	0.24%	1186007112	0.00%	0.00%	0.144%	37.29%	54.25%	44.53%
1185996137	SMA-FLS	408.50	227.40	297.40	1190599484.50	0.39%	1188431954	0.21%	0.00%				
tai256c	PMA-FLS	10826.00	155.00	225.00	44819951.40	0.07%	44807062	0.04%	0.00%	0.048%	39.86%	37.46%	42.69%
44787190	SMA-FLS	17311.50	136.20	206.20	44841668.80	0.12%	44815296	0.06%	0.00%				

Table 3.5 Testing results of *wil100* and *tho150* benchmarks

		CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate	R1	R2	R3	R4
wil100	PMA-FLS	218.00	226.10	292.80	273458.20	0.15%	273236	0.07%	0.00%	0.132%	46.16%	25.87%	35.77%
273038	SMA-FLS	294.09	190.40	253.70	273818.40	0.29%	273340	0.11%	0.00%				
tho150	PMA-FLS	1428.50	308.50	368.20	8158144.60	0.30%	8140370	0.08%	0.00%	0.186%	38.56%	55.17%	62.05%
8133864	SMA-FLS	3186.81	245.40	311.70	8173382.40	0.49%	8149502	0.19%	0.00%				

Analysis on Solution Quality

From Tables 3.2 to 3.5, we can observe that the PMA-FLS outperforms the SMA-FLS significantly in terms of solution quality, especially on large QAPs. On the whole, there has been much improvement on the solution quality of the PMA-FLS, i.e. evaluated based on the *Average*, *Average gap*, *Best*, *Gap* and *Success rate* criteria on all the large scale QAPs considered. In regard to the two metrics on solution quality, R_1 and R_2 , the positive values of R_1 and R_2 on large scale QAPs indicate that the PMA-FLS clearly converges to better quality solution than the SMA-FLS. For example, in Table 3.2, the PMA-FLS is observed to arrive at better solutions than the SMA-FLS on all the *sko* type problems considered here. The PMA-FLS outperforms the SMA-FLS significantly in terms of solution quality, especially on large *sko* problems, i.e. $n > 81$. In Table 3.3, on the *lipa-a* QAP benchmark problems, although the success rate of the PMA-FLS did not improve, the *Average gap* still display some improvement. While on the *lipa-b* QAP benchmark problems, the result based on *Success rate* indicates that the PMA-FLS has greater success in locating the global optimum. In another words, the PMA-FLS is shown capable of locating the best-known solution more frequently than the SMA-FLS. When judged against existing results available in the literature, the comparison with the results of Merz and Freislenben (1997) (Abbreviated as MF1997) is shown in Table 3.6. The best results are highlighted in bold. It is noted in Table 3.4 that our search results on the *tai256c* benchmark problem (the largest instance of QAP) using the PMA-FLS is significantly better than that found in (Merz and Freislenben 1997). In (Merz and

Freislenben 1997), the *Average gap* and the *Gap* of *tai256c* were given as 0.12% and 0.06% respectively. On the other hand, we were able to attain improved *Average gap* and *Gap* of 0.07% and 0.04%, respectively. Further, it is worth nothing that the PMA-FLS is also capable of attaining search quality that is significantly better than that obtained in (Merz and Freislenben 1997) on the *tho150* problem. As shown in Table 3.5, on the *tho150* benchmark problem, the *Average gap* and *Gap* obtained in (Merz and Freislenben 1997) were 0.32% and 0.14%, respectively, while we were able to reduce these values to 0.30% and 0.08%.

Table 3.6 Comparison of PMA-FLS with MF1997

instance	best known	PMA-FLS(Average Gap)	PMA-FLS(Gap)	MF1997(Average Gap)	MF1997(Gap)
tai256c	44787190	0.07%	0.04%	0.12%	0.06%
tho150	8133398	0.30%	0.08%	0.32%	0.14%

Analysis on Computational Cost

As observed in Tables 3.2 to 3.5, on the whole, the values of the two metrics on computational cost, R_3 and R_4 , were found to be positive on large scale benchmarks ($n > 100$), indicating that the PMA-FLS is capable of converging to good quality solutions at lower CPU time as compared to the SMA-FLS. For example, it is noted in Table 3.5 that the CPU time on the *will100* problem is much lower than that presented in (Bungartz and Trajkovski 2002), even though 8 processors were used in the latter, while only 2 processors have been employed here. On the other hand, the PMA-FLS proves to be computationally inefficient on smaller size QAP problems, i.e. ($n < 100$). This is evident from the negative R_3 and R_4 metrics on smaller QAP problems.

In addition, one may observe that there exists significant difference in R_3 or R_4 , indicative of the difference in computational time among QAPs of varying sizes. The difference in computational efficiency between large scale problems and smaller problems can be observed for the *ske* family benchmarks in Figure 3.9. On smaller problems, for example, *ske64* and *ske72*, the PMA-FLS consumes more CPU time than the SMA-FLS. Compared to larger *ske* problems ($n > 81$), the ratio of communication to computing time on smaller problems is relatively much larger. The communication overhead of smaller problems incurs a larger proportion of the total CPU time than the actual computation time. Hence the communication overhead dominates the total CPU time. This results in the PMA-FLS being computationally less efficient on smaller problems. Nevertheless, the PMA-FLS is appropriate for large-size problems, since the computational time takes up a larger proportion of the total CPU time while the communication overhead for migration constitutes only a small proportion in comparison.

Figure 3.9 Ratio of communication to computing time and percentage of communication overhead for different *sko* benchmarks in two-island PMA-FLS model

Comparing to existing related work in the literature, the PMA-FLS is certainly competitive. A total of 28 QAP benchmarks have been studied and the empirical results were presented in Tables 3.2 to 3.5. On the whole, general improvements in both search quality and CPU time can be observed for QAP problem sizes greater than 60. Especially for larger scale QAPs, our studies show that the PMA-FLS consistently generates good solution quality at a considerably shorter CPU time.

Scalability Study of Multi-island PMA-FLS

To further investigate the scalability of the island model PMA-FLS on large scale QAPs, we extend our study of the two-island model to multi-island model. All the control parameters in the study of multi-island model are kept in consistent to the two-island model scheme except the number of islands given by M . Using M number of islands in the parallel MA and a total population size of 240, there are $240/M$ individuals in each

subpopulations. In our implementation, the number of processing nodes is kept equal to M , and the elite size was varied accordingly with the subpopulation size. Since we set the elite size of SMA-FLS to be 4, therefore when $M=2$, the elites for each subpopulation is 2, and when M is more than 3, the elites for each subpopulation is kept as 1.

Before we conduct the study on the scalability of the multi-island model, we present our investigation on the effect of varying the multi-island's migration interval on the convergence rate of the PMA-FLS. We fixed all other control parameters pertaining to migration policy, and focus only on the migration interval to investigate its impact on the performance of the multi-island model. Through a series of empirical study, the main control parameters of migration operation are kept in consistent to the two-island model scheme except that we vary the migration interval from 5 to 15, and use a ring migration topology by choosing the elitist of each subpopulation to migrate from one subpopulation to the nearest neighbor subpopulation in the same direction.

We studied the effect of varying the migration interval of the multi-island model on the convergence rate of the PMA-FLS on three different and relatively large scale QAP benchmarks, i.e. *sko100b*, *tai100b* and *tho150* with results shown in Figure 3.10 to Figure 3.12.

Figure 3.10 *sko100b* benchmark

Figure 3.11 *tai100b* benchmark Figure 3.12 *tho150* benchmark

As observed in Figures 3.10 to 3.12, for multi-island model, different migration intervals

showed little impact on the extent to which the computation time of multi-island model is reduced. Another point that can be observed from Figures 3.10 to 3.12 is that on the two benchmark problems, *sko100b* and *tho150*, the multi-island model did not result in significant solution quality improvement, while on *tai100b* type benchmark problem, the improvement of the solution quality using the multi-island model is remarkable. The distance matrices of both *sko100b* and *tho150* type benchmarks are rectangular, while the distance matrix of *tai100b* type benchmarks is asymmetric and randomly generated (Burkard *et al.* 1997). Hence, one may confer that the capability of multi-island model could be related to the characteristics of the problems. However, further studies would be necessary to address this issue appropriately.

Further, we study the scalability of multi-island PMA-FLS. With the increase in the number of processors, the speedup of our model is more evident. The speedup is defined as the ratio between the CPU time of sequential algorithm to that of the parallel counterpart. On the three large benchmark problems considered above, we conduct studies using 1 to 10 processors to measure the speedup. The migration interval is fixed at 10 when studying the speedup. Figures 3.13 to 3.15 present the empirical results of this study.

Figure 3.13 Speedup of *sko100b* benchmark

Figure 3.14 Speedup of *tail100b* benchmark Figure 3.15 Speedup of *tho150* benchmark

In Figures 3.13 to 3.15, it is observed that the speedup of multi-island model PMA-FLS is remarkable. As the size and complexity of the problem increases, the speedup of our

multi-island model PMA-FLS becomes more evident. The results show that for the large scale benchmark problems considered, the total CPU time reduces accordingly along with the increase in the number of computing nodes. It also indicates that for large scale QAPs, the overhead due to the communication among the processors is small and increases slightly when the number of islands is increased. This strongly demonstrates the high scalability of the island model PMA-FLS realized in a networked computing environment.

1.8 Study of Migration Topology of PMA-FLS

Migration Topology

In the island model PMA-FLS discussed above, four control parameters pertaining to the migration operation, which have a significant effect on performance are migration interval, migration rate, migration policy and migration topology. This section extends the above work on the PMA-FLS to empirically investigate the impact of migration topology of the island model on the performance when solving large scale QAPs. This work has been reported in a technical paper (Tang *et al.* 2004).

Migration is a key feature of the island model (Cantu-Paz 1999c, Cantu-Paz E. 2001). Since the control parameters on migration process affect the quality of the solutions and

the efficiency of algorithm, the users should determine these parameters carefully. Leading work on migration rates and intervals was performed by Tanese (1989). Recently, Alba and Troya (2000) and Sekaj (2004) performed a detailed study analyzing different island model parameters. Particularly, Alba and Luque (2004, 2005) analyzed the effects of different migration parameters in the growth curves and takeover times. Skolicki and Jong (2005) also studied the influence of migration rates and intervals on island models in a systematic experimental way with conclusion that the best performance is achieved with moderate migration intervals and small migration rates. Among these control parameters, some issues deserved deeper consideration, such as migration topology.

In the parallel GA, the migration topology is decisive for the efficiency and the reliability (Cantu-Paz and Goldberg 2000). If the topology has a dense connectivity (or a short diameter, or both) good solutions will spread fast to all the demes and may quickly take over the population. On the other hand, if the topology is sparsely connected (or has a long diameter), solutions will spread slower and the demes will be more isolated from each other, permitting the appearance of different solutions. These solutions may come together at a later time and recombine to form potentially better individuals.

Concerning this parameter, the island model PGA may be categorized as following either the *island model* (fully-connected demes) or the *stepping-stone model* (interaction restricted by customized logical or physical neighborhood). The choice of one arrangement against the other surely should observe the tradeoff of “time spent in

computation versus time spent in communication” (Cantu-Paz 2000a). As well, the deme connections may be either *static* (fixed *a priori*) or *dynamic* (reconfigured along the run).

Much research work in the literature has focused on this issue. Cantu-Paz (Cantu-Paz and Goldberg 1999, Cantu-Paz 1999b) theoretically investigated how the degree of the connectivity graph affects the chance that the desired solution is reached after two epochs. Cantu-Paz (Cantu-Paz 2000b) considered that topology with different numbers of neighbors affects the distribution of fitness, and found that increasing the number of neighbors results in a higher selection intensity, and suggests that migration preserves more diversity as the number of neighbors increases.

In this work, we study the performance of the PMA-FLS with one-way ring topology and random topology, respectively. The ring topology is a loosely connected topology, where the communication is limited to occur between the adjacent nodes, as depicted in Figure 3.16. The elitist emigrants (individuals) are sent to the nearest neighbor in the ring. Then in the receiving subpopulations, the immigrants replace the least fit individuals.

Figure 3.16 One-way ring topology

The random topology used in our work is a dynamic stepping stone scheme. Each

recipient subpopulation receives the immigrants from certain donor subpopulation chosen at random for each migration phrase, such that each subpopulation received migrants from a single subpopulation other than itself.

Empirical Study

For convenience, the PMA-FLS with one-way ring topology is denoted as PMA-FLS-ring, while PMA-FLS-rand indicates the PMA-FLS with random topology. In this section, we present the empirical study of the PMA-FLS-ring and PMA-FLS-rand on three large scale QAP benchmarks. In particular, we evaluate the PMA-FLS-ring and PMA-FLS-rand both in terms of computation time and solution quality. The comparison in terms of scalability for both algorithms is investigated as well.

The configuration of the PMA control parameters is summarized in Table 3.7.

Table 3.7 PMA parameters setting

MA parameters	Multi-island PMA
Population size	240
Subpopulation size	$240/M$
Elite size	2 ($M=2$)
	1 ($M \geq 3$)
Maximum number of generations	500
Fitness scaling factor S_f	3
Crossover probability P_c	0.8
Mutation probability P_m	0.05
Zerofit threshold constant K_z	5

$M \equiv$ number of islands (processing nodes)

Through a series of empirical study, the following migration control parameters have been adopted in the PMA.

Migration Interval – Migration occurs every 10 generations;

Migration Rate – One chromosome per migration phase;

Migration Policy – The simple elitist strategy, where the best individual in one subpopulation replaces the worst in the other;

Migration Topology – One-way ring topology for PMA-FLS-ring; random topology for PMA-FLS-rand.

The empirical results on the benchmarks are summarized in Tables 3.8 to 3.10. The sizes of all three problems considered in the study are relatively large. The benchmarks where the PMA-FLS-ring outperforms the PMA-FLS-rand, in terms of solution quality are highlighted in bold.

Table 3.8 Testing results of *sko100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b 153890	3-island	PMA-FLS-ring	188.80	185.60	255.60	154441.00	0.36%	153960	0.05%	0.00%
		PMA-FLS-rand	156.80	103.80	173.80	154468.20	0.38%	154126	0.15%	0.00%
	4-island	PMA-FLS-ring	174.50	282.50	352.50	154213.80	0.21%	153952	0.04%	0.00%
		PMA-FLS-rand	145.40	206.50	276.50	154273.00	0.25%	154104	0.14%	0.00%
	6-island	PMA-FLS-ring	148.80	213.30	283.30	154254.60	0.24%	154074	0.12%	0.00%
		PMA-FLS-rand	139.10	184.80	248.00	154307.80	0.27%	154104	0.14%	0.00%
	8-island	PMA-FLS-ring	136.90	173.20	243.20	154295.60	0.26%	153910	0.01%	0.00%
		PMA-FLS-rand	169.00	263.40	333.40	154257.00	0.24%	154078	0.12%	0.00%
	10-island	PMA-FLS-ring	119.60	150.80	220.80	154195.80	0.20%	153936	0.03%	0.00%
		PMA-FLS-rand	124.20	180.80	250.80	154322.80	0.28%	154090	0.13%	0.00%

Table 3.9 Testing results of *tai100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tai100b 1185996137	3-island	PMA-FLS-ring	191.70	179.00	249.00	1189426579.70	0.29%	1187378490	0.12%	0.00%
		PMA-FLS-rand	130.60	85.30	155.30	1189303083.40	0.28%	1186934079	0.08%	0.00%
	4-island	PMA-FLS-ring	178.10	268.80	332.00	1187539521.00	0.13%	1186007112	0.00%	0.00%
		PMA-FLS-rand	117.50	144.70	214.70	1187950822.00	0.17%	1186301510	0.03%	0.00%
	6-island	PMA-FLS-ring	160.10	233.70	296.70	1187892570.00	0.16%	1185996137	0.00%	10.00%
		PMA-FLS-rand	154.80	212.00	275.00	1187949722.00	0.17%	1185996137	0.00%	10.00%
	8-island	PMA-FLS-ring	153.50	261.10	318.30	1187905557.00	0.16%	1185996137	0.00%	10.00%
		PMA-FLS-rand	132.00	179.20	242.20	1188106218.00	0.18%	1185996137	0.00%	10.00%
	10-island	PMA-FLS-ring	148.00	250.00	320.00	1187927883.00	0.16%	1186052259	0.00%	0.00%
		PMA-FLS-rand	129.20	201.20	270.60	1188552881.00	0.22%	1186710060	0.06%	0.00%

Table 3.10 Testing results of *tho150* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tho150	3-island	PMA-FLS-ring	1168.40	180.20	250.20	8166026.60	0.40%	8148836	0.18%	0.00%
8133864		PHGA-FLS-rand	990.30	149.60	219.60	8166282.40	0.40%	8159792	0.32%	0.00%
	4-island	PMA-FLS-ring	935.10	317.70	376.80	8162408.00	0.35%	8145990	0.15%	0.00%
		PHGA-FLS-rand	640.70	139.40	209.40	8171126.00	0.46%	8158848	0.31%	0.00%
	6-island	PMA-FLS-ring	885.30	332.50	386.00	8157363.67	0.29%	8151408	0.22%	0.00%
		PHGA-FLS-rand	971.20	374.20	429.90	8164029.00	0.37%	8151152	0.21%	0.00%
	8-island	PMA-FLS-ring	827.70	406.10	457.60	8161512.55	0.34%	8151436	0.22%	0.00%
		PHGA-FLS-rand	664.20	278.80	340.60	8168117.40	0.42%	8153310	0.24%	0.00%
	10-island	PMA-FLS-ring	605.10	418.50	462.40	8165464.60	0.39%	8154998	0.26%	0.00%
		PHGA-FLS-rand	529.70	296.00	361.60	8171063.40	0.46%	8163262	0.36%	0.00%

1.8.1.1 Analysis on Empirical Results

From Tables 3.8 to 3.10, we can observe that the solution quality found by PMA-FLS-ring is usually better than the PMA-FLS-rand. This is especially obvious as the number of islands is increased. Evaluation is based on the *Average*, *Average gap*, *Best* and *Gap* criteria on all the three large scale QAPs considered. As for the computational time used in both migration topology schemes, it is noted that the PMA-FLS-ring usually took longer than the PMA-FLS-rand. This indicates that the PMA-FLS-ring is able to carry on with the search longer than the PMA-FLS-rand because it is able to maintain genetic diversity across the subpopulations more effectively and hence avoiding the tendency for premature convergence.

From the results, one may conclude that the island model is well suited for managing diversification and intensification during the search process. Each deme might be seen as intensification in a particular region and a great number of demes provide some

diversification in the global space. From this point of view, migration represents a tradeoff between exploration of new individuals and exploitation of highly fit individuals which have already been found. The logical relationship between subpopulations imposed by the topology of the island model PMA has an effect on this tradeoff as well. The ring topology used for the PMA ensures local communications between subpopulations. The benefit of this design is that migration occurs locally between adjacent subpopulations on the ring. This yields local exploitation of fit individuals, while globally the separate subpopulations are free to explore different types of strings independently. So this loosely connected topology help to overcome local optimum and improve the performance to some extent.

1.8.1.2 Analysis on Scalability

Observing the trend of computational time of the PMA-FLS-ring and PMA-FLS-rand, we are motivated to further investigate the scalability of the two algorithms. Figures 3.17 to 3.19 present the empirical results of scalability on the three large benchmark problems.

Figure 3.17 Total CPU time of *sko100b* benchmark Figure 3.18 Total CPU time of *tai100b* benchmark Figure 3.19 Total CPU time of *tho150* benchmark

In Figures 3.17 to 3.19, the results show that for the large scale benchmark problems considered, the total CPU time of PMA-FLS-ring reduces accordingly along with the increase in the number of computing nodes. However, the decrease of CPU time of the PMA-FLS-rand does not follow an obvious degressive trend when the number of islands

increases. It indicates that for large scale QAPs, the scalability of the PMA-FLS-ring is much better than the PMA-FLS-rand due to more appropriate tradeoff between exploration of new individuals and exploitation of highly fit individuals found. This again demonstrates that the ring topology is more helpful for improving the performance of PMA in solving large scale QAPs.

1.9 Summary

The island model PMA proposed is shown to significantly reduce the processing time in solving large scale QAPs using networked clusters of computers. The cluster is essentially a network of workstations using NetSolve as the brokering tool to facilitate distributed optimization. To verify the performance of the PMA-FLS, we conducted empirical study on several large scale QAP benchmark problems and compared it with the SMA-FLS. Further, the number of processing nodes was varied to investigate on the scalability of the island model PMA-FLS. Performance analysis of the island model PMA-FLS is measured both in terms of the solution quality and the improvement in execution time. According to the results, it can be concluded that the island model PMA-FLS is more powerful for handling large size QAPs. The experimental results show that substantial speedup can be achieved with the island model PMA-FLS, especially for the larger size benchmarks, in particular for $n \geq 100$. Finally, the impact of certain migration parameters, such as migration interval and migration topology, on the

performance of PMA has also been analyzed. From the results, it can be concluded that the PMA with ring topology maintains a more appropriate tradeoff between exploration and exploitation and hence more helpful in improving the performance of PMA for large scale QAPs.

Diversity-based Static Adaptive Strategy

1.10 Introduction

Parallel Memetic Algorithms (PMAs) are a class of modern parallel meta-heuristics that combine evolutionary algorithms, local search, parallel and distributed computing technologies for global optimization. Recent studies on PMAs for large-scale complex combinatorial optimization problems have shown that they converge to high quality solutions significantly faster than canonical GAs and MAs. With evolutionary algorithms (EAs), there is flexibility to partition the population of individuals or islands of EA subpopulations among multiple computing nodes. It is important that the intrinsic parallelism of EAs is retained when designing any MAs. Best of all, parallel EAs possess diversity preservation capabilities that alleviate the effect of premature convergence. Some extensions of MAs to parallel MAs (PMAs) have been proposed, such as the PMA-FLS studied in Chapter 3. It is worth noting that a crucial aspect of MAs or PMAs is to strike an optimum balance between the level of exploration provided by the GA, against the level of exploitation posed by the local search procedure throughout the

memetic search.

However, the extent of the application of local searches in canonical memetic algorithm is typically based on the principle of “more is better”. In canonical MAs or PMAs, it is common practice for the local search procedure to be applied on every individual/chromosome in the GA population(s) (see Figure 2.3). This is a very computationally intensive and inefficient search process. At the same time, exhaustive local search may lead to ineffective search due to premature fall in diversity during the PMA search.

Memetic algorithm may be regarded as a marriage between a population-based global search and the local improvement made by each of the individuals. This has the potential to exploit the complementary advantages of EAs (generality, robustness, global search efficiency), and problem-specific local search (exploiting application-specific problem structure, rapid convergence toward local minima). In recent years, a number of independent researchers have addressed several issues relating to the trade-off between exploration and exploitation in MAs, as reviewed in Chapter 2 (Section 2.2.3). From literature survey, there has not been much work that considered balancing global and local search in the context of parallel MA. In particular, there is very little focus in literature on how local search frequency affects the diversity level of PMAs. We will first show that excessive local search in island model PMA can be counter-productive in the sections that follow. To address this issue, we proposed a diversity-adaptive strategy (PMA-SLS) for controlling the local search frequency in island model parallel memetic

algorithms. In contrast to canonical MAs and PMAs, the diversity-based adaptive approach controls the number of individuals undergoing the local search procedure throughout the PMA evolutionary search process. PMA-SLS uses a static adaptation strategy, maintaining population diversity throughout the PMA search by using a pre-defined Gaussian distribution to adjust the local search frequency. The partial experimental results have been published in our papers (Tang *et al.* 2005, Tang *et al.* 2006a).

The rest of this chapter is organized as follows. Section 4.2 describes the canonical island model parallel memetic algorithm (PMA), with special study on the diversity profile during the canonical PMA search process. Based on the analysis on the population diversity of PMA, the diversity-based static adaptive strategy is proposed to adaptively control the local search frequency of PMA in Section 4.3. Section 4.4 presents the numerical results obtained from empirical study and provides a comprehensive quantitative/statistical comparison of PMA-SLS, PMA, and PMA-FLS (studied in Chapter 3) in the context of large scale QAPs. The search performances of the various algorithms in terms of solution quality, computational time, and solution precision are also reported in the section. Finally, we summarize this chapter in Section 4.5.

1.11 Canonical Island Model PMA

We focus on island model parallel memetic algorithm for solving large-scale combinatorial optimization problems. The pseudo-code of a canonical PMA is outlined in Figure 4.1.

```

BEGIN
Initialize  $M$  subpopulations of size  $N$  each
WHILE (termination condition not met)
    FOR each subpopulation or island
        Evaluate all individuals in the subpopulation
        For each individual in the subpopulation
            ● Apply local search to the individuals in the subpopulation.
            ● Proceed with local improvement and replace the genotype in the
              subpopulation with the improved solution.
        End For
        Create a new population based on Selection, Mutation and Crossover.
    END FOR
    For every  $P$  migration interval
        Send  $K < N$  best individuals to a neighbouring subpopulation
        Receive  $K$  individuals from a neighbouring subpopulation
        Replace  $K$  individuals in the subpopulation
    END For
END WHILE
END

```

Figure 4.1 Pseudo-code of the canonical island model parallel memetic algorithm

Initially, M subpopulations are randomly generated. Individuals in the subpopulations will then undergo the local search procedure in the spirit of Lamarckian learning. This form of learning forces the genotype to reflect the result of improvement through the placement of locally improved individual back into the population in order to compete for reproductive opportunities. The local search procedure considered here is based on

the k -gene exchange as described in Chapter 3. Subsequently new subpopulations are created through selection, mutation and crossover. For every P migration interval, the K best performing individuals in each subpopulation migrate to its neighbouring subpopulation based on the one-way ring topology according to the study in Section 3.6. Meanwhile, the subpopulation being considered will receive K individuals from a neighbouring subpopulation. The replacement scheme may be a random walk. Alternatively, the worst performing K individuals are replaced with the K migrants from its neighbour. The entire procedure repeats until the stopping conditions are satisfied.

It is generally accepted that good diversity profile over the entire evolutionary process is a primary advantage of using island model parallel memetic algorithm for solving global optimization problems. As preliminary study, we consider using the 1) 2-island PMA and 2) 2-island PGA for solving the *sko100b* QAP benchmark. Note that PGA represents a canonical parallel GA. In contrast to PMA, no form of local search is used throughout the PGA search. The diversity of each subpopulation can be measured by various means. One simple approach is based on Shannon's information entropy (Davidor and Ben-Kiki 1992), which represents an overall measure for describing the state of the dynamical system represented by the population. This is analogous to the state of a physical or information system.

Let S denotes the set of individuals that make up a subpopulation. The set S is further divided into partitions or subsets $S_1, S_2, \dots, S_Q$. Each subset S_j is a grouping of individuals with the same fitness value. The ratio of the number of individuals in a partition S_j over

the entire subpopulation can therefore be written as follows:

$$p_j = \frac{|S_j|}{\sum_{i=1}^Q |S_i|} \quad (4.1)$$

where $|S_j|$ is the cardinality of the set S_j . Based on partitioning of individuals according to the fitness values, one approach to describe the state of the dynamical system is based on Shannon's information entropy E as follows (Rosca 1995).

$$E = -\sum_{j=1}^Q p_j \log(p_j) \quad (4.2)$$

For illustration, the diversity of each subpopulation in the 2-island PMA and PGA based on the entropy measure is depicted in Figure 4.2. From Figure 4.2, it is worth highlighting the significant drop in the entropy measure of the PMA in comparison to the PGA. It appears that PMA loses search diversity much earlier than PGA due to possible excessive local searches. This significant drop in diversity for the PMA is indicative of the benefits derived from using local search in speeding up convergence rate of the search. However, it also implies the high risk of the PMA, losing out on search diversity prematurely as a result of the extensive local searches. It can be observed that this effect is more significant at the later stage of the search.

Figure 4.2 Entropy measure for PMA and PGA on the *sko100b* QAP problem

To minimize the risk of premature convergence in the PMA, it is reasonable to ask whether the effects on performance might be reduced by adapting the local search frequency in the PMA search.

1.12 Diversity-based Static Adaptive Strategy (PMA-SLS)

In Figure 4.2, it is noted that the population diversity degrades gradually with time. Since poor diversity generally occurs at the final stages of the parallel evolutionary search, it makes sense to consider reducing the local search frequency as the search progresses. In this manner, it is hoped that the high search efficiency of the canonical PMA at the initial stages of the search is preserved, while at the same time, lower local search frequency is

enforced at the later stages of the search to reduce the risk of premature convergence. Here, we present a diversity-based static adaptive strategy (PMA-SLS) for controlling the local search frequency in the PMA search. The pseudo-code of PMA-SLS is outlined in Figure 4.3.

```

BEGIN
Initialize  $M$  subpopulations of size  $N$  each
WHILE (termination condition not met)
    FOR each subpopulation or island
        Evaluate all individuals in the subpopulation
        Calculate the number of individuals undergoing local search using PMA-SLS
        strategy
        {
            
$$\gamma(\text{gen}, \mu, \sigma, \eta) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{1}{2} \left(\frac{\text{gen} - \mu}{\sigma}\right)^2\right) * \eta$$

            
$$\phi(\text{gen}, \xi; \mu, \sigma, \eta) = \gamma * \xi$$

        }
        For randomly selective  $\phi(\cdot)$  individuals in the subpopulation
            • Apply local search to the selective individuals in the subpopulation.
            • Proceed with local improvement and replace the genotype in the
              subpopulation with the improved solution.
        End For
        Create a new population based on Selection, Mutation and Crossover.
    END FOR
    For every  $P$  migration interval
        Send  $K < N$  best individuals to a neighbouring subpopulation
        Receive  $K$  individuals from a neighbouring subpopulation
        Replace  $K$  individuals in the subpopulation
    END For
END WHILE
END

```

Figure 4.3 Pseudo-code of PMA-SLS

In particular, we model the local search frequency γ of the parallel MA search process as a normal or Gaussian distribution:

$$\gamma(gen; \mu, \sigma, \eta) = \frac{1}{\sqrt{2\pi} \bullet \sigma} \exp\left(-\frac{1}{2} \left(\frac{gen - \mu}{\sigma}\right)^2\right) * \eta, \quad (4.3)$$

where gen is the evolution generation ($gen \geq 0$), μ and σ are the mean and standard deviation of the specified Gaussian distribution, respectively. η represents a scaling factor for the number of chromosomes local search applied. Using the taxonomy of adaptive evolutionary algorithm provided in (Eiben *et al.* 1999), the present adaptive strategy is termed here as diversity-based static adaptive strategy.

Using Eq.(4.3) and a subpopulation size, ξ , the number of chromosomes to apply local search, $\phi(.)$, at gen^{th} generation is then defined as

$$\phi(gen, \xi; \mu, \sigma, \eta) = \gamma * \xi, \quad (4.4)$$

where $\phi(.)$ denotes the number of selected candidates whereby local search is applied at the gen^{th} generation.

1.13 Empirical Study

In this section, we present the empirical study of the island model parallel memetic algorithm with the diversity-based static adaptive local search strategy, comparing with MA (canonical memetic algorithm), PMA and PMA-FLS on several large scale QAP benchmarks. In particular, we evaluate the performance of different algorithms both in terms of computation time and solution quality. The statistical significance based on

t -test for PMA-SLS compared with PMA is evaluated for its performance in terms of computation cost. For convenience, the abbreviations for the different algorithms used in our study are summarized below:

PMA-SLS — Island model parallel memetic algorithm with diversity-based static adaptive strategy;

PMA-FLS — Island model parallel memetic algorithm with fixed local search strategy studied in Chapter 3;

PMA — Island model parallel memetic algorithm with complete local search strategy;

MA — Canonical memetic algorithm.

The configuration of the PMA control parameters is the same as Table 3.7 except that the maximum number of generation is set to 180 here. Through a series of empirical study and based on results and experience from the work in Chapter 3, the following migration control parameters have been adopted in the PMA.

Migration Interval — Migration occurs every 10 generations;

Migration Rate — One chromosome per migration phase;

Migration Policy — Elitist strategy, whereby the best individual in one subpopulation replaces the worst in the other;

Migration Topology — One-way ring topology.

As for the parameters pertaining to the diversity-based static adaptive local search strategy of PMA-SLS, based on preliminary computational experiments, the adopted Gaussian function was configured by the parameters, where μ , σ and η were set to 0, 200 and 500, respectively. Local search frequency γ in Eq. (4.4) was updated every 10 generations.

Results Comparison — PMA-SLS vs. PMA

Firstly, we demonstrate the advantage of parallel memetic algorithm with diversity-based static adaptive local search strategy compared to complete local search. We test both PMA and PMA-SLS on the two-island model for the same benchmark, *sko100b*. The results are shown in Table 4.1.

Table 4.1 Comparison between PMA-SLS and PMA

sko100b	2-island	CPU time	Generation	TG	Solution	Gap
153890	PMA	1461	81	151	153954	0.04%
	PMA-SLS	823	84	154	153950	0.04%

Figure 4.4 Comparison of entropy between PMA-SLS and PMA

In particular, we monitor closely the change in diversity of each sub-population for PMA-SLS and PMA by measuring the entropy of each sub-population shown in Figure 4.4. From Figure 4.4, it is observed that PMA-SLS can consistently maintain a good level diversity as the evolution progresses. However, the diversity of PMA shows a significant drop in entropy, especially at the later stage, indicating that local search has a tendency to speed up convergence significantly. From an evolutionary process point of view, PMA results in poorer diversity due to excessive localized searches, especially at the later stages of evolution. This indicates a likelihood of higher ratio of duplicate solutions in the population. As a result, there is a high level of redundancy in local searches being applied to the whole sub-population over many generations. On the other hand, PMA-SLS adjusts the local search frequency according to a specific Gaussian function. In addition, as shown in Table 4.1, both PMA and PMA-SLS achieved almost the same level of solution quality, but PMA usually incurs higher computational cost due to the intensive local search. Thus, we can conclude that PMA-SLS is able to reduce the computational time significantly with little or no loss of solution quality. This is mainly attributed to its capability in maintaining a higher level of population diversity.

Tables 4.2 to 4.7 summarize the empirical results of testing on a diverse set of large scale QAP benchmarks. The results of PMA-SLS are highlighted in bold for ease of comparison. Tables 4.2 and 4.3 present a detailed comparison study between PMA-SLS, MA, PMA and PMA-FLS (the results presented in Chapter 3, PMA-FLS [1] is from (Tang *et al.* 2003a), PMA-FLS [2] is from (Tang *et al.* 2004)) on *sko100b* and *tai100b*

benchmarks, respectively. Tables 4.4 to 4.7 show the simulation results on the other classes QAPs, namely, *sko100**, *tai100a*, *will00* and *tho150*, respectively.

Table 4.2 Results of testing on *sko100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b		MA	3096.50	127.30	160.50	153955.60	0.04%	153890	0.00%	20.00%
153890	2-island	PMA-SLS	875.20	113.60	168.40	154012.80	0.08%	153904	0.01%	0.00%
		PMA	1350.00	94.70	145.90	153950.40	0.04%	153890	0.00%	20.00%
		PMA-FLS[1]	183.60	171.80	241.80	154215.00	0.21%	153924	0.02%	0.00%
	4-island	PMA-SLS	898.00	137.10	178.10	153990.80	0.07%	153902	0.01%	0.00%
		PMA	1445.90	122.20	174.60	153952.20	0.04%	153898	0.01%	0.00%
		PMA-FLS[2]	174.50	282.50	352.50	154213.80	0.21%	153952	0.04%	0.00%
	6-island	PMA-SLS	429.40	130.20	168.20	153985.00	0.07%	153890	0.00%	10.00%
		PMA	694.30	104.80	154.50	153925.40	0.02%	153890	0.00%	20.00%
		PMA-FLS[2]	148.80	213.30	283.30	154254.60	0.24%	154074	0.12%	0.00%
	10-island	PMA-SLS	289.30	95.20	148.80	153987.80	0.06%	153890	0.00%	10.00%
		PMA	439.00	111.20	144.40	153942.60	0.04%	153890	0.00%	30.00%
		PMA-FLS[2]	119.60	150.80	220.80	154195.80	0.20%	153936	0.03%	0.00%

Table 4.3 Results of testing on *tai100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tai100b	2-island	PMA-SLS	782.40	106.70	134.00	1186275856.50	0.02%	1185996137	0.00%	40.00%
1185996137		PMA-FLS[1]	186.90	175.30	245.30	1188882832.20	0.24%	1186007112	0.00%	0.00%
	4-island	PMA-SLS	647.50	92.60	102.30	1186007361.40	0.00%	1185996137	0.00%	80.00%
		PMA-FLS[2]	178.10	268.80	332.00	1187539521.00	0.13%	1186007112	0.00%	0.00%
	6-island	PMA-SLS	356.60	88.30	104.90	1186058956.40	0.01%	1185996137	0.00%	70.00%
		PMA-FLS[2]	160.10	233.70	296.70	1187892570.00	0.16%	1185996137	0.00%	10.00%
	10-island	PMA-SLS	220.70	68.70	82.70	1186053344.20	0.00%	1185996137	0.00%	80.00%
		PMA-FLS[2]	148.00	250.00	320.00	1187927883.00	0.16%	1186052259	0.00%	0.00%

Table 4.4 Results of testing on *sko100** benchmarks

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100a 152002	2-island	PMA-SLS	883.60	133.80	175.10	152188.20	0.12%	152042	0.03%	0.00%
		PMA-FLS[1]	194.00	203.40	273.40	152322.80	0.21%	152122	0.08%	0.00%
	4-island	PMA-SLS	885.20	142.40	176.80	152119.00	0.08%	152058	0.04%	0.00%
	6-island	PMA-SLS	431.90	138.90	176.90	152109.40	0.07%	152067	0.04%	0.00%
	10-island	PMA-SLS	283.20	98.10	156.00	152102.80	0.06%	152042	0.03%	0.00%
sko100c 147862	2-island	PMA-SLS	939.30	121.80	168.40	147934.80	0.05%	147862	0.00%	10.00%
		PMA-FLS[1]	184.40	205.80	275.80	148140.40	0.18%	148050	0.13%	0.00%
	4-island	PMA-SLS	845.90	111.40	160.50	147908.20	0.03%	147862	0.00%	10.00%
	6-island	PMA-SLS	416.80	106.40	151.80	147885.60	0.02%	147862	0.00%	20.00%
	10-island	PMA-SLS	284.20	107.90	151.00	147895.40	0.02%	147862	0.00%	10.00%
sko100d 149576	2-island	PMA-SLS	883.00	111.00	166.90	149803.60	0.15%	149618	0.03%	0.00%
		PMA-FLS[1]	232.10	259.90	327.40	150036.80	0.31%	149732	0.10%	0.00%
	4-island	PMA-SLS	881.20	146.70	180.00	149752.00	0.12%	149630	0.04%	0.00%
	6-island	PMA-SLS	436.80	135.80	173.80	149699.40	0.08%	149578	0.00%	0.00%
	10-island	PMA-SLS	312.60	120.10	167.20	149681.60	0.07%	149584	0.01%	0.00%
sko100e 149150	2-island	PMA-SLS	845.40	121.00	166.70	149205.80	0.04%	149150	0.00%	10.00%
		PMA-FLS[1]	235.50	252.90	322.90	149642.20	0.33%	149198	0.03%	0.00%
	4-island	PMA-SLS	898.50	114.30	164.50	149202.60	0.04%	149150	0.00%	10.00%
	6-island	PMA-SLS	452.10	113.70	156.90	149179.20	0.02%	149150	0.00%	30.00%
	10-island	PMA-SLS	274.20	91.20	130.00	149176.40	0.02%	149150	0.00%	40.00%
sko100f 149036	2-island	PMA-SLS	888.40	104.60	153.70	149232.80	0.13%	149126	0.06%	0.00%
		PMA-FLS[1]	206.50	214.80	284.80	149496.60	0.31%	149228	0.13%	0.00%
	4-island	PMA-SLS	872.10	126.10	166.70	149150.40	0.08%	149036	0.00%	10.00%
	6-island	PMA-SLS	451.70	136.10	172.30	149205.40	0.11%	149078	0.03%	0.00%
	10-island	PMA-SLS	300.30	107.00	161.40	149203.40	0.11%	149114	0.05%	0.00%

Table 4.5 Results of testing on *tail00a* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tail00a 21125314	2-island	PMA-SLS	860.00	127.20	164.60	21458262.60	1.58%	21382118	1.22%	0.00%
		PMA-FLS[1]	222.80	238.50	308.50	21464686.20	1.61%	21335594	1.00%	0.00%
	4-island	PMA-SLS	889.60	140.20	170.90	21420954.60	1.40%	21352956	1.08%	0.00%
	6-island	PMA-SLS	451.40	152.50	180.00	21373508.00	1.17%	21270370	0.69%	0.00%
	10-island	PMA-SLS	309.60	123.60	169.50	21382655.00	1.21%	21295312	0.80%	0.00%

Table 4.6 Results of testing on *wil100* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
wil100 273038	2-island	PMA-SLS	882.10	114.50	166.20	273198.80	0.06%	273054	0.01%	0.00%
		PMA-FLS[1]	218.00	226.10	292.80	273458.20	0.15%	273236	0.07%	0.00%
	4-island	PMA-SLS	895.20	115.20	165.60	273228.60	0.07%	273054	0.01%	0.00%
	6-island	PMA-SLS	445.00	99.30	164.30	273103.80	0.02%	273044	0.00%	0.00%
	10-island	PMA-SLS	295.20	93.90	161.20	273128.60	0.03%	273054	0.01%	0.00%

Table 4.7 Results of testing on *tho150* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tho150	2-island	PMA-SLS	7290.30	140.30	174.00	8148332.60	0.18%	8142700	0.11%	0.00%
8133398		PMA-FLS[1]	1428.50	308.50	368.20	8158144.60	0.30%	8140370	0.09%	0.00%
	4-island	PMA-SLS	5258.40	148.40	180.00	8144249.60	0.13%	8138428	0.06%	0.00%
		PMA-FLS[2]	935.10	317.70	376.80	8162408.00	0.36%	8145990	0.15%	0.00%
	6-island	PMA-SLS	3267.40	156.20	180.00	8145297.20	0.15%	8142554	0.11%	0.00%
		PMA-FLS[2]	885.30	332.50	386.00	8157363.67	0.29%	8151408	0.22%	0.00%
	10-island	PMA-SLS	2004.20	136.00	177.20	8144993.20	0.14%	8142102	0.11%	0.00%
		PMA-FLS[2]	605.10	418.50	462.40	8165464.60	0.39%	8154998	0.27%	0.00%

Analysis of Results

An inspection of the experimental results indicates that the island model parallel memetic algorithm with diversity-based static adaptive local search strategy can significantly improve the effectiveness and efficiency in solving large scale QAPs. The higher success rate of PMA-SLS also indicates the improved solution precision due to the higher level of diversity maintained during the evolution process for PMA-SLS. In Tables 4.2 and 4.3, comparison among PMA-SLS, PMA and PMA-FLS, shows that PMA and PMA-SLS can get much better quality than PMA-FLS, and PMA-SLS can reduce the computational time significantly with little or no lost of solution quality compared to PMA.

From a solution quality point of view, both SMA and PMA achieved much better solution quality compared to PMA-FLS. This is evident from the much improved solution gap and the higher success rate achieved, which can be attributed to the powerful search capability of memetic algorithm armed with intensive local search. From the viewpoint of computational time, compared to the serial MA, much shorter computational time is consumed by PMA-SLS and PMA to achieve almost the same level of solution quality, indicating the advantage of employing parallel memetic

algorithms. The comparison between PMA-SLS, PMA and PMA-FLS on *sko100b* benchmark is shown in Figure 4.5.

(a) Computation time

(b) Solution quality

Figure 4.5 Comparison between PMA-SLS, PMA and PMA-FLS on *sko100b* benchmark

The plot in Figure 4.5(b) shows that PMA-SLS and PMA improve the solution quality significantly compared to PMA-FLS. It is noted that the maximum number of generations for PMA-FLS was set at 500. Instead, the maximum number of generations for PMA-SLS and PMA was set to 180. This is indicative of the enhanced search capability and quick convergence speed of the PMA. As for the computational time

shown in Figure 4.5(a), the greater reliance on local search makes PMA more time-consuming than the PMA-FLS. In this respect, the island model paradigm of the parallel memetic algorithm and the distributed computing technology can help reduce the computational time significantly. Furthermore, the diversity-based static adaptive local search strategy used in PMA-SLS improves the efficiency of the PMA remarkably.

To determine the statistical significance of the reduced computational time by PMA-SLS, a t -test was used ($p < 0.05$), with the null hypothesis being that there is no difference between PMA-SLS and PMA. Based on t -test for statistical significance, the mean and the associated results of the one tail difference of two independent means, taken over 10 independent trials for *sko100b* between PMA-SLS and PMA, are presented in Table 4.8.

Table 4.8 The mean and the one tail difference of t -test for statistical significance of *sko100b* between PMA-SLS and PMA

	PMA-SLS Mean	PMA Mean	p
2-island	875.2	1350	0.00010793
4-island	898	1445.9	1.0808E-11
6-island	429.4	694.3	1.331E-05
10-island	289.3	439	0.00010352

In Table 4.8, the mean of PMA-SLS is smaller than that of PMA and the value $p < 0.05$ indicates that PMA-SLS indeed was able to reduce the computational time compared to PMA with high level of statistical significance. This validates the notion of PMA-SLS being able to search more efficiently than PMA to achieve comparable solution quality.

Similar to the *sko100b* benchmark, the effect of multiple islands processing is plotted as in Figure 4.6 for the *tai100b* benchmark. The results show that PMA-SLS can achieve much better solution quality with comparable computational time, especially for the case

where the number of processor increases to 10 machines. It is also observed that the *tai100b* QAP benchmark shows a much higher *Success rate*, indicating that the PMA-SLS has greater success in locating the global optimum. This implies that the PMA-SLS is capable of locating the best-known solution more frequently than the PMA-FLS.

(a) Computation time

(b) Solution quality

Figure 4.6 Comparison between PMA-SLS and PMA-FLS on *tai100b* benchmark

In Tables 4.4 and 4.6, based on observations of the two criteria, *Gap* and *Success Rate*, the results of the different benchmarks (*sko100** and *will100*) show that PMA-SLS can significantly improve the solution quality in comparable computational time, especially so when the number of processor increases to ten machines. In addition, the results in Table 4.5 show that PMA-SLS is even more superior compared to PMA-FLS, even for the seemingly difficult class of benchmarks, *tai100a*. Remarkable improvement in terms of solution quality was observed. The *tai100a* corresponds to a class of problems randomly generated by Taillard using a uniform distribution. In (Taillard 1995), Taillard noted that for this type of randomly generated instances, finding good solutions (about 1% and 2%) is easy, but it is extremely difficult to find the (pseudo-) optimum (solutions that we conjectured to be optimal, but for which the optimality is not proven (Taillard 1995)). This type of randomly generated instance is not that significant for practical applications of the QAP. As such, a set of non-uniformly generated random instances (*tai*b*) with the same characteristics as real-life problems was defined. As shown in Table 4.7, for the very large scale benchmark, *tho150*, PMA-SLS also can improve the solution quality significantly.

When judged against existing results available in the literature, the comparison with the results of Merz and Freislenben (1999b) and Merz and Freislenben (2000b) (Abbreviated as MF1999b and MF2000b, respectively) is shown in Table 4.9. The best results are highlighted in bold. It is noted that the results of several instances using PMA-SLS is competitive or better than that of the MAs developed by other authors. For example, the

results of *tai100b* for PMA-SLS are much better than that shown in (Merz and Freisleben 1999b). The *Average gap* of *tai100b* was reported as 0.026% in (Merz and Freisleben 1999b), with the *Success rate* being less than 50%. On the other hand, *Average gap* which is very near to 0% was achieved by our PMA-SLS using 4 or 10 machines, and the *Success rate* is very commendable, being as high as 80%. Furthermore, it is worth nothing that the PMA-SLS is also capable of attaining search quality that is significantly better than that obtained in (Merz and Freisleben 2000b) on the *sko100a* problem. As shown in Table 4.4, on the *sko100a* benchmark, the *Average gap* obtained in (Merz and Freisleben 2000b) was 0.096%, while we were able to reduce this value to 0.06%. As for the very large benchmark, *tho150*, the empirical results in Table 4.7 show that the best solution quality on average, 0.13%, is improved compared to that reported in (Merz and Freisleben 2000b), which was 0.151%.

Table 4.9 Comparison of PMA-SLS with MF1999b and MF2000b

	instance	<i>tai100b</i>	<i>sko100a</i>	<i>tho150</i>
	best known	1185996137	152002	8133398
PMA-SLS	Average gap	0.000%	0.060%	0.13%
	Success rate	80.00%	0.00%	0.00%
MF1999b	Average gap	0.026%	0.169%	NA
	Success rate	46.47%	0.00%	NA
MF2000b	Average gap	0.038%	0.096%	0.151%
	Success rate	NA	NA	NA

1.14 Summary

In this chapter, we proposed and experimentally validated the island model parallel memetic algorithm with diversity-based static adaptive local search strategy for several large scale QAPs. The empirical results are evaluated both in terms of solution quality and computational time, and a comprehensive comparative study with PMA-FLS, PMA and some recent important results of MA available in literature. According to the results, it is shown that PMA-SLS arrives at solutions that are competitive to the PMA at significantly lower computation efforts for large scale QAPs considered, with little or no lost of solution quality. We concluded that the higher success rate of PMA-SLS coupled with improved solution precision is due mainly to the intrinsic parallelism and the ability of the PMA-SLS to manage a more desirable diversity profile during the evolutionary process of the PMA-SLS.

The results obtained have opened up several issues for future research. The issue on the effect of other parameters pertaining to the selective local search strategy on PMA-SLS's performance need to be explored, in order to achieve more optimal parameters setup for further improvement in performance.

Diversity-based Dynamic Adaptive Strategy

1.15 Introduction

Chapter 4 described a diversity-based static adaptive strategy for controlling local search frequency in PMA (PMA-SLS). PMA-SLS adjusts the local search frequency of PMA based on parameterized Gaussian distribution. The empirical results has shown that PMA-SLS arrives at solutions that are competitive to the PMA at significantly lower computation efforts on the diverse large scale QAPs considered with little or no lost of solution quality. This we concluded that the higher success rate of PMA-SLS coupled with improved solution precision is due mainly to the intrinsic parallelism and the ability of the PMA-SLS to manage a more desirable diversity profile during the evolutionary process of the PMA-SLS.

However, in the PMA-SLS strategy, its efficient use presupposes tedious tuning of the parameters for the Gaussian function in Eq.(4.3). The Gaussian function used to decide on the local search frequency was problem specific. The performance of PMA-SLS is

therefore sensitive to the parameters setting and hence, less robust. PMA-SLS was configured through trial-and-error experimentation without generalization or analysis of the characteristics of the PMA with respect to population diversity, an important characteristic indicative of the population convergence level.

In this chapter, instead of fastidious tuning of the parameters setting for the Gaussian function, a diversity-based dynamic adaptive local search strategy (PMA-DLS) is proposed such that the local search frequency is adaptively adjusted based on the online fluctuation of population diversity. The empirical results show that PMA-DLS converges to competitive solutions at significantly lower computational cost when compared to the canonical MA and PMA. Furthermore, compared to PMA-SLS using static adaptation strategy, it is found that the diversity-based dynamic adaptation strategy displays better robustness in terms of solution quality across the class of QAP problems considered without requiring extra effort in selecting suitable parameters. The partial experimental results have been published in our papers (Tang *et al.* 2006b, Tang *et al.* 2006c).

The rest of this chapter is organized as follows. Based on the analysis on the population diversity of PMA, the diversity-based dynamic adaptive strategy is proposed to adaptively control the local search frequency of PMA in Section 5.2. Section 5.3 presents the numerical results obtained from empirical study and provides a comprehensive quantitative/statistical comparison of PMA-DLS, PMA-SLS, PMA, and PMA-FLS (studied in Chapter 3) in the context of large scale QAPs. The search performances of the various algorithms in terms of solution quality, computational time, and solution

precision are also reported in the section. Finally, we summarize this chapter in Section 5.4.

1.16 PMA-DLS — Parallel Memetic Algorithm with Diversity-based Dynamic Adaptive Strategy

In Figure 4.2, it is noted that the population diversity degrades gradually with the evolving generation. Online entropy measure may also be used to provide dynamic information about the stage of the evolutionary search process and the degree of diversity of each subpopulation. Since population diversity represents a crucial characteristic of the PMA, the approach considered here makes use of online entropy measure to adapt the local search frequency. The method considered here is the diversity-based dynamic adaptive strategy or PMA-DLS in short. The pseudo-code of PMA-DLS is outlined in Figure 5.1.

```

BEGIN
Initialize  $M$  subpopulations of size  $N$  each
WHILE (termination condition not met)
    FOR each subpopulation or island
        Evaluate all individuals in the subpopulation
        Calculate the number of individuals undergoing local search using PMA-DLS
        strategy
        {
            
$$\beta(gen) = 1 + \frac{E(gen) - E(gen - k)}{E(gen - k)}$$

            
$$\phi(gen) = \begin{cases} \xi, & gen = 0 \\ \text{Min}[\phi(gen - k) * \lfloor \beta(gen) \rfloor, \xi], & gen > 0 \end{cases}$$

        }
        For randomly selective  $\phi(.)$  individuals in the subpopulation
            ● Apply local search to the selective individuals in the subpopulation.
            ● Proceed with local improvement and replace the genotype in the
              subpopulation with the improved solution.
        End For
        Create a new population based on Selection, Mutation and Crossover.
    END FOR
For every  $P$  migration interval
    Send  $K < N$  best individuals to a neighbouring subpopulation
    Receive  $K$  individuals from a neighbouring subpopulation
    Replace  $K$  individuals in the subpopulation
END For
END WHILE
END

```

Figure 5.1 Pseudo-code of PMA-DLS

Here, the dynamic local search frequency β in PMA-DLS can be defined based on the online entropy ratio given by

$$\beta(gen) = 1 + \frac{E(gen) - E(gen - k)}{E(gen - k)}, \quad (5.1)$$

where $E(gen)$ and $E(gen - k)$ ($gen \geq k$) are the population entropy measure (as per Eq.(4.2)) at the gen^{th} and $(gen - k)^{th}$ generation, respectively.

Thus, the number of chromosomes that undergo local learning at the gen^{th} generation in each subpopulation, $\phi(.)$, is defined based on online diversity of the subpopulation as per Eq.(5.2).

$$\phi(gen) = \begin{cases} \xi, & gen = 0 \\ \text{Min}[\phi(gen-k) * \lfloor \beta(gen) \rfloor, \xi], & gen > 0 \end{cases} \quad (5.2)$$

where ξ is the subpopulation size.

1.17 Empirical Study

To demonstrate the capability of the proposed strategy described in the above section, a series of empirical studies on solving complex combinatorial optimization problem, in particular the large scale quadratic assignment problem (QAP) were conducted. The experimental setup is the same as that in Chapter 4 (see Section 4.4). We present a series of empirical comparison of results for PMA-DLS, PMA-SLS, PMA and PMA-FLS on several large scale QAP benchmarks (The above abbreviations of algorithms are same as described in Section 4.4). In particular, we evaluate the performance of different algorithms both in terms of computation time and solution quality. The statistical significance based on t -test for PMA-SLS and PMA-DLS compared with PMA is evaluated for its performance in terms of computation cost.

Results Comparison — PMA-DLS vs. PMA-SLS

For parameters pertaining to PMA-SLS in Eq. (4.4) in Chapter 4, the subpopulation size, ξ , is a constant for certain number of islands in the PMA and μ is set to zero. The other two parameters (σ, η) were tuned in order to adjust the local search frequency for each generation gen . To decide on the appropriate configuration, significant effort was expended on parameters tuning in order to achieve a desirable level of performance. Various parameters setting for Gaussian function can be experimented to configure the PMA-SLS. For example, in Figure 5.2, three Gaussian functions denoted as γ_1 , γ_2 and γ_3 with different parameters setting are shown. The corresponding number (Num) of individuals where local search is applied can be determined based on Eq. (4.4). Local search frequency γ in Eq. (4.4) was updated every 10 generations. According to Figure 5.2, application of γ_1 , γ_2 and γ_3 will result in different local search frequency applied in the PMA. γ_3 , the upper curve results in higher frequency of local search while γ_2 , the lower curve indicates a lower frequency. Based on the application of γ_1 , γ_2 and γ_3 , the corresponding PMA-SLS-1, PMA-SLS-2, and PMA-SLS-3 were derived respectively. Meanwhile, PMA-DLS is more straightforward, with fewer parameters setting required. Only parameter k is required to be set in Eq. (5.2). Here, k is set to 10.

Figure 5.2 Application of Gaussian functions to determine number of chromosomes selected for local search according to PMA-SLS

Table 5.1 Comparison of PMA-DLS and PMA-SLS with γ_1 , γ_2 and γ_3

		CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b 2-island 153890	PMA-SLS-1	875.20	113.60	168.40	154012.80	0.08%	153904	0.01%	0.00%
	PMA-SLS-2	563.80	134.10	174.70	154114.60	0.16%	153962	0.05%	0.00%
	PMA-SLS-3	903.20	119.20	171.30	154016.60	0.08%	153904	0.01%	0.00%
	PMA-DLS	859.40	125.30	169.00	154020.80	0.08%	153920	0.02%	0.00%

We first carried out experimental study to gauge the effect of the choice of Gaussian function on the performance of PMA-SLS. The results presented in Table 5.1 are based on comparison of PMA-SLS and PMA-DLS on one particular benchmark. This experiment shows that PMA-DLS could produce good solutions with 0.08% average gap, consuming 859.40 seconds of CPU time. In comparison, the 3 variants of PMA-SLS vary in terms of solution quality and CPU time. In terms of CPU time, PMA-SLS-3 requires as much as 903.20 seconds while PMA-SLS-2 takes up 563.80 seconds of CPU time. On solution quality, the average gap of the PMS-SLS with the three configurations falls into the range of 0.08% to 0.16%. This may be due to the different number of individuals undergoing local search in PMA-SLS, especially at the later stages of the evolution

process. For example, the number of individuals whereby local search is applied in PMA-SLS-3 is much larger than that for PMA-SLS-2 (when $gen > 100$). Consequently, PMA-SLS-3 produced better solution (0.08%) quality than PMA-SLS-2 (0.16%). However, PMA-SLS-3 takes up more computational time. Between the 3 selection functions experimented, it appears that γ_1 ($\sigma = 200, \eta = 500$) produced the most competitive results in terms of solution quality and computational cost.

Results Comparison — PMA-DLS, PMA-SLS and PMA

To demonstrate the advantage of PMA-SLS and PMA-DLS, a comparison among PMA-SLS, PMA-DLS and PMA on the two-island model for the same benchmark, *sko100b*, is shown in Table 5.2.

Table 5.2 Comparison among PMA-SLS, PMA-DLS and PMA

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b 2-island	PMA		1350.00	94.70	145.90	153950.40	0.04%	153890	0.00%	20.00%
	PMA-SLS		875.20	113.60	168.40	154012.80	0.08%	153904	0.01%	0.00%
	PMA-DLS		859.40	125.30	169.00	154020.80	0.08%	153920	0.02%	0.00%

In Table 5.2, PMA-SLS and PMA-DLS produce competitive solutions although the frequency of local search of PMA-SLS and PMA-DLS never exceed that of PMA which maintain the highest local search frequency throughout the evolution process. This is due mainly to the ability of PMA-SLS and PMA-DLS to manage a more desirable diversity profile as the search progresses, especially at the later stage of the evolution process, compared to the poor diversity profile in PMA. The diversity of each subpopulation for PMA-SLS, PMA-DLS and PMA, measured by the entropy, was traced in our simulation and shown in Figure 5.3.

Figure 5.3 Comparison of diversity among PMA-SLS, PMA-DLS and PMA

According to Figure 5.3, PMA-SLS and PMA-DLS can consistently maintain a good level of diversity as the evolution progresses. However, the diversity of PMA shows a significant drop in entropy, especially at the later stages, indicating that local search has a tendency to speed up convergence significantly. From an evolutionary process point of view, PMA results in poorer diversity due to excessive localized searches, especially at the later stage of evolution. On the other hand, PMA-SLS adjusts the local search frequency according to a specific Gaussian function. PMA-DLS adjusts the local search frequency based on changes in population diversity. The number of individuals to apply local search is then adjusted dynamically, enabling PMA-DLS to maintain a consistent

level of population diversity. This in turn enhances the capacity of PMA-DLS to produce good solutions. A significant observation from Table 5.2 is that PMA-SLS, PMA-DLS and PMA achieved almost the same level of solution quality, with PMA incurring higher computational cost due to the intensive local search. PMA-SLS and PMA-DLS therefore show a potential for reducing computational time significantly with little or no loss of solution quality. This is mainly attributed to its capability to maintain a higher level of population diversity.

Overall Comparison of Results and Analysis

For the purpose of detailed comparison among PMA-SLS, PMA-DLS and PMA, Tables 5.3 to 5.8 summarize the empirical results of testing on a diverse set of large scale QAP benchmarks. Tables 5.3 and 5.4 present a detailed simulation results for PMA-DLS, PMA, PMA-SLS and PMA-FLS on *sko100b* and *tai100b* benchmarks, respectively. Tables 5.5 to 5.8 show the simulation results on the other classes of QAP, namely, *sko100**, *tai100a*, *wil100* and *tho150*, respectively.

Table 5.3 Results of testing on *sko100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b		MA	3096.50	127.30	160.50	153955.60	0.04%	153890	0.00%	20.00%
153890	2-island	PMA-DLS	859.40	125.30	169.00	154020.80	0.08%	153920	0.02%	0.00%
		PMA-SLS	875.20	113.60	168.40	154012.80	0.08%	153904	0.01%	0.00%
		PMA-FLS[1]	183.60	171.80	241.80	154215.00	0.21%	153924	0.02%	0.00%
		PMA	1350.00	94.70	145.90	153950.40	0.04%	153890	0.00%	20.00%
	4-island	PMA-DLS	885.70	131.40	170.00	153977.40	0.06%	153900	0.01%	0.00%
		PMA-SLS	898.00	137.10	178.10	153990.80	0.07%	153902	0.01%	0.00%
		PMA-FLS[2]	174.50	282.50	352.50	154213.80	0.21%	153952	0.04%	0.00%
		PMA	1445.90	122.20	174.60	153952.20	0.04%	153898	0.01%	0.00%
	6-island	PMA-DLS	413.80	126.80	164.20	153951.20	0.04%	153890	0.00%	20.00%
		PMA-SLS	429.40	130.20	168.20	153985.00	0.07%	153890	0.00%	10.00%
		PMA-FLS[2]	148.80	213.30	283.30	154254.60	0.24%	154074	0.12%	0.00%
		PMA	694.30	104.80	154.50	153925.40	0.02%	153890	0.00%	20.00%
	10-island	PMA-DLS	276.80	98.30	159.50	153974.40	0.05%	153924	0.02%	0.00%
		PMA-SLS	289.30	95.20	148.80	153987.80	0.06%	153890	0.00%	10.00%
		PMA-FLS[2]	119.60	150.80	220.80	154195.80	0.20%	153936	0.03%	0.00%
		PMA	439.00	111.20	144.40	153942.60	0.04%	153890	0.00%	30.00%

Table 5.4 Results of testing on *tai100b* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tai100b	2-island	PMA-DLS	694.70	94.70	124.30	1186119285.20	0.01%	1185996137	0.00%	50.00%
1185996137		PMA-SLS	782.40	106.70	134.00	1186275856.50	0.02%	1185996137	0.00%	40.00%
		PMA-FLS[1]	186.90	175.30	245.30	1188882832.20	0.24%	1186007112	0.00%	0.00%
	4-island	PMA-DLS	633.40	105.00	122.00	1186121434.00	0.01%	1185996137	0.00%	80.00%
		PMA-SLS	647.50	92.60	102.30	1186007361.40	0.00%	1185996137	0.00%	80.00%
		PMA-FLS[2]	178.10	268.80	332.00	1187539521.00	0.13%	1186007112	0.00%	0.00%
	6-island	PMA-DLS	342.40	65.20	107.20	1186132401.50	0.01%	1185996137	0.00%	70.00%
		PMA-SLS	356.60	88.30	104.90	1186058956.40	0.01%	1185996137	0.00%	70.00%
		PMA-FLS[2]	160.10	233.70	296.70	1187892570.00	0.16%	1185996137	0.00%	10.00%
	10-island	PMA-DLS	214.50	87.00	112.80	1186064568.60	0.01%	1185996137	0.00%	80.00%
		PMA-SLS	220.70	68.70	82.70	1186053344.20	0.00%	1185996137	0.00%	80.00%
		PMA-FLS[2]	148.00	250.00	320.00	1187927883.00	0.16%	1186052259	0.00%	0.00%

Table 5.5 Results of testing on *sko100** benchmarks

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100a 152002	2-island	PMA-DLS	852.80	118.80	164.50	152156.10	0.11%	152069	0.03%	0.00%
		PMA-SLS	883.60	133.80	175.10	152188.20	0.12%	152042	0.03%	0.00%
		PMA-FLS[1]	194.00	203.40	273.40	152322.80	0.21%	152122	0.08%	0.00%
	4-island	PMA-DLS	855.30	132.40	171.60	152104.10	0.07%	152059	0.04%	0.00%
		PMA-SLS	885.20	142.40	176.80	152119.00	0.08%	152058	0.04%	0.00%
	6-island	PMA-DLS	416.60	131.60	170.80	152126.20	0.08%	152044	0.03%	0.00%
		PMA-SLS	431.90	138.90	176.90	152109.40	0.07%	152067	0.04%	0.00%
	10-island	PMA-DLS	273.50	134.20	174.20	152093.60	0.06%	152035	0.03%	0.00%
		PMA-SLS	283.20	98.10	156.00	152102.80	0.06%	152042	0.03%	0.00%
sko100c 147862	2-island	PMA-DLS	847.30	120.60	167.90	147928.60	0.05%	147862	0.00%	10.00%
		PMA-SLS	939.30	121.80	168.40	147934.80	0.05%	147862	0.00%	10.00%
		PMA-FLS[1]	184.40	205.80	275.80	148140.40	0.18%	148050	0.13%	0.00%
	4-island	PMA-DLS	826.30	112.20	163.00	147894.00	0.02%	147862	0.00%	20.00%
		PMA-SLS	845.90	111.40	160.50	147908.20	0.03%	147862	0.00%	10.00%
	6-island	PMA-DLS	401.20	124.20	173.00	147887.20	0.02%	147868	0.00%	30.00%
		PMA-SLS	416.80	106.40	151.80	147885.60	0.02%	147862	0.00%	20.00%
	10-island	PMA-DLS	258.40	102.40	125.60	147885.20	0.02%	147862	0.00%	40.00%
		PMA-SLS	284.20	107.90	151.00	147895.40	0.02%	147862	0.00%	10.00%
sko100d 149576	2-island	PMA-DLS	869.60	136.10	170.80	149742.20	0.11%	149656	0.05%	0.00%
		PMA-SLS	883.00	111.00	166.90	149803.60	0.15%	149618	0.03%	0.00%
		PMA-FLS[1]	232.10	259.90	327.40	150036.80	0.31%	149732	0.10%	0.00%
	4-island	PMA-DLS	813.50	137.00	177.20	149729.20	0.10%	149648	0.05%	0.00%
		PMA-SLS	881.20	146.70	180.00	149752.00	0.12%	149630	0.04%	0.00%
	6-island	PMA-DLS	429.00	134.20	168.40	149707.60	0.09%	149620	0.03%	0.00%
		PMA-SLS	436.80	135.80	173.80	149699.40	0.08%	149578	0.00%	0.00%
	10-island	PMA-DLS	279.60	154.60	180.00	149685.20	0.07%	149608	0.02%	0.00%
		PMA-SLS	312.60	120.10	167.20	149681.60	0.07%	149584	0.01%	0.00%
sko100e 149150	2-island	PMA-DLS	809.40	111.30	148.00	149198.20	0.03%	149150	0.00%	30.00%
		PMA-SLS	845.40	121.00	166.70	149205.80	0.04%	149150	0.00%	10.00%
		PMA-FLS[1]	235.50	252.90	322.90	149642.20	0.33%	149198	0.03%	0.00%
	4-island	PMA-DLS	864.80	119.80	173.60	149188.80	0.03%	149150	0.00%	10.00%
		PMA-SLS	898.50	114.30	164.50	149202.60	0.04%	149150	0.00%	10.00%
	6-island	PMA-DLS	425.00	107.80	144.80	149183.60	0.02%	149150	0.00%	40.00%
		PMA-SLS	452.10	113.70	156.90	149179.20	0.02%	149150	0.00%	30.00%
	10-island	PMA-DLS	251.20	61.00	131.00	149180.80	0.02%	149150	0.00%	40.00%
		PMA-SLS	274.20	91.20	130.00	149176.40	0.02%	149150	0.00%	40.00%
sko100f 149036	2-island	PMA-DLS	825.70	98.90	155.20	149218.03	0.12%	149096	0.04%	0.00%
		PMA-SLS	888.40	104.60	153.70	149232.80	0.13%	149126	0.06%	0.00%
		PMA-FLS[1]	206.50	214.80	284.80	149496.60	0.31%	149228	0.13%	0.00%
	4-island	PMA-DLS	813.90	130.20	173.40	149144.20	0.07%	149036	0.00%	20.00%
		PMA-SLS	872.10	126.10	166.70	149150.40	0.08%	149036	0.00%	10.00%
	6-island	PMA-DLS	423.20	151.40	180.00	149145.20	0.07%	149092	0.04%	0.00%
		PMA-SLS	451.70	136.10	172.30	149205.40	0.11%	149078	0.03%	0.00%
	10-island	PMA-DLS	282.40	111.20	158.80	149162.40	0.08%	149104	0.05%	0.00%
		PMA-SLS	300.30	107.00	161.40	149203.40	0.11%	149114	0.05%	0.00%

Table 5.6 Results of testing on *tai100a* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tai100a 21125314	2-island	PMA-DLS	866.20	127.90	161.80	21442193.71	1.50%	21379594	1.18%	0.00%
		PMA-SLS	860.00	127.20	164.60	21458262.60	1.58%	21382118	1.22%	0.00%
		PMA-FLS[1]	222.80	238.50	308.50	21464686.20	1.61%	21335594	1.00%	0.00%
	4-island	PMA-DLS	889.20	156.20	180.00	21380930.80	1.21%	21362016	1.12%	0.00%
		PMA-SLS	889.60	140.20	170.90	21420954.60	1.40%	21352956	1.08%	0.00%
	6-island	PMA-DLS	433.60	146.40	180.00	21368255.10	1.15%	21237278	0.53%	0.00%
		PMA-SLS	451.40	152.50	180.00	21373508.00	1.17%	21270370	0.69%	0.00%
	10-island	PMA-DLS	294.80	120.80	168.40	21347190.00	1.05%	21306288	0.86%	0.00%
		PMA-SLS	309.60	123.60	169.50	21382655.00	1.21%	21295312	0.80%	0.00%

Table 5.7 Results of testing on *wil100* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
wil100 273038	2-island	PMA-DLS	833.90	120.10	167.10	273147.20	0.04%	273078	0.01%	0.00%
		PMA-SLS	882.10	114.50	166.20	273198.80	0.06%	273054	0.01%	0.00%
		PMA-FLS[1]	218.00	226.10	292.80	273458.20	0.15%	273236	0.07%	0.00%
	4-island	PMA-DLS	881.40	137.00	180.00	273101.60	0.02%	273066	0.01%	0.00%
		PMA-SLS	895.20	115.20	165.60	273228.60	0.07%	273054	0.01%	0.00%
	6-island	PMA-DLS	433.40	128.20	178.00	273092.80	0.02%	273056	0.01%	0.00%
		PMA-SLS	445.00	99.30	164.30	273103.80	0.02%	273044	0.00%	0.00%
	10-island	PMA-DLS	272.40	88.20	153.00	273102.60	0.02%	273054	0.01%	0.00%
		PMA-SLS	295.20	93.90	161.20	273128.60	0.03%	273054	0.01%	0.00%

Table 5.8 Results of testing on *tho150* benchmark

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
tho150 8133398	2-island	PMA-DLS	7136.20	139.10	177.00	8146734.00	0.16%	8142504	0.11%	0.00%
		PMA-SLS	7290.30	140.30	174.00	8148332.60	0.18%	8142700	0.11%	0.00%
		PMA-FLS[1]	1428.50	308.50	368.20	8158144.60	0.30%	8140370	0.09%	0.00%
	4-island	PMA-DLS	4993.80	141.60	172.00	8143158.00	0.12%	8137465	0.05%	0.00%
		PMA-SLS	5258.40	148.40	180.00	8144249.60	0.13%	8138428	0.06%	0.00%
		PMA-FLS[2]	935.10	317.70	376.80	8162408.00	0.36%	8145990	0.15%	0.00%
	6-island	PMA-DLS	3027.20	156.80	180.00	8142624.80	0.11%	8137004	0.04%	0.00%
		PMA-SLS	3267.40	156.20	180.00	8145297.20	0.15%	8142554	0.11%	0.00%
		PMA-FLS[2]	885.30	332.50	386.00	8157363.67	0.29%	8151408	0.22%	0.00%
	10-island	PMA-DLS	1953.80	155.40	180.00	8140869.60	0.09%	8139868	0.07%	0.00%
		PMA-SLS	2004.20	136.00	177.20	8144993.20	0.14%	8142102	0.11%	0.00%
		PMA-FLS[2]	605.10	418.50	462.40	8165464.60	0.39%	8154998	0.27%	0.00%

In Table 5.3, from the viewpoint of computational time, compared to the serial MA, much shorter computational time is consumed by PMA-SLS, PMA-DLS, PMA-FLS and PMA, indicating the advantage of employing parallel memetic algorithms. From a solution quality point of view, PMA, PMA-DLS and PMA-SLS can achieve much better quality than PMA-FLS. This is evident from the much improved solution gap and the

higher success rate achieved, which can be attributed to the powerful search capability of memetic algorithm. Furthermore, PMA-DLS and PMA-SLS can reduce the computational time significantly with little or no lost in solution quality compared to PMA which benefited from the more desirable population diversity profile as a result of the diversity-adaptive strategies employed. The comparison among PMA-SLS, PMA-DLS, PMA and PMA-FLS on *sko100b* benchmark is shown in Figure 5.4.

(a) Computation time

(b) Solution quality

Figure 5.4 Comparison among PMA-SLS, PMA-DLS, PMA and PMA-FLS on *sko100b* benchmark

The plot in Figure 5.4 (b) shows that PMA-DLS, PMA-SLS and PMA improve the solution quality significantly compared to PMA-FLS. It is noted that the maximum number of generations for PMA-FLS was set at 500. Instead, the maximum number of generations for PMA-DLS, PMA-SLS and PMA was set to 180. This is indicative of the powerful search capability and quick convergence speed of the PMA. As for the computational time shown in Figure 5.4 (a), the greater reliance on local search makes PMA more time-consuming than the PMA-FLS. However, with the island model paradigm of the parallel memetic algorithm, a distributed computing framework can help to reduce the computational time significantly. Furthermore, the diversity-based adaptive local search strategy both in static and dynamic manner used in PMA-SLS and PMA-DLS, respectively, improves the efficiency of the PMA remarkably.

In addition, it is observed from experimental results that lower quality solutions are obtained with shorter CPU time, and higher quality solutions are obtained with longer CPU time as shown in Figure 5.5 for *sko100b* as an example. The data points on each line from the first data point to the last one denote MA, PMA, PMA-DLS, PMA-SLS and PMA-FLS, respectively. From Figure 5.5, it is also clear that PMA-DLS and PMA-SLS produce solutions that are competitive with that obtained in PMA and MA at significantly less computational cost. Moreover, PMA-DLS and PMA-SLS achieve much higher solution quality than PMA-FLS with little increase in CPU time. The trend is more evident when the number of processors increases.

Figure 5.5 Two-dimensional plot of MA, PMA, PMA-DLS, PMA-SLS and PMA-FLS on *sko100b* benchmark

To determine the significance of the reduced computational time by PMA-DLS, a statistical t -test was used ($p < 0.05$), with the null hypothesis of having no difference between PMA-DLS and PMA. Based on t -test for statistical significance, the mean and the associated results of the one tail difference of two independent means, taken over 10 independent trials for *sko100b* for PMA-DLS and PMA, are presented in Table 5.9.

Table 5.9 Mean and the one tail difference of t -test between PMA-DLS and PMA

	PMA-DLS Mean	PMA Mean	p - PMA-DLS vs. PMA
2-island	859.4	1350	4.17256E-05
4-island	885.7	1445.9	3.96277E-11
6-island	413.8	694.3	1.36381E-05
10-island	276.8	439	3.83569E-05

In Table 5.9, for all cases, the mean of PMA-DLS is smaller than that of PMA and the value $p < 0.05$ indicates that PMA-DLS indeed was able to reduce the computational time compared to PMA with high level of statistical significance. This validates the notion of

PMA-DLS being able to search more efficiently than PMA to achieve a comparable solution quality.

Similar to the *sko100b* benchmark, the effect of multiple islands processing is plotted as in Figure 5.6 for the *tai100b* benchmark. The results show that PMA-DLS and PMA-SLS can achieve much better solution quality with comparable computational time, especially for the case where the number of processors increases to 10 machines. It is also observed that the *tai100b* QAP benchmark shows a much higher *Success rate*, indicating that the PMA-DLS has greater success in locating the global optimum. This implies that the PMA-DLS is capable of locating the best-known solution more frequently than the PMA-FLS.

(a) Computation time

(b) Solution quality

Figure 5.6 Comparison among PMA-DLS, PMA-SLS and PMA-FLS on *tai100b* benchmark

In Tables 5.5 to 5.8, it is observed that PMA-DLS can significantly improve the solution quality with comparable computational time, even for the seemingly difficult class of benchmarks, *tai100a*, especially so when the number of processors increases to 10 machines. In particular, for the very large scale benchmark, *tho150*, PMA-DLS also shows significant improvement of solution quality.

Comparison with Other Results

Based on these results, we compare the performance of our algorithm to recent, high performing algorithms from the literature which shown PMA-DLS outperform several competitors on a number of instances. As shown in Table 5.10, it is noted that the results for PMA-DLS of several instances are much better than that of the MAs and other algorithms developed by different authors. Here, MF1999b, MF2000b, ATO2000, and ZSTF2004 represent the results of Merz and Freisleben (1999b), Merz and Freisleben (2000b), Ahuja *et al.* (2000), and Zhang *et al.* (2004), respectively. The best results are

highlighted in bold. For example, our results on the *tail00a* benchmark problem in Table 5.6 using the PMA-DLS are better than that found in (Merz and Freisleben 2000b). The *Average gap* of 1.089% for *tail00a* was reported in (Merz and Freisleben 2000b), while the *Average gap* of PMA-DLS for *tail00a* on 10 machines is 1.05%. Also the results of *tail00b* for PMA-SLS are much better than that shown in (Merz and Freisleben 1999b). The *Average gap* of *tail00b* was reported as 0.026% in (Merz and Freisleben 1999b), with the *Success rate* being less than 50%. On the other hand, *Average gap* achieved by our PMA-DLS (0.01%) is much better than that in (Merz and Freisleben 1999b), and the *Success rate* is very commendable, being as high as 80%. Furthermore, it is worth nothing that the PMA-DLS is also capable of attaining search quality that is significantly better than that obtained in (Merz and Freisleben 2000b) on the *sko100a* problem. As shown in Table 5.5, on the *sko100a* benchmark, the *Average gap* obtained in (Merz and Freisleben 2000b) was 0.096%, while we were able to reduce this value to 0.06%. As for the very large benchmark, *tho150*, the empirical results in Table 5.8 show that the best solution quality on average, 0.09%, is much better compared to that reported in (Merz and Freisleben 2000b), which was 0.151%. And the results of benchmarks of *sko100** and *wil100* are all better than AOT2000.

Table 5.10 Comparison of PMA-DLS, PMA-SLS with MF1999b, MF2000b, ATO2000, and ZSTF2004

instance	best known	PMA-DLS	PMA-SLS	MF1999b	MF2000b	AOT2000	ZSTF2004
sko100a	152002	0.06%	0.06%	0.169%	0.096%	0.21%	0.066%
sko100b	153890	0.04%	0.06%	NA	NA	0.14%	NA
sko100c	147862	0.02%	0.02%	NA	NA	0.20%	NA
sko100d	149576	0.07%	0.07%	NA	NA	0.17%	NA
sko100e	149150	0.02%	0.02%	NA	NA	0.24%	NA
sko100f	149036	0.07%	0.08%	NA	NA	0.29%	NA
wil100	273038	0.02%	0.02%	NA	NA	0.20%	NA
tail00a	21125314	1.05%	1.17%	1.135%	1.089%	NA	0.779%
tail00b	1185996137	0.01%	0.01%	0.026%	0.038%	NA	0.135%
tho150	8133398	0.09%	0.13%	NA	0.151%	NA	0.091%

Discussion on the Advantages of PMA-DLS

Based on the comparison between PMA-SLS and PMA-DLS, there are three advantages of PMA-DLS which determine the number of individuals undergoing local search based on online dynamic population diversity. First, the number of individuals to be selected for local search is made dynamic and adaptive to online fluctuation of population diversity. Using this diversity-based dynamic adaptation mechanism, it is able to set a high number of individuals for undergoing local search if the population diversity is high. If the population diversity is very low, it is able to decrease the number of candidates undergoing local search to reduce the additional computational effort. In addition, for the island model PMA, the diversity-based dynamic adaptive local search is able to adjust the number of individuals for local search according to the different diversity fluctuation tendency in each island.

Secondly, the PMA-DLS adjusts the local search frequency online, avoiding the laborious task of parameters tuning. Therefore, PMA-DLS is desirable in producing more

robust search performance, resulting in overall improvement in solution quality.

Thirdly, an intrinsic characteristic of PMA-DLS is the Markovian property, in deciding the frequency of applying local search. Eq.(5.2) computes the number of chromosomes that undergo local learning in the current generation based on the previous k generations and the current generation. This property is consistent with the theoretical foundation of various evolutionary algorithms, such as genetic algorithms and memetic algorithms (Suzuki 1995).

1.18 Summary

This chapter proposes a diversity-based dynamic adaptive strategy for controlling the local search frequency of PMA. Instead of fastidious tuning of the parameters setting for the Gaussian function, a diversity-based dynamic adaptive local search strategy is employed in PMA-DLS such that the local search frequency is adaptively adjusted based on the online fluctuation of population diversity measured by population entropy. This diversity-adaptive approach avoids premature convergence resulting from fast decreasing population diversity, as well as reduces the computational effort.

The experimental study verifies that PMA-DLS shows the ability of producing solutions that are competitive with that obtained in PMA at significantly less computational cost for solving large scale QAP. The higher success rate of PMA-DLS also indicates the

improved solution precision due to the intrinsic parallelism and the higher level of diversity maintained during the evolutionary process. Furthermore, PMA-DLS achieves more reliable solutions than PMA-SLS and is more robust, being less sensitive to the parameters setting. Hence, there is not much effort expended on tedious parameters tuning which quite often frustrates the setting-up process for PMA-SLS.

Without doubt, our study will elicit more relevant research work on this topic. The issue on applying different levels of the local search frequency in selecting memes in multi-meme PMA paradigm is meaningful and challenging, deserving further study. The diversity-based adaptive PMA also demonstrates its potential in solving other computationally demanding optimization problems.

Hierarchical Model PMA in Heterogeneous Computing Environment

1.19 Introduction

In earlier chapters, our island model PMA is always implemented in a homogeneous computing environment where all the nodes have the same computing capability in one cluster of workstations. It showed the significant improvement on the PMA performance in terms of both solution quality and computational time. In addition, vast majority of current distributed EAs assume homogenous environments. Those studies assume an ideal situation that all the computing resources have the same computing capability, but in a real system, such an assumption is overly simplistic.

It is worth nothing that distributed computing environment usually consists of heterogeneous computing nodes, in terms of their computational power, operating platform, network connectivity and latency. Yet the vast majority of island model PEAs in existing literature is designed for operation on clusters with homogeneous compute

nodes. Hence, it is necessary to understand how a distributed EA will behave when running on heterogeneous computing nodes in order to utilize them effectively in a distributed computing environment. A recent study in (Gong *et al.* 2005) evaluated a PGA on the line topology of heterogeneous computing resources. The ascending and descending order arrangements of computing capability were investigated on their differences in chromosome variety, migration frequency and solution quality. In addition, Alba *et al.* (2002) investigated the use of a ring topology on heterogeneous compute nodes, each differing in terms of operating platform as well as computational power. He concluded that the added diversity due to differing subpopulation evolution speeds helps the GA reach the global optimum more efficiently.

This chapter presents a study of the proposed island model PMA in a simulated distributed computing environment of heterogeneous configurations of computing nodes. In particular, we will consider PMA-DLS, which has been established as the best performing model among those described in earlier chapters. The performance of PMA-DLS is investigated on several large scale QAPs. Empirical results show that heterogeneous computing nodes have a negative impact on existing PMA-DLS performance. Consequently, a hierarchical model (hPMA-DLS) is proposed to mitigate this effect where the lower level of a master-slave model is deployed under the upper level of island model. The empirical study shows that the hPMA-DLS is capable of improving the algorithm efficiency significantly in the heterogeneous computing environment.

Section 6.2 will first study the influence of heterogeneous computing environment on the performance of our previously proposed PMA-DLS. To mitigate the impact of the heterogeneity of the computing resources, a hierarchical model (hPMA-DLS) is proposed. In Section 6.3, empirical study of hPMA-DLS on several large scale QAPs is conducted in the simulated heterogeneous computing resources in comparison with PMA-DLS. Finally, Section 6.4 summarizes this chapter.

1.20 Hierarchical Model (hPMA-DLS) in Heterogeneous Computing Environment

In this section, we seek to answer the following questions:

- 1) What is the effect of heterogeneous (in terms of computational power) computing nodes in island model PMA-DLS, compared to using homogeneous compute nodes?
- 2) Is there any model modification that may mitigate the impact on the performance of island model PMA in the heterogeneous computing environment?

Here, we study the influence of heterogeneous computing environment on the performance of our proposed PMA-DLS comparing the previous homogeneous computing environment on several large scale QAP benchmarks in terms of both computation time and solution quality. The experimental setup is the same as that in

Chapter 4 (see Section 4.4). For each QAP benchmark problem, we carried out ten optimization runs and the algorithms were evaluated based on their average performance.

Due to the difficulty of using a real grid environment for experimental evaluation, we simulate such an environment using the same cluster of PCs, which were used to evaluate the performance of the proposed island model PMA in the previous chapters. All processors of the cluster have the same computing capability. Our focus is on the heterogeneity of computing power amongst computing nodes in a distributed computing environment. To simplify the experiments, we simulate computing nodes with slower computing power by running several times instances of the subpopulation evolution on a single computing node, in effect reducing the computing capability allocated to each subpopulation evolution. The extra computational steps act as dummy procedures on a node which simulates a slower computing node. For example, if two computing nodes with different computing power represented by $1:1/3$, it means that the latter node is three times slower than the former. Therefore, the former runs at the full speed of a single computing node, while the later has to simulate to run three instances of the subpopulation evolution on a single computing node. We explored configurations with the simulated heterogeneous computing nodes shown in Table 6.1. Note that the significant difference in the computational capabilities by computing nodes.

Table 6.1 Configurations of heterogeneous computing nodes

Number of islands	Configuration
2-island	1:1/3
4-island	1:1/3:1/5:1/6
6-island	1:1/3:1/5:1/6:1/7:1/9

In Table 6.2, the PMA-DLS performance is compared between heterogeneous and homogeneous computing environment on three large scale QAPs. For convenience, PMA-DLS running in heterogeneous computing environment is abbreviated as PMA-DLSa, while PMA-DLS running in homogeneous computing environment is denoted as PMA-DLSb which are the same results in Chapter 5. It is observed that there is no significant difference of solution quality between PMA-DLSa and PMA-DLSb, since the algorithmic behavior is preserved to be the same island model. Nevertheless, as to the PMA-DLSa, due to the heterogeneity of the computing environment considered, the slower computing nodes have become the bottleneck of the island model PMA-DLS since it uses a synchronous migration model which waits for all the subpopulation evolutions to complete before the migration operation and subsequent search generations may proceed. This results in the poorer search efficiency of island model PMA-DLS in the heterogeneous computing environment. The comparisons can be shown more clearly in Figure 6.1 with *sko100b* benchmark as an example.

Table 6.2 Comparison between PMA-DLSa and PMA-DLSb

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b	2-island	PMA-DLSa	2965.90	103.80	159.40	154048.00	0.10%	153920	0.02%	0.00%
		PMA-DLSb	859.40	125.30	169.00	154020.80	0.08%	153920	0.02%	0.00%
153890	4-island	PMA-DLSa	3032.90	120.40	166.20	153980.60	0.06%	153890	0.00%	10.00%
		PMA-DLSb	885.70	131.40	170.00	153977.40	0.06%	153900	0.01%	0.00%
	6-island	PMA-DLSa	2806.80	107.20	155.20	153964.80	0.05%	153940	0.03%	0.00%
		PMA-DLSb	413.80	126.80	164.20	153951.20	0.04%	153890	0.00%	20.00%
273038	2-island	PMA-DLSa	3007.50	133.00	174.60	273175.20	0.05%	273054	0.01%	0.00%
		PMA-DLSb	833.90	120.10	167.10	273147.20	0.04%	273078	0.01%	0.00%
	4-island	PMA-DLSa	3176.10	126.30	174.90	273148.40	0.04%	273044	0.00%	0.00%
		PMA-DLSb	881.40	137.00	180.00	273101.60	0.02%	273066	0.01%	0.00%
	6-island	PMA-DLSa	3058.60	122.80	170.00	273154.40	0.04%	273078	0.01%	0.00%
		PMA-DLSb	433.40	128.20	178.00	273092.80	0.02%	273056	0.01%	0.00%
sko100d	2-island	PMA-DLSa	2992.50	131.70	172.90	149750.60	0.12%	149698	0.08%	0.00%
		PMA-DLSb	869.60	136.10	170.80	149742.20	0.11%	149656	0.05%	0.00%
	4-island	PMA-DLSa	3207.90	132.00	175.90	149730.40	0.10%	149598	0.01%	0.00%
		PMA-DLSb	813.50	137.00	177.20	149729.20	0.10%	149648	0.05%	0.00%
	6-island	PMA-DLSa	3139.60	141.40	180.00	149686.40	0.07%	149662	0.06%	0.00%
		PMA-DLSb	429.00	134.20	168.40	149707.60	0.09%	149620	0.03%	0.00%

(a) Computation time

(b) Solution quality

Figure 6.1 Comparison between PMA-DLSa and PMA-DLSb on *sko100b* benchmark

To deal with the heterogeneity of the computing resources, a simple solution to maintain the benefit of parallelization is to configure the subpopulation size according to the computational capabilities of the computing nodes. Nevertheless, such an approach has the disadvantage of possibly altering the standard behavior of a synchronous island model PMA. Since the search behavior could be unpredictable in such approach, it is advisable to be more conservative by maintaining uniform subpopulation sizes. A preferable solution should still provide speedup regardless of the heterogeneity in the computing environment, while preserving the standard behavior of the island model. To achieve such a solution, we present a hierarchical model (hPMA-DLS) as depicted in Figure 6.2.

Figure 6.2 The hierarchical model (hPMA-DLS)

The core idea of the hierarchical model (hPMA-DLS) lies in maintaining the island model with identical subpopulation sizes at the upper level implemented sequentially (island model), while farming ensembles of chromosomes selected to do local search based on PMA-DLS strategy across the computing nodes available in the cluster through NetSolve at the lower level (master-slave model). Once all the ensembles of local searches are completed, the improved chromosomes are then marshaled back to corresponding island/subpopulation to undergo evolutionary operations involving genetic crossover, mutation, selection, etc. Based on the analysis on the computational complexity of k -gene exchange local search in Section 3.3.3 and preliminary tests we have done in the previous chapters, it is indicated that the local search in our PMA procedure is responsible for the bulk of the total CPU time during the search process, making it the obvious choice for farming in parallel at the lower level. Furthermore, due to the relatively independent execution of local search for each chromosome in PMA procedure, it is reasonable and easy to distribute the chromosomes in ensemble for local search in proportion to the heterogeneous computing power of the nodes available in the cluster, which is the main issue of our focus in the chapter. In the hierarchical model (hPMA-DLS), the slave nodes are solely meant for local search purpose and all subpopulations' island evolutionary operations proceed at the master node. Consequently, we can maintain the uniformity of the subpopulation size at the master side and allocate non-uniform ensembles of chromosomes for local search on the slave nodes. In this way, it would be possible to achieve a more balanced distribution of tasks among

heterogeneous computing nodes, as well as combine two levels of parallelism, which makes a joint exploitation of the hierarchical model.

To minimize the idling time of fast processing nodes while waiting for synchronization in hPMA-DLS, we hope to obtain a well-balanced execution time in each computing node and for each island model search generation. The proposed hierarchical model (hPMA-DLS) is to take advantage of the unbalanced computing power of each node, and assign the ensembles of chromosome for local search by demand, that is, the faster a slave node is, the more chromosomes it will process. In each M -subpopulation (M processing nodes available) hPMA-DLS search generation, each non-uniform ensemble of individuals, $C_i(gen)$, assigned to the i -th processing node for local search at the gen^{th} generation is estimated by:

$$C_i(gen) = \frac{t_i}{\sum_{j=1}^M t_j} \times \sum_{q=1}^M \phi_q(gen), \quad (6.1)$$

where $\phi_q(gen)$ is the number of chromosomes that undergo local learning defined based on online diversity of the q -th subpopulations at the gen^{th} generation as per Eq.(5.2), M is the number of islands (or processing nodes available), and t_i is the simulated configuration of computing power on i -th processing node as shown in Table 6.1.

1.21 Empirical Study

The comparison of performance between hPMA-DLS and PMA-DLSa on three large scale QAPs is shown in Table 6.3. The configurations with the simulated heterogeneous computing nodes are the same as shown in Table 6.1. The experimental setup and parameters setting for hPMA-DLS are the same as that in Chapter 4 (see Section 4.4). The size of the ensembles of individuals are defined using Eq.(6.1) in the hPMA-DLS is shown in Figure 6.3, where one specific run of *sko100b* benchmark is used as an example for 2-island, 4-island and 6-island. It is observed that hPMA-DLS is capable of distributing the chromosomes in ensembles for local search in proportion to the heterogeneous computing power of the nodes available in the cluster efficiently during the evolutionary search process which achieves a more balanced distribution of tasks among the slave nodes.

Table 6.3 Comparison between hPMA-DLS and PMA-DLSa

			CPU time	Generation	TG	Average	Average gap	Best	Gap	Success rate
sko100b	2-island	hPMA-DLS	2000.60	105.70	152.80	154045.00	0.10%	153890	0.00%	10.00%
		PMA-DLSa	2965.90	103.80	159.40	154048.00	0.10%	153920	0.02%	0.00%
	4-island	hPMA-DLS	1872.00	95.50	163.00	153986.00	0.06%	153974	0.05%	0.00%
		PMA-DLSa	3032.90	120.40	166.20	153980.60	0.06%	153890	0.00%	10.00%
	6-island	hPMA-DLS	1481.50	121.90	170.10	153980.56	0.04%	153910	0.01%	0.00%
		PMA-DLSa	2806.80	107.20	155.20	153964.80	0.05%	153940	0.03%	0.00%
wil100	2-island	hPMA-DLS	2153.83	124.17	171.00	273201.82	0.06%	273092	0.02%	0.00%
		PMA-DLSa	3007.50	133.00	174.60	273175.20	0.05%	273054	0.01%	0.00%
	4-island	hPMA-DLS	1823.00	120.33	167.67	273135.33	0.04%	273054	0.01%	0.00%
		PMA-DLSa	3176.10	126.30	174.90	273148.40	0.04%	273044	0.00%	0.00%
	6-island	hPMA-DLS	1564.60	144.60	180.00	273134.80	0.03%	273044	0.00%	0.00%
		PMA-DLSa	3058.60	122.80	170.00	273154.40	0.04%	273078	0.01%	0.00%
sko100d	2-island	hPMA-DLS	2169.83	124.50	172.50	149740.53	0.11%	149692	0.08%	0.00%
		PMA-DLSa	2992.50	131.70	172.90	149750.60	0.12%	149698	0.08%	0.00%
	4-island	hPMA-DLS	1744.33	108.67	168.50	149710.61	0.09%	149638	0.04%	0.00%
		PMA-DLSa	3207.90	132.00	175.90	149730.40	0.10%	149598	0.01%	0.00%
	6-island	hPMA-DLS	1361.20	84.20	141.40	149674.40	0.06%	149610	0.02%	0.00%
		PMA-DLSa	3139.60	141.40	180.00	149686.40	0.07%	149662	0.06%	0.00%

(a) 2-island

(b) 4-island

(c) 6-island

Figure 6.3 The size of ensembles of individuals for hPMA-DLS on *sko100b* benchmark

Most importantly, the results in Figure 6.4 show that the hPMA-DLS can help enhance the efficiency of the island model PMA-DLS search without deterioration in the solution quality in the heterogeneous computing environment. Considering *sko100b* benchmark as an example, Figure 6.4 (a) shows that hPMA-DLS can reduce the computational time significantly, while there is no lost of solution quality observed in Figure 6.4 (b).

(a) Computation time

(b) Solution quality

Figure 6.4 Comparison between hPMA-DLS and PMA-DLSa on *sko100b* benchmark

1.22 Summary

In this chapter, we have explored the behavior of island model PMA-DLS on a heterogeneous configuration consisting of nodes with different computational power. It is shown that heterogeneous computing nodes can degrade the performance of island model

PMA-DLS. Therefore, a hierarchical model PMA-DLS (hPMA-DLS) was proposed, which includes the island model (the upper level) and master-slave model (the lower level). It achieves a more balanced distribution of tasks among heterogeneous computing nodes, as well as combines two levels of parallelism, which makes a joint exploitation of the hierarchical model. The empirical study on several large scale QAPs shows that hPMA-DLS can enhance the efficiency of the island model PMA-DLS search without deterioration in the solution quality.

As future work, we need to investigate the behavior of hPMA-DLS in a real grid environment to gain further insight on its validity.

Conclusions and Future Work

In this Chapter, a summary of the merits and the limitations of the work conducted is offered and areas for future research are suggested to conclude this dissertation.

1.23 Conclusions

A variety of heuristic approaches are considered vital in algorithms capable of providing good solutions in reasonable time as reviewed in Chapter 2. Although significant development of heuristics for various combinatorial optimization problems has been achieved, the extremely expensive computational overhead to tackle large scale problems is still a challenge in its own right. The solution of these problems requires both improvement in mathematical programming algorithms and the utilization of powerful computational platforms. The studies in this dissertation concentrated on a particular metaheuristics — MAs. The combination of efficient neighborhood search strategies and evolutionary algorithms has been shown to be highly effective for many combinatorial optimization problems. While the neighborhood search is responsible for finding local optimum solutions in small regions of the search space, the evolutionary

component is responsible for exploring new promising regions. At the same time, it is noted that computing technology will allow for further advancement in heuristic algorithms by making available computational horsepower unseen before in the past. Nowadays, distributed computing is an up-and-coming computing technology which has caught much attention of researchers with its novel approach of organizing and accessing computational resources. Based on the intrinsic parallelism of EAs, in this dissertation, we developed an island model parallel memetic algorithm (PMA) which combines evolutionary algorithms, local search, parallel and distributed computing technologies for large scale combinatorial optimization.

Firstly, we developed the island model PMA which is shown to significantly reduce the processing time on solving large scale QAPs using networked clusters of computers. To verify the performance of the PMA-FLS, we conducted empirical study on several large scale QAP benchmark problems and compare it with its sequential counterpart, SMA-FLS. Furthermore, the number of processing nodes was varied to investigate on the scalability of the island model PMA-FLS. Performance analysis of the island model PMA-FLS is measured both in terms of the solution quality and the improvement in execution time. According to the results, it can be concluded that the island model PMA-FLS is more powerful for handling large size QAPs. The experimental results show that substantial speedup can be achieved with the island model PMA-FLS, especially for the larger size benchmarks, in particular for $n \geq 100$.

Secondly, the impact of certain migration parameters, such as migration topology, on the performance of PMA has also been analyzed. We investigated the effect of varying the migration topology on the performance of the PMA-FLS. Two topologies, one-way ring topology and random topology were studied. The empirical results of PMA-FLS-ring and PMA-FLS-rand were evaluated both in terms of computational time and solution quality. The comparison in terms of scalability for both algorithms was investigated as well. It shows that the solution quality found by PMA-FLS-ring is generally better than the PMA-FLS-rand. The PMA-FLS-ring is more effective in maintaining genetic diversity to prevent the premature convergence. This is especially obvious with the increase in the number of islands. In addition, the scalability of the PMA-FLS-ring is much better than the PMA-FLS-rand. From the results, it can be concluded that the PMA with ring topology maintains a more appropriate tradeoff between exploration and exploitation and hence more helpful in improving the performance of PMA for large scale QAPs.

Thirdly, we proposed and experimentally validated the island model parallel memetic algorithm with diversity-based static adaptive local search strategy (PMA-SLS) for several large scale QAPs. The empirical results were evaluated both in terms of solution quality and computational time, and a comprehensive comparative study with PMA-SLS, PMA-FLS, PMA and some recent important results of MA available in literature. According to the results, it is shown that PMA-SLS arrives at solutions that are competitive to the PMA at significantly lower computation efforts on a diverse set of large scale QAPs considered, with little or no loss of solution quality. This we concluded

that the higher success rate of PMA-SLS coupled with improved solution precision is due mainly to the intrinsic parallelism and the ability of the PMA-SLS to manage a more desirable diversity profile during the evolutionary process of the PMA-SLS. However, the issue on the effect of other parameters pertaining to the selective local search strategy on PMA-SLS's performance need to be explored, in order to achieve more optimal parameters setup for further improvement in performance.

Fourthly, we proposed a diversity-based dynamic adaptive strategy for controlling the local search frequency of PMA. Instead of fastidious tuning of the parameters setting for the Gaussian Function in PMA-SLS, a diversity-based dynamic adaptive local search strategy is employed in PMA-DLS such that the local search frequency is adaptively adjusted based on the online fluctuation of population diversity measured by population entropy. This diversity-adaptive approach avoids premature convergence resulting from fast decreasing population diversity, as well as reduces the computational effort.

The experimental study verifies that PMA-DLS's capability of producing solutions that are competitive with that obtained in PMA at significantly less computational cost for solving large scale QAP. The higher success rate of PMA-DLS also indicates the improved solution precision due to the intrinsic parallelism and the higher level of diversity maintained during the evolutionary process. Furthermore, PMA-DLS achieves more reliable solutions than PMA-SLS and is more robust, being less sensitive to the parameters setting. Hence, there is not much effort expended on tedious parameters tuning which quite often frustrates the setting-up process for PMA-SLS.

Finally, we explored the behavior of island model PMA-DLS on a heterogeneous configuration consisting of nodes with different computational power. It is shown that heterogeneous computing nodes can degrade the performance of island model PMA-DLS. Therefore, a hierarchical model PMA-DLS (hPMA-DLS) was proposed, which includes island model (the upper level) and master-slave model (the lower level). It achieves a more balanced distribution of tasks among heterogeneous computing nodes, as well as combines two levels of parallelism, which makes a joint exploitation of the hierarchical model. The empirical study on several large scale QAPs shows that hPMA-DLS can enhance the efficiency of the island model PMA-DLS search without deterioration in the solution quality.

1.24 Future Work

The results obtained have opened up several issues for future research. There are several avenues of research that can be followed from this dissertation.

From the parallel model point of view, while the results reported in this dissertation focused on the island model parallel MA, there is room to explore the applicability of employing other kinds of parallel EA models, even more complex hierarchical models. As for the island model, the issue on the effect of other control parameters of migration progress on the performance of island model PMA need to be explored to achieve more

optimal parameters setup for the migration operation. For example, the migration policies with different emigrant/replacement mechanisms should be more investigated. Instead of choosing the best chromosome which we use in the thesis, we could apply other selection methods (proportional selection, tournament, random, etc.). Also, we could choose between sending out a copy of the selected individual (we do so in the thesis) or the individual itself. Furthermore, we may devise novel adaptive migration policies in which the choice of what individuals to migrate and/or replace is not defined *a priori* but according to a more knowledge-oriented perspective.

With regard to the design of memetic algorithms, although k -gene exchange local search is utilized in our proposed memetic algorithm, other heuristics for local search purpose can be explored to enhance the performance of parallel MA. Furthermore, extending the memetic algorithms paradigms by including not just one local searcher but a family of them, i.e., multi-meme memetic algorithms, is also a very promising direction.

Without doubt, this dissertation will elicit more relevant research work on striking balance between global and local search in PMA. From literature survey, there has not been much work that considered balancing global and local search in the context of parallel MA. In particular, there is very little focus in literature on several key issues of local search which have significant impact on the performance of PMAs. It is definitely one of the research lines of the future work. In this work, we utilized population diversity to adapt the local search frequency. Furthermore, it is possible that other performance profile could be used to track the progress of the search, especially the various

algorithmic components of the memetic algorithm. That is, one might want to take into consideration the reuse of knowledge gained about the search landscape(s) the algorithm is exploring throughout the optimization process. The issue on applying different levels of the local search frequency in selecting memes in multi-meme PMA paradigm is meaningful and challenging, deserving further study.

As for the deployment of parallel evolutionary algorithm in a distributed computing environment, investigation of the behavior of hPMA-DLS in a real grid environment is useful to gain further insight on its validity. Further work on the parallel evolutionary algorithm models is necessary in order to allow their adaptation to grids taking into account the characteristics of such execution environments in terms of volatility, heterogeneity, large scale and multi-administrative domain.

The successful application of the island model parallel MA within a distributed computing environment in solving several large scale QAPs demonstrates its potential in solving other computationally demanding optimization problems. In our opinion the potential of future algorithmic progresses for the QAPs and other difficult optimization problems can be achieved by a marriage between the island model parallel memetic algorithm paradigm and the power that novel computational platforms have to offer.

Author's Publications

1.25 Journals

1. Tang J., Lim M. H. and Ong Y. S., "Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinatorial Optimization Problems," *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, online published on October 18, 2006. DOI: 10.1007/s00500-006-0139-6.
2. Tang J., Lim M. H., Ong Y. S. and Er M. J. (2006) "Parallel Memetic Algorithm with Selective Local Search for Large Scale Quadratic Assignment Problems," *International Journal of Innovative Computing, Information and Control*, vol. 2, no. 6, pp. 1399-1416, ISSN: 1349-4198.

1.26 Conferences

1. Tang J., Lim M. H., Ong Y. S. and L.Q. Song, "Hierarchical Model Parallel Memetic Algorithm in Heterogeneous Computing Environment," to appear:

- 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, September 25-28, 2007, Singapore.
2. Tang J., Lim M. H. and Ong Y. S., "Design Issues of Island Model Parallel Memetic Algorithms", to appear: *International Symposium on INnovations in Intelligent SysTems and Applications (INISTA 2007)*, Istanbul, Turkey.
 3. Tang J., Lim M. H. and Ong Y. S., "Adaptation for Parallel Memetic Algorithm Based on Population Entropy," *Genetic and Evolutionary Computation Conference (GECCO'06)*, July 8-12, 2006, Seattle, Washington, USA.
"Nominated for Best Paper Award".
 4. Xu Y. L., Lim M. H., Ong Y. S. and Tang J., "A GA-ACO-Local Search Hybrid Algorithm for Solving Quadratic Assignment Problem," *Genetic and Evolutionary Computation Conference (GECCO'06)*, July 8-12, 2006, Seattle, Washington, USA.
 5. Tang J., Lim M. H., Ong Y. S. and Er M. J., "Solving Large Scale Combinatorial Optimization Using PMA-SLS," *2005 Genetic and Evolutionary Computation Conference (GECCO'05)*, Program Track on Evolutionary Combinatorial Optimization, June 25-29, 2005, Washington D.C., USA.
 6. Tang J., Lim M. H., Ong Y. S. and Er M. J., "Study of Migration Topology in Island Model Parallel Hybrid-GA for Large Scale Quadratic Assignment Problems," *Eighth International Conference on Control, Automation, Robotics*

and Vision, ICARCV 2004, Special Session on Computational Intelligence on the Grid, December 6-9, 2004, Kunming, China.

7. Tang J., Lim M. H. and Ong Y. S., "Island Model Parallel Hybrid-GA for Large Scale Combinatorial Optimization," *2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Special Session on Optimization using Genetic, Evolutionary, Social and Behavioral Algorithms, December 15-18, 2003, Singapore.
8. Tang J., Lim M. H. and Ong Y. S., "A Parallel Hybrid GA for Combinatorial Optimization Using Grid Technology," *2003 Congress on Evolutionary Computation (CEC'03)*, December 8-12, 2003, Canberra, Australia, Vol. 3, pp. 1895-1902, 2003.

Bibliography

- Aarts E., Laarhoven P. van, Lenstra J., and Ulder N. (1994) "A computational study of local search algorithms for job-shop scheduling," *ORSA Journal on Computing*, vol. 6, pp. 118-125.
- Aarts E., Korst J. H. M., and Laarhoven P. van. (1997) "Simulated annealing," In: *Local Search in Combinatorial Optimization*, Aarts E. and Lenstra J. K. (Eds.), John Wiley and Sons, pp.91-120.
- Abbass H. A. (2001) "A memetic pareto evolutionary approach to artificial neural networks," In: *The Australian Joint Conference on Artificial Intelligence*, Adelaide, Australia, Springer. Lecture Notes in Artificial Intelligence, vol. 2256, pp. 1-12.
- Ahuja R. K., Orlin J. B., and Tiwari A. (2000) "A descent genetic algorithm for the quadratic assignment problem," *Computers & Operations Research*, vol. 27, no. 10, pp. 917-934.
- Alba E. (2005) *Parallel metaheuristics: A new class of algorithms*. John Wiley.
- Alba E. and Luque G. (2004) "Growth curves and takeover time in evolutionary algorithms," In: *Genetic and Evolutionary Computation Conference (GECCO-2004)*, Deb K. *et al.* (Eds.), pp. 864-876, Seattle, Washington.
- Alba E. and Luque G. (2005) "Theoretical models of selection pressure for dEAs:

- Topology influence,” In: *2005 IEEE Congress on Evolutionary Computation (CEC-2005)*, pp. 214-221.
- Alba E. and Tomassini M. (2002) “Parallelism and evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp 443-462.
- Alba E. and Troya J. M. (2000) “Influence of the migration policy in parallel GAs with structured and panmictic populations,” *Applied Intelligence*, vol. 12, no. 3, pp. 163-181.
- Alba E., Nebro A. J., and Troya J. M. (2002) “Heterogeneous computing and parallel genetic algorithms,” *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1362-1385.
- Alba E., Dorronsoro B., and Alfonso H. (2005) “Cellular memetic algorithm,” *Journal of Computer Science and Technology*, vol. 5, no. 4, pp. 257-263.
- Anstreicher K. M., Brixius N. W., Goux J.-P., and Linderoth J. (2002) “Solving large quadratic assignment problems on computational grids,” *Mathematical Programming*, vol. 91, no. 3, pp. 563-588.
- Armour G. C. and Buffa E. S. (1963) “A heuristic algorithm and simulation approach to the relative location of facilities,” *Management Science*, vol. 9, no. 2, pp. 294-309.
- Bachelet V., Preux P., and Talbi E.-G. (1996) “Parallel hybrid metaheuristics: application to the quadratic assignment problem,” In: *Proceedings of the Parallel Optimization Colloquium*, Versailles, France.
- Back T., Fogel D. B., and Michalewicz Z. (Eds.) (1997) *Handbook of Evolutionary Computation*. Institute of Physics Publishing Ltd, Bristol, Philadelphia, New York,

Oxford University Press, UK.

Baldwin J. (1896) "A new factor in evolution," *American Naturalist*, vol. 30, pp. 441-451.

Bambha N. K., Bhattacharyya S. S., Teich J., and Zitzler E. (2004) "Systematic integration of parameterized local search into evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 137-155.

Battiti R. and Tecchiolli G. (1994) "The reactive tabu search," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 126-140.

Binato S., Hery W. J., Loewenstern D., and Resende M. G. C. (2001) "A greedy randomized adaptive search procedure for job shop scheduling," In: *Essays and Surveys on Metaheuristics*, Hansen P. and Ribeiro C.C. (Eds), Kluwer Academic Publishers.

Blum C. and Roli A. (2003) "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308.

Bradwell R. and Brown K. (1999) "Parallel asynchronous memetic algorithms," In: *Proceedings of the Genetic and Evolutionary Computation Conference*, Evolutionary Computation and Parallel Processing Workshop, Orlando, Florida.

Braun H. C. (1990) "On solving traveling salesman problems by genetic algorithms," In: *Parallel Problem Solving from Nature*, Schwefel H.P. and Manner R. (Eds.), Springer-Verlag, Berlin, Germany, pp. 129-133.

Brimberg J., Hansen P., Mladenovic N., and Taillard E. (2000) "Improvements and comparison of heuristics for solving the multisource weber problem," *Operations*

- Research*, vol. 48, no. 3, pp. 444-460.
- Brixius N. W. (2000) *Solving Large-Scale Quadratic Assignment Problems*. Ph.D. Thesis, the University of Iowa.
- Brown E. D., Huntley L. C., and Spillance R. A. (1989) "A parallel genetic heuristic for the quadratic assignment problem," In: *Proceedings of the 3rd Conference on Genetic Algorithms*, Schaffer J. D. (Ed.), Morgan Kaufmann, pp. 406-415.
- Bui T. G. and Moon B. R. (1994) "A new genetic approach for the traveling salesman problem," In: *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 7-12.
- Bungartz H. and Trajkovski I. (2002) "Efficient strategies for optimization with genetic algorithms," *Selcuk Journal of Applied mathematics*, vol. 3, no. 2, pp. 3-22.
- Burkard R.E. and Derigs U. (1980) "Assignment and matching problems: solution methods with FORTRAN-programs," *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag: Berlin.
- Burkard R. E. and Rendl F. (1984) "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operational Research*, vol. 17, pp. 169-174.
- Burkard R. E., Karisch S. E., and Rendl F. (1997) "QAPLIB—A quadratic assignment problem library," *Journal of Global Optimization*, vol. 10, pp. 391-403. Available from <<http://www.opt.math.tu-graz.ac.at/qaplib/>>
- Burkard R., Cela E., Pardalos P., and Pitsoulis L. (1998) "The quadratic assignment problem," In: *Handbook of Combinatorial Optimization*, Pardalos P. and Du D. Z.

- (Eds.), Kluwer, Boston, pp. 241-338.
- Burke E. K. and Landa Silva J. D. (2004) "The design of memetic algorithms for scheduling and timetabling problems," In: *Recent Advances in Memetic Algorithms*, Krasnogor N. *et al.* (Eds), Springer, pp. 289-311.
- Burke E. K. and Smith A. (1999) "A memetic algorithm to schedule planned maintenance for the national grid," *ACM Journal of Experimental Algorithms*, vol. 4, no. 1, pp. 1084-1096.
- Burke E. K. and Smith A. (2000) "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 122-128.
- Caar B., Hart W., Krasnogor N., Burke E., Hirst J., and Smith J. (2002) "Alignment of protein structures with a memetic evolutionary algorithm," In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, Morgan Kaufmann.
- Cantu-Paz E. (1998) "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2. pp. 141-171, Paris: Hermes.
- Cantu-Paz E. (1999a) "Migration policies and takeover times in genetic algorithms," In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E. (Eds.), San Francisco, CA: Morgan Kaufmann, pp. 775.
- Cantu-Paz E. (1999b) "Topologies, migration rates, and multi-population parallel genetic algorithms," *IlliGAL Report*, no. 99007.

Cantu-Paz E. (1999c) *Designing Efficient and Accurate Parallel Genetic Algorithms*.

Ph.D. Thesis, University of Illinois at Urbana-Champaign.

Cantu-Paz E. (2000a) *Efficient and Accurate Parallel Genetic Algorithms*, Norwell, MA: Kluwer.

Cantu-Paz E. (2000b) "On the effects of migration on the fitness distribution of parallel evolutionary algorithms," In: *Workshop on Evolutionary Computation and Parallel Processing at the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pp. 3-6.

Cantu-Paz E. (2001) "Migration policies, selection pressure, and parallel evolutionary algorithms," *Journal of Heuristics*, vol. 7, no. 4, pp. 311-334.

Cantu-Paz E. and Goldberg D. E. (1999) "On the scalability of parallel genetic algorithms," *Evolutionary Computation*, vol. 7, no. 4, pp. 429-449.

Cantu-Paz E. and Goldberg D. E. (2000) "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, no.186 (2-4), pp. 221-238, New York: Elsevier.

Carrizo F. T. J. and Moscato P. (1992) "A computational ecology for the quadratic assignment problem," In: *Proceedings of the 21st Meeting on Informatics and Operations Research*, (Buenos Aires), SADIO.

Casanova H., and Dongarra J. (1998) "NetSolve: A network enabled server, examples and users," In: *Proceedings of the Heterogeneous Computing Workshop*, Orlando, Florida.

Catlett C. and Smarr L. (1992) "Metacomputing," *Communications of the ACM*, vol. 35,

pp. 44-52.

Cejchan P. (1998) *Personal Communication*.

Cela E. (1998) *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Chakrapani J. and Skorin-Kapov J. (1992) "A connectionist approach to the quadratic assignment problem," *Computers & Operations Research*, vol. 19, no. 3-4, pp. 287-295.

Cheng R. and Gen M. (1997) "Parallel machine scheduling problems using memetic algorithms," *Computers in Engineering*, vol. 33, no. 3-4, pp. 761-764.

Coello C. A. C. (2000) "An updated survey of GA-based multiobjective optimization techniques," *ACM Computing Surveys*, vol. 32, no. 2, pp. 109-143.

Connolly D. T. (1990) "An improved annealing scheme for the QAP," *European Journal of Operational Research*, vol. 46, pp. 93-100.

Cotta C., Mendes A., Garcia V., Franca P., Moscato P. (2003) "Applying memetic algorithms to the analysis of microarray data," *Application of Evolutionary Computing*, Raidl G. (Eds.), Lecture Notes in Computer Science, Springer-Verlag, vol. 2611, pp. 22-32.

Croes G. A. (1958) "A method for solving traveling salesman problems," *Operations Research*, vol. 5, pp. 791-812.

Culberson J. (1998) "On the futility of blind search: an algorithmic view of "no free lunch"," *Evolutionary Computation*, vol. 6, no. 2, pp. 109-128.

Cung V. D., Mautor T., Michelon P., and Tavares A. (1997) "A scatter search based

- approach for the quadratic assignment problem," In: *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC)*, Baeck T., Michalewicz Z., and Yao X. (Eds.), Indianapolis, USA, IEEE Press, pp. 165-170.
- Davidor Y. and Ben-Kiki O. (1992) "The interplay among the genetic algorithm operators: Information theory tools used in a holistic way," In: *Parallel Problem Solving From Nature II*, Manner R. and Manderick B. (Eds.), pp. 75-84. Amsterdam, Elsevier Science Publishers B.V.
- Davis L. (1991) *Handbook of Genetic Algorithms*. New York: Van Nostrand.
- Dawkins R. (1976) *The Selfish Gene*. Oxford Univ. Press, Oxford, UK.
- Deb K. and Goel T. (2001) "A hybrid multi-objective evolutionary approach to engineering shape design," In: *Proceedings of 1st International Conference on Evolutionary Multi-Criterion Optimization*, pp. 385-399, Zurich, Switzerland.
- Deuck G. (1993) "New optimization heuristics," *Journal Computational Physics*, vol. 104, pp. 86-92.
- Deuck G. and Scheuer T. (1990) "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, pp. 161-175.
- Dickey J. W. and Hopkins J. W. (1972) "Campus building arrangement using TOPAZ," *Transportation Research*, vol. 6, pp. 59-68.
- Digalakis J. and Margaritis K. (2001) "A parallel memetic algorithm for solving optimization problems," In: *Proceedings of 4th Meta-heuristics International Conference*, July 16-20, 2001 Porto-Portugal, vol. I, pp 121-125.

- Digalakis J. and Margaritis K. (2004) "Performance comparison of memetic algorithms," *Journal of Applied Mathematics and Computation*, Elsevier Science, vol. 158, no. 25, pp. 237-252.
- Dorigo M., Maniezzo V., and Colorni A. (1991) "Positive feedback as a search strategy," *Technical Report*, Politecnico di Milano, Italy.
- Dorigo M., Maniezzo V., and Colorni A. (1996) "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 26, no. 1, pp. 29–41.
- Drezner Z. (2003) "A new genetic algorithm for the quadratic assignment problem," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 320-330.
- Eiben A. E., Hinterding R., and Michalewicz Z. (1999) "Parameter control in evolutionary algorithm," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 124-141.
- Elshafei A. N. (1977) "Hospital layout as quadratic assignment problem," *Operational Research Quarterly*, vol. 28, pp. 167-179.
- Feo T. A. and Resende M. G. C. (1995) "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109-133.
- Festa P. and Resende M. G. C. (2002) "GRASP: An annotated bibliography," In: *Essays and Surveys on Metaheuristics*, Ribeiro C.C. and Hansen P. (Eds.), Kluwer Academic Publishers, pp. 325-367.
- Fleurent C. and Ferland J. A. (1994) "Genetic hybrids for the quadratic assignment problem," In: *Quadratic Assignment and Related Problems*, Pardalos, P. &

- Wolkowicz, H. (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 173-187.
- Flood, M. M. (1956) "The traveling-salesman problem," *Operations Research*, vol. 4, pp. 61-75.
- Fogel L. J. (1962) "Toward inductive inference automata," In: *Proceedings of the International Federation for Information processing Congress*, Munich, pp. 395-399.
- Fogel G. B., Porto V. W., Weekes D. G., Griffey R. H., McNeil J. A., Lesnik E., Ecker D. J., and Sampath R. (2002) "Discovery of RNA structural elements using evolutionary computation," *Nucleic Acids Research*, vol. 30, no. 23, pp. 5310-5317.
- Foster I. and Kesselman C. (1999) *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco.
- Freisleben B. and Merz P. (1996) "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," In: *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, Back T., Kitano H., and Michalewicz Z. (Eds.), IEEE Press, pp. 616-621.
- Gambardella L., Taillard E., and Dorigo M. (1999) "Ant colonies for the QAP," *Journal of the Operations Research Society*, vol. 50, pp. 167-176.
- Garcia V. J., Franca P. M., Mendes A. D. S., and Moscato P. (2006) "A parallel memetic algorithm applied to the total tardiness machine scheduling problem," In: *The 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*.
- Garey M. R. and Johnson D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

- Geoffrion A. M. and Graves G. W. (1976) "Scheduling parallel production lines with changeover costs: Practical application of a Quadratic Assignment/LP approach," *Operations Research*, vol. 24, pp. 595-610.
- Gero J. S., Kazakov V. A., and Schnier T. (1997) "Genetic engineering and design problems," *Evolutionary Algorithms in Engineering Applications*, Dasgupta D. and Michalewicz Z. (Eds.), Springer, pp. 47-68.
- Glover F. W. (1986) "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol. 13, no. 5, pp. 533-549.
- Glover F. W. and Laguna M. (1997) *Tabu Search*, Kluwer Academic Publishers.
- Glover F. W., Laguna M., and Marti R. (2000) "Fundamentals of scatter search and path relinking," *Control and Cybernetics*, vol. 29, no. 3, pp. 653-684.
- Glover F. W. and Kochenberger G. A. (Eds.). (2003) *Handbook of Metaheuristics*, vol. 57 of International series in operations research and management science. Kluwer Academic Publishers, Boston Hardbound.
- Goldberg D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.
- Goldberg D. and Voessner S. (1999) "Optimizing global-local search hybrids," In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, Banzhaf W., Daida J., Eiben A., Garzon M., Honavar V., Jakaiela M., and Smith R. (Eds.), Morgan Kaufmann, pp. 220-228.
- Gong Y., Nakamura M., and Tamaki S. (2005) "Parallel genetic algorithms on line topology of heterogeneous computing resources," In: *Proceedings of the 2005*

- Conference on Genetic and Evolutionary Computation (GECCO-2005)*, Beyer H. (Ed.), Washington DC, USA, ACM Press, New York, NY, pp. 1447-1454.
- Gorges-Schleuter M. (1989) "ASPARAGOS: An asynchronous parallel genetic optimization strategy," In: *Proceedings of the 3rd International Conference on Genetic Algorithms*, Schaffer J. D. (Ed.), Morgan Kaufmann, pp. 422-427.
- Guimaraes F. G., Wanner E. F., Campelo F., Takahashi R. H. C., Igarashi H., Lowther D. A., and Ramirez J. A. (2006) "Local learning and search in memetic algorithms," In: *2006 IEEE Congress on Evolutionary Computation (CEC-2006)*, Vancouver, BC, Canada, pp. 2936-2943.
- Hansen P. and Mladenovic N. (1998) "Variable neighborhood search for the p -median," *Location Sciences*, vol. 5, no. 4, pp. 207-226.
- Hansen P. and Mladenovic N. (2001) "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449-467.
- Hart W. E. (1994) *Adaptive global optimization with local search*. Ph.D. Thesis, University of California, San Diego.
- Hart W. E. (2001) "A convergence analysis of unconstrained and bound constrained evolutionary pattern search," *Evolutionary Computation*, vol. 9, no. 1, pp. 1-23.
- He L. and Mort N. (2000) "Hybrid genetic algorithms for telecommunications network back-up routing," *BT Technology Journal*, vol. 18, no. 4, pp. 42-56.
- Hertz A., Taillard E., and Werra Dominique de. (1997) "Tabu search," In: *Local Search in Combinatorial Optimization*, Aarts E. and Lenstra J. K. (Eds.), John Wiley and Sons,

pp. 121-136.

Holland J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Harbor, MI.

Holstein D. and Moscato P. (1999) "Memetic algorithms using guided local search: A case study," In: *New Ideas in Optimization*, Corne D., Dorigo M., and Glover F. (Eds.), McGraw Hill, Berkshire, England, pp. 235–243.

Houck C., Joines J., and Kay M. (1996) "Utilizing Lamarckian Evolution and the Baldwin Effect in Hybrid Genetic Algorithms," *NCSU-IE Technical Report 96-01*, Meta-Heuristic Research and Applications Group, Department of Industrial Engineering, North Carolina State University.

Hubert L.J. (1987) "Assignment methods in combinatorial data analysis," *Statistics: Textbooks and Monographs*, New York: Dekker.

Ishibuchi H. and Murata T. (1996) "Multi-objective genetic local search algorithm," In: *Proceedings of the 1996 International Conference on Evolutionary Computation*, Fukuda T. and Furuhashi T. (Eds.), IEEE, Nagoya, Japan, pp. 119-124.

Ishibuchi H. and Murata T. (1998) "Multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 392-403.

Ishibuchi H., Murata T., and Tomioka S. (1997) "Effectiveness of genetic local search algorithms," In: *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 505-512.

Ishibuchi H. and Shibata Y. (2003) "An empirical study on the effect of mating restriction

- on the search ability of EMO algorithms,” In: *Lecture Notes in Computer Science* vol. 2632: Evolutionary Multi-Criterion Optimization, pp. 433-477, Springer, Berlin.
- Ishibuchi H. and Yoshida T. (2002) “Hybrid evolutionary multi-objective optimization algorithms,” In: *Soft Computing Systems: Design, Management and Applications* (Frontiers in Artificial Intelligence and Applications, vol. 87), Abraham A., Ruiz del Solar J., and Koppen M. (Eds), pp. 163-172.
- Ishibuchi H., Yoshida T. and Murata T. (2002) “Selection of initial solutions for local search in multiobjective genetic local search,” In: *Congress on Evolutionary Computation (CEC-2002)*, vol. 1, pp. 950-955, Piscataway, New Jersey, IEEE Service Center.
- Ishibuchi H., Yoshida T., and Murata T. (2003) “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204-223.
- Ishii S. and Sato M. (1998) “Constrained neural approaches to quadratic assignment problems,” *Neural Networks*, vol. 11, pp. 1073-1082.
- Jaszkiewicz A. (1998) “Genetic local search for multiple objective combinatorial optimization,” *Technical Report RA-014/98*, Institute of Computing Science, Poznan University of Technology.
- Jaszkiewicz A. (2002) “Genetic local search for multi-objective combinatorial optimization,” *European Journal of Operational Research*, vol. 137, no. 1, pp. 50-71.
- Jaszkiewicz A. (2003) “Do multiple-objective metaheuristics deliver on their promises?”

- A computational experiment on the set-covering problem,” *IEEE transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 133-143.
- Johnson D. S. and Papadimitriou C. H. (1985) “Computational complexity,” In: *The Traveling Salesman Problem*, Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., and Shmoys D. B. (Eds.), John Wiley & Sons, pp. 37-85.
- Jones T. (1995) *Evolutionary Algorithms, Fitness Landscapes and Search*. Ph.D. thesis, University of New Mexico, Albuquerque, NM.
- Kallel L., Naudts B., and Reeves C. (2001) “Properties of fitness functions and search landscapes,” In: *Theoretical Aspects of Evolutionary Computing*, Kallel L., Naudts B. and Rogers A. (Eds.), Springer, Berlin, Heidelberg, New York.
- Kilby P., Prosser P., and Shaw P. (1999) “Guided local search for the vehicle routing problem with time windows,” In: *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Vob S., Martello S., Osman I., and Roucairol C. (Eds.), Kluwer Academic, pp. 473-486.
- Kirkpatrick S., Gelatt C. D., and Vecchi M. P. (1983) “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671–680.
- Knowles J. (2002) *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. Ph.D. Thesis, University of Reading.
- Knowles J. and Corne D. (2000) “M-PAES: A memetic algorithm for multiobjective optimization,” In: *2000 Congress on Evolutionary Computation*, Piscataway, New Jersey, IEEE Service Center, vol. 1, pp. 325-332.
- Knowles J. and Corne D. (2002) “Towards landscape analyses to inform the design of a

- hybrid local search for the multiobjective quadratic assignment problem,” In: *Soft Computing Systems: Design, Management and Applications*, pp. 271-279, IOS Press.
- Knowles J. and Corne D. (2004) “Memetic algorithms for multiobjective optimization: issues, methods and prospects,” In: *Recent Advances in Memetic Algorithms*, Krasnogor N. *et al.* (Eds), Springer, pp. 313-352.
- Kolen A. and Pesch E. (1994) “Genetic local search in combinatorial optimization,” *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, vol. 48, pp. 273-284.
- Koopmans T. C. and Beckmann M. J. (1957) “Assignment problems and the location of economic activities,” *Econometrica*, vol. 25, pp. 53-76.
- Krarup J. and Pruzan P. M. (1978) “Computer-aided layout design,” *Math. Programming Study*, vol. 9, pp. 85-94.
- Krasnogor N. (2002) *Studies on the Theory and Design Space of Memetic Algorithms*. Ph.D. Thesis, University of the West of England, Bristol, U.K.
- Krasnogor N. (2004a) “Self-generating metaheuristics in bioinformatics: The protein structure comparison case,” In: *Genetic Programming and Evolvable Machines*, Kluwer academic Publishers, vol. 5, pp. 181-201.
- Krasnogor N. (2004b) “Towards robust memetic algorithms,” In: *Recent Advance in Memetic Algorithms*, Series Studies in Fuzziness and Soft Computing, Hart W. E., Krasnogor N, and Smith J. E. (Eds.), pp. 185-207, Springer-Verlag.
- Krasnogor N., Blackburne B., Burke E., and Hirst J. (2002) “Multimeme algorithms for protein structure prediction,” In: *Proceedings of the Parallel Problem Solving from*

Nature VII, *Lecture Notes in Computer Science*, Springer, pp. 769-778.

Krasnogor N. and Gustafson S. (2002) "Toward truly "memetic" memetic algorithms: discussion and proof of concepts," In: *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, Hart W., Knowles J., Krasnogor N., Roy R., Smith J. E., Corne D., Fogel G., and Tiwari A. (Eds.), PEDAL (Parallel, Emergent and Distributed Architectures Lab), University of Reading, ISBN: 0-9643481-0-9.

Krasnogor N. and Gustafson S. (2003) "The local searcher as a supplier of building blocks in self-generating memetic algorithms," In: *Fourth International Workshop on Memetic Algorithms (WOMA4)*, Smith J., Hart W.E. and Krasnogor N. (Eds.), GECCO 2003 workshop proceedings.

Krasnogor N. and Gustafson S. (2004) "A study on the use of "self-generation" in memetic algorithms," *Natural Computing*, vol. 3, pp. 53-76.

Krasnogor N. and Pelta D. (2002) "Fuzzy memes in multimeme algorithms: a fuzzy-evolutionary hybrid," In: *Fuzzy Sets Based Heuristics for Optimization, Studies in Fuzziness and Soft Computing*, Verdegay J. (Ed.), Physica Verlag, pp. 49-66.

Krasnogor N. and Smith J. (2000) "A memetic algorithm with self-adaptive local search: TSP as a case study," In: *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO-2000)*, Whitley D., Goldberg D., Cantu-Paz E., Spector L., Parmee I., and Beyer H.-G. (Eds.), Morgan Kaufman, San Francisco, pp. 987-994.

Krasnogor N. and Smith J. (2001) "Emergence of profitable search strategies based on a simple inheritance mechanism," In: *Proceedings of the 2001 Genetic and*

- Evolutionary Computation Conference (GECCO-2001)*, Spector L., Goodman E., Wu A., Langdon W., Voigt H. M., Gen M., Sen S., Dorigo M., Pezeshek S., Garzon M., and Burke E. (Eds.), Morgan Kaufmann, San Francisco, pp. 432-439.
- Krasnogor N. and Smith J. (2002) "Multimeme algorithms for the structure prediction and structure comparison of protein," In: *Proceedings of the Bird of a Feather Workshop, GECCO-2002*, pp. 42-44.
- Krasnogor N. and Smith J. (2005) "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474-488.
- Ku K. and Mak M. (1998) "Empirical analysis of the factors that affect the Baldwin effect," In: *Lecture Notes in Computer Science*, Proceedings of Parallel Problem Solving from Nature—PPSN-V, pp. 481–490.
- Ku K. W. C., Mak M. W., and Siu W. C. (2000) "A study of the Lamarckian evolution of recurrent neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, pp. 31-42.
- Laarhoven P. J. M. van, Aarts E. H. L., and Lenstra J. K. (1992) "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, pp. 113-125.
- Lai H. J. (1995) "Introduction to combinatorial optimization," *Graph Theory, Combinatorics, and Algorithms*, vol. 2.
- Land M. W. S. (1998) *Evolutionary Algorithms with Local Search for Combinatorial Optimization*. Ph.D. Thesis, University of California, San Diego.
- Laszewski G. and Muhlenbein H. (1991) "Partitioning a graph with a parallel genetic

- algorithm,” In: *Parallel Problem Solving from Nature*, Schwefel H.-P. and Manner R. (Eds.), Berlin, pp. 165-169, Springer.
- Laursen P. S. (1993) “Simulated annealing for the QAP — Optimal tradeoff between simulation time and solution quality,” *European Journal of Operational Research*, vol. 69, pp. 238-243.
- Li Y. and Pardalos P. M. (1992) “Generating quadratic assignment test problems with known optimal permutations,” *Computational Optimization and Applications*, vol. 1, pp. 163-184.
- Li Y., Pardalos P. M., and Resende M. (1994) “A greedy randomized adaptive search procedure for the quadratic assignment problem,” In: *Quadratic Assignment and Related Problems*, Pardalos P. and Wolkowicz H. (Eds.), AMS: Providence, Rhode Island, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 173-187.
- Lim M. H., Yuan Y., and Omatu S. (2000) “Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem,” *Computational Optimization and Applications*, vol. 15, pp. 249-268.
- Lim M. H., Yuan Y., and Omatu S. (2002) “Extensive testing of a hybrid genetic algorithm for quadratic assignment problem,” *Computational Optimization and Applications*, vol. 23, pp. 47-64.
- Ljubic I. (2004) *Exact and Memetic Algorithms for Two Network Design problems*. Ph.D. Thesis, Faculty of Computer Science, Vienna University of Technology.
- Ljubic I. and Raidle G. R. (2003) “A memetic algorithm for minimum-cost

- vertex-biconnectivity augmentation of graphs,” *Journal of Heuristics*, vol. 9, no. 5, pp. 401-428.
- Loiola E. M., Abreu N. M. M. de, Boaventura-Netto P. O., Hahn P., and Querido T. (2007) “A survey for the quadratic assignment problem,” *European Journal of Operational Research*, vol. 176, pp. 657-690.
- Lourenco H. G., Martin O. C., and Stutzle T. (2003) “Iterated local search,” In: *Handbook of Metaheuristics*, Glover F. and Kochenberger G. A. (Eds.), pp. 321-353. Kluwer Academic Publishers.
- Lozano M., Herrera F., Krasnogor N., and Molina D. (2004) “Real-coded memetic algorithms with crossover hill-climbing,” *Evolutionary Computation*, vol. 12, no. 2, pp. 273-302.
- Magyar G., Johnsson M., and Nevalainen O. (2000) “An adaptive hybrid genetic algorithm for the three-matching problem,” *IEEE transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 135-146.
- Maniezzo V. (1998) “Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem,” *Tech. Rep. CSR 98-1*, C.L. in Scienze dell'Informazione, Universita di Bologna, Sede di Cesena, Italy.
- Maniezzo V., Colorni A., and Dorigo M. (1994) “The ant system applied to the quadratic assignment problem,” *Tech. Rep. 94/28*, IRIDIA, University de Bruxelles.
- Mavridou T.D. and Pardalos P.M. (1997) “Simulated annealing and genetic algorithms for the facility layout problem: A survey,” *Computational Optimization and Applications*, vol. 7, pp.111-126.

- McCormick E. J. (1970) *Human Factors Engineering*, New York, McGraw-Hill.
- Melab N., Talbi E-G., and Cahon S. (2006a) “On parallel evolutionary algorithms on the computational grid,” *Studies in Computational Intelligence (SCI)*, vol. 22, pp. 117-132.
- Melab N., Cahon S., and Talbi E-G. (2006b) “Grid computing for parallel bioinspired algorithms,” *Journal of Parallel and Distributed Computing*, vol. 66, pp. 1052-1061.
- Mendes A., Cotta C., Garcia V., Franca P., and Moscato P. (2005) “Gene ordering in microarray data using parallel memetic algorithms,” In: *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, vol. 00, pp. 604-611.
- Mendes A. and Linhares A. (2004) “A multiple-population evolutionary approach to gate matrix layout,” *International Journal of Systems Science*, vol. 35, no. 1, pp. 13-23.
- Merz P. (2000) *Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies*, Ph.D. Thesis, Parallel Systems Research Group, Department of Electrical Engineering and Computer Science, University of Siegen, Germany.
- Merz P. (2001) “On the performance of memetic algorithms in combinatorial optimization,” In: *Second Workshop on Memetic Algorithms (2nd WOMA)*, Hart W., Krasnogor N., and Smith J. (Eds.), pp. 168–173, San Francisco, California, USA.
- Merz P. (2004a) “NK-fitness landscapes and memetic algorithms with greedy operators and k -opt local search”, In: *Recent Advance in Memetic Algorithms*, Series Studies in Fuzziness and Soft Computing, Hart W. E., Krasnogor N, and Smith J. E. (Eds.), pp.

209-228, Springer-Verlag.

Merz P. (2004b) "Advanced fitness landscape analysis and the performance of memetic algorithms," *Evolutionary Computation*, vol. 12, no. 3, pp. 303-325.

Merz P. and Freisleben B. (1997) "A genetic local search approach to the quadratic assignment problem," In: *Proceedings of International Conference on Genetic Algorithms (ICGA'97)*, Morgan Kaufmann: San Mateo, CA, pp. 465-472.

Merz P. and Freisleben B. (1998) "On the effectiveness of evolutionary search in high-dimensional *nk*-landscapes," In: *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, IEEE Press, pp. 741-745.

Merz P. and Freisleben B. (1999a) "Fitness landscapes and memetic algorithm design," In: *New Ideas in Optimization*, Corne D. et al. (Eds.), McGraw-Hill, London, pp. 245-260.

Merz P. and Freisleben B. (1999b) "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," In: *Proceedings of International Congress on Evolutionary Computation (CEC-1999)*, Washington DC, pp. 2063-2070.

Merz P. and Freisleben B. (2000a) "Fitness landscape, memetic algorithms, and greedy operators for graph bipartitioning," *Journal of Evolutionary Computation*, vol. 8, no. 1, pp. 61-91.

Merz P. and Freisleben B. (2000b) "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337-352.

- Metropolis N., Rosenbluth A. W., Teller A. H., and Teller E. (1953) "Equation of state calculation by fast computing machines," *Journal of Chemical Physics*, vol. 21 pp. 1087–1091.
- Middendorf M., Reischle F., and Schmeck H. (2002) "Multi colony ant algorithms," *Journal of Heuristics*, Kluwer Academic Publishers, vol. 8, pp. 305-320.
- Mills P. and Tsang E. (2000) "Guided local search for solving SAT and weighted MAX-SAT problems," In: *SAT2000 --- Highlights of Satisfiability Research in the Year 2000*, IOS Press, Amsterdam, The Netherlands, Gent I. P., Maaren H. van, and Walsh T. (Eds.), pp. 89-106.
- Misevicius A. (2006) "A fast hybrid genetic algorithm for the quadratic assignment problem," In: *Proceedings of 2006 Genetic and Evolutionary Computation Conference (GECCO-2006)*, pp. 1257-1264, Seattle, USA.
- Molina D., Herrera F., and Lozano M. (2005) "Adaptive local search parameters for real-coded memetic algorithms," In: *The IEEE Congress on Evolutionary Computation (CEC-2005)*, vol. 1, pp. 888-895.
- Mori N., Kita H., and Nishikawa Y. (1998) "Adaptation to a Changing Environment by means of the Feedback Thermodynamical Genetic Algorithm," *Lecture Notes in Computer Science*, vol. 1498, pp. 149-158.
- Morris G. M., Goodsell D. S., Halliday R. S., Huey R., Hart W. E., Belew R. K., and Olson A. J. (1998) "Automated docking using a Lamarkian genetic algorithm and an empirical binding free energy function," *Journal of Computational Chemistry*, vol. 19, no. 14, pp. 1639–1662.

- Moscato P. (1989) *On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms*, Tech. Rep. 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, California, USA.
- Moscato P. (2001) *NP Optimization Problems, Approximability and Evolutionary Computation: From Practice to Theory*. Ph.D. Dissertation, Universidade Estadual de Campinas, Brazil.
- Moscato P. (2002) Memetic algorithms' home page. Online available at <http://www.densis.fee.unicamp.br/moscato/memetichome.html>.
- Moscato P. and Cotta C. (2002) "A gentle introduction to memetic algorithms," In: book chapter of *Handbook of Metaheuristics*, Glover F. and Kochenberger G. A. (Eds.), Klumer Academic Publishers.
- Moscato P. and Norman M. G. (1992) "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," In: *Parallel Computing and Transputer Applications*, Valero M., Onate E., Jane M., Larriba J. L., and Suarez B. (Eds.), Amsterdam, Netherlands, IOS Press, pp. 177-186.
- Muhlenbein H. (1989) "Parallel genetic algorithms, population genetics and combinatorial optimization," In: *Proceedings of the 3rd International Conference on Genetic Algorithms*, Schaffer (Ed.), Morgan Kaufmann, pp. 416-421.
- Muhlenbein H. and Paab G. (1996) "From recombination of genes to the estimation of distribution," In: *Proceedings of the 4th Conference on Parallel Problem Solving from Nature – PPSN IV*, Voigt H. M., Ebeling W., Rechenberg I., and Schwefel H. P.

- (Eds.), *Lecture Notes in Computer Science*, vol. 1411, pp. 178-187, Berlin, Springer.
- Murata T. and Ishibuchi H. (1998) "Constructing multi-objective genetic local search algorithms for multi-objective flowshop scheduling problems," In: *Proceedings of the 1998 Japan-USA Symposium on Flexible Automation*, pp. 1353-1356, Ohtsu, Japan.
- Murata T., Ishibuchi H. and Gen M. (2000) "Cellular genetic local search for multi-objective optimization," In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Whitley D., Goldberg D., Cantu-Pza E., Spector L., Parmee I., and Beyer H.-G. (Eds.), pp. 307-314, San Francisco, California, Morgan Kaufmann.
- Murata T., Ishibuchi H. and Gen M. (2001) "Specification of genetic search directions in cellular multi-objective genetic algorithms," In: *First International Conference on Evolutionary Multi-Criterion Optimization*, Zitzler E., Deb K., Thiele L., Coello C.A.C., and Corne D. (Eds.), Springer-Verlag. *Lecture Notes in Computer Science*. vol. 1993, pp. 82-95.
- Murata T., Kaige S. and Ishibuchi H. (2003) "Generalization of dominance relation-based replacement rules for memetic EMO algorithms," In: *Proceedings of Genetic and Evolutionary Computation (GECCO-2003)*, Cantu-Pza E. *et al.* (Eds.), Part I, Springer, *Lecture Notes in Computer Science*, vol. 2723, pp. 1234-1245.
- Nel G. (2004) *A Memetic Genetic Program for Knowledge Discovery*, Master Thesis, University of Pretoria.
- Neri F., Toivanen J., Cascella G. L., and Ong Y. S. (2006) "An adaptive multimeme algorithm for designing HIV multidrug therapies," *IEEE/ACM Transactions on*

Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics, Accepted 2006.

Ning, Z., Ong Y. S., Wong K. W., and Lim M. H. (2003) "Choice of memes in memetic Algorithm," In: *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems*.

Nissen V. (1994) "Solving the quadratic assignment problem with clues from nature," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 66-72.

Nissen V. and Paul H. (1995) "A modification of threshold accepting and its application to the quadratic assignment problem," *OR Spektrum*, vol. 17, pp. 205-210.

Ong Y. S. and Keane A. J. (2004) "Meta-Lamarckian in memetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99-110.

Ong Y. S., Lim M. H., Zhu N., and Wong K. W. (2006) "Classification of adaptive memetic algorithms: a comparative study," *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 36, no. 1, pp. 141-152.

Ozcan E. (2006) "An empirical investigation on memes, self-generation and nurse rostering," In: *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT'2006)*, Masaryk University, Brno, Czech Republic E.K. Burke, H. Rudov (Eds.), pp. 246-263.

Pardalos P. and Pitsoulis L. (2000) *Nonlinear Assignment Problems — Algorithms and Applications*, Kluwer Academic: Dordrecht.

Pardalos P. and Wolkowicz H. (1994) "DIMACS series in discrete mathematics and

- theoretical computer science”, *American Math. Soc.*, vol. 16.
- Pelta D. and Krasnogor N. (2004) “Multimeme algorithms using fuzzy logic based memes for protein structure prediction,” In: Book chapter in “*Recent Advance in Memetic Algorithms*”, Series Studies in Fuzziness and Soft Computing, Hart W. E., Krasnogor N, and Smith J. E. (Eds.), pp. 49-64, Springer-Verlag.
- Pitsoulis L. S. and Resende M. G. C. (2002) “Greedy randomized adaptive search procedures,” In: *Handbook of Applied Optimization*, Pardalos P. M. and Resende M. G. C. (Eds.), pp. 168-183, Oxford University Press.
- Prais M. and Ribeiro C. C. (2000) “Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment,” *INFORMS Journal on Computing*, vol. 12, pp. 164-176.
- Preux P. and Talbi E. G. (1999) “Towards hybrid evolutionary algorithms,” *International Transactions in Operational Research*, vol. 6, no. 6, pp. 557-570.
- Puchinger J., Raidl G. R., and Gruber M. (2005) “Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem,” In: *The 6th Metaheuristics International Conference (MIC 2005)*, pp. 775-780, Vienna, Austria.
- Quintero A. and Pierre S. (2002) “A memetic algorithm for assigning cells to switches in cellular mobile networks,” *IEEE Communications Letters*, vol. 6, no. 11, pp. 484-486.
- Rechenberg I. (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen evolution*. Frommann-Holzboog.

- Resende M. G. C. and Ribeiro C. C. (1997) "A GRASP for graph planarization," *Networks*, vol. 29, pp. 173-189.
- Resende M. G. C., Pardalos P. M., and Eksioglu S. D. (2001) "Parallel metaheuristics for combinatorial optimization," *Advanced Algorithmic Techniques of Parallel Computation with Applications*, Correa R. *et al.* (Eds.), Kluwer, pp. 179-206.
- Rosca J. (1995) "Entropy-driven adaptive representation." In: *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, J. Rosca (Ed.), Tahoe City, California, USA, pp. 23-32.
- Rudolph G. (1996) "Convergence of evolutionary algorithms in general search spaces," In: *Proceedings of the International Congress of Evolutionary Computation (CEC-1996)*, pp. 50-54.
- Sahni S. and Gonzalez T. (1976) "P-complete approximation problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555-565.
- Sekaj I. (2004) "Robust parallel genetic algorithms with re-initialisation," In: Yao X. *et al.* (Eds.), *Parallel Problem Solving from Nature-PPSN VIII, 8th International Conference*, Birmingham, UK, Proceedings, vol. 3242 of Lecture Notes in Computer Science, pp. 411-419, Springer.
- Shahidi N., Esmailzadeh H., Abdollahi M., Ebrahimi E., and Lucas C. (2004) "Self-adaptive memetic algorithm: An adaptive conjugate gradient approach," In: *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 6-11.
- Sharpe O. J. (2000) *Towards a Rational Methodology for Using Evolutionary Search*

Algorithms. Ph.D. Thesis, COGS, University of Sussex.

Skolicki Z. and Jong K. D. (2005) “The influence of migration sized and intervals on island models,” In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005)*, Springer-Verlag.

Skorin-Kapov J. (1990) “Tabu search applied to the quadratic assignment problem,” *ORSA Journal on Computing*, vol. 2, no. 1, pp. 33-45.

Skorin-Kapov J. (1994) “Extensions of a tabu search adaptation to the quadratic assignment problem,” *Computers & Operations Research*, vol. 21, no. 8, pp. 855-865.

Smith J. E. (2002) “Co-evolving memetic algorithms: Initial investigations,” In: *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN-VII), Lecture Notes In Computer Science*, vol. 2439, pp. 537-548.

Smith J. E. (2003a) “Co-evolving memetic algorithms: A learning approach to robust scalable optimization,” In: *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2003)*, vol. 1, pp. 498-505.

Smith J. E. (2003b) “Protein structure prediction with co-evolving memetic algorithms,” In: *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2003)*, vol. 4, pp. 2346-2353.

Smith J. E. (2004) “The co-evolution of memetic algorithms for protein structure prediction,” In: Book chapter in “*Recent Advance in Memetic Algorithms*”, Series Studies in Fuzziness and Soft Computing, Hart W. E., Krasnogor N, and Smith J. E. (Eds.), pp. 105-128, Springer-Verlag.

- Sorensen K. and Sevaux M. (2006) “MA|PM: Memetic algorithms with population management,” *Computers & Operations Research*, vol. 33, no. 5, pp. 1214-1225.
- Steenbeek A. G., Marchiori E., and Eiben A. E. (1998) “Finding balanced graph bi-partitions using a hybrid genetic algorithm,” In: *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98)*, IEEE Press, Piscataway, New Jersey, pp. 90—95.
- Steinberg L. (1961) “The backboard wiring problem: A placement algorithm,” *SIAM Review*, vol. 3, pp. 37-50.
- Storn R. and Price K. (1995) “Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces,” *Technical Report TR-95-012*, International Computer Science Institute, Berkeley, CA.
- Stutzle T. (2006) “Iterated local search for the quadratic assignment problem,” *European Journal of Operational Research*, vol. 174, pp. 1519-1539.
- Stutzle T. and Dorigo M. (1999) “ACO algorithms for the quadratic assignment problem,” In: *New Ideas in Optimization*, Corne D., Dorigo M., and Glover F. (Eds.), McGraw-Hill, London, pp. 33-50.
- Stutzle T. and Hoos H. (1999) “MAX-MIN ant system and local search for combinatorial optimization problems,” In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Voss S., Martello S., Osman I., and Roucairol C. (Eds.), Kluwer, Boston, pp. 313-329.
- Sudholt D. (2006) “On the analysis of the (1+1) memetic algorithm,” In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2006)*, Seattle,

Washington, USA.

Suzuki J. (1995) "A Markov chain analysis on simple genetic algorithms," *IEEE Transactions on System, Man, and Cybernetics*, vol. 25, no. 4, pp. 655-659.

Taillard E. D. (1991) "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, vol. 17, pp. 443-455.

Taillard E. D. (1995) "Comparison of iterative searches for the quadratic assignment problem," *Location Science*, vol. 3, pp.87-105.

Talbi E. G. (2002) "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, pp. 541-564.

Talbi E. G., Rahoual M., Mabed M. H., Dhaenens C. (2001) "A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop," In: *Proceedings of 1st International Conference on Evolutionary Multi-Criterion Optimization*, pp. 416-428, Zurich, Switzerland.

Tan K. C., Lee T. H., and Khor E. F. (2001) "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 565-588.

Tanese R. (1989) "Distributed genetic algorithms," In: *ICGA-3*, Schaffer J. D. (Ed.), pp. 434-439.

Tang J., Lim M. H., and Ong Y. S. (2003a) "A parallel hybrid GA for combinatorial optimization using grid technology," In: *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2003)*, Canberra, Australia, vol. 3, pp. 1895-1902.

Tang J., Lim M. H., and Ong Y. S. (2003b) "Island model parallel hybrid-GA for large

scale combinatorial optimization,” In: *2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Special Session on Optimization using Genetic, Evolutionary, Social and Behavioral Algorithms, Singapore.

Tang J., Lim M. H., Ong Y. S., and Er M. J. (2004) “Study of migration topology in island model parallel hybrid-GA for large scale quadratic assignment problems,” In: *The Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV2004)*, Special Session on Computational Intelligence on the Grid, Kunming, China.

Tang J., Lim M. H., Ong Y. S., and Er M. J. (2005) “Solving large scale combinatorial optimization using PMA-SLS,” In: *Proceedings of 2005 Genetic and Evolutionary Computation Conference (GECCO’05)*, Program Track on Evolutionary Combinatorial Optimization, Washington D.C., USA.

Tang J., Lim M. H., Ong Y. S., and Er M. J. (2006a) “Parallel memetic algorithm with selective local search for large scale quadratic assignment problems,” *International Journal of Innovative Computing, Information and Control*, vol. 2, no. 6, pp. 1399-1416, ISSN: 1349-4198.

Tang J., Lim M. H., and Ong Y. S. (2006b) “Adaptation for parallel memetic algorithm based on population entropy,” In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO’06)*, Seattle, Washington, USA. “Nominated for Best Paper Award”.

Tang J., Lim M. H. and Ong Y. S. (2006c) “Diversity-adaptive parallel memetic

- algorithm for solving large scale combinatorial optimization problems,” *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, online published on October 18, DOI: 10.1007/s00500-006-0139-6.
- Tate D. M. and Smith A. E. (1995) “A genetic approach to the quadratic assignment problem,” *Computers & Operations Research*, vol. 22, no. 1, pp. 73-83.
- Thonemann U. W. and Bolte A. (1994) “An improved simulated annealing algorithm for the quadratic assignment problem,” *Technical report*, School of Business, Department of Production and Operations Research, University of Paderborn, Germany.
- Ulder N. L. J., Aarts E. H. L., Bandelt H. J., Laarhoven P. J. M. van, *et al.* (1991) “Genetic local search algorithms for the traveling salesman problems,” In: *Parallel Problem Solving from Nature - Proceedings of 1st Workshop*, PPSN 1, Schwefel H. P. and Anner R. M. (Eds.), Dortmund, Germany, pp. 109-116, Springer-Verlag, Berlin, Germany.
- Vazquez M. and Whitley L. (2000) “A hybrid genetic algorithm for the quadratic assignment problem,” In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2000)*, Whitley D., Goldberg D., Cantu-Paz E., Spector L., Parmee I., and Beyer H.-G. (Eds.), pp. 135–142.
- Voudouris C. and Tsang E. (1999) “Guided local search,” *European Journal of Operational Research*, vol. 113, no. 2, pp. 469-499.
- Whitley D. (1993) “An executable model of a simple genetic algorithm,” In: *Foundations of Genetic Algorithms*, Whitley D. (Ed.), Morgan Kaufmann, San Mateo, vol. 2, pp.

45-62.

Wiering M. (2004) "Memory-based memetic algorithms," In: *Proceedings of the Thirteenth Belgian-Dutch Conference on Machine Learning (Benelearn'04)*, edited by Ann Nowe.

Wilhelm M. R. and Ward T. L. (1987) "Solving quadratic assignment problems by simulated annealing," *IIE Transactions*, vol. 19, no. 1, pp. 107-119.

Wolpert D. and Macready W. (1997) "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82.

Xu X. and He H. G. (2005) "A theoretical model and convergence analysis of memetic evolutionary algorithms," In: LCNC 2005, Wang L., Chen K, and Ong Y. S. (Eds.), *Lecture Notes in Computer Science*, vol. 3611, pp. 1035-1043.

Xu Y. L., Lim M. H., and Yan K. H. (2003) "Hybrid-GA with stochastic selection local search for solving quadratic assignment problems," In: *Proceedings of 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Special Session on Optimization using Genetic, Evolutionary, Social and Behavioral Algorithms, Singapore.

Yamada T. and Nakano R. (1996) "Scheduling by genetic local search with multi-step crossover," In: *Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV*, Voigt H.-M., Ebeling W., Rechenberg I., and Schwefel H.-P. (Eds.), pp. 960-969, Springer.

Yao X. (ed.) (1999) *Evolutionary Computation: Theory and Applications*, World Scientific Publ. Co., Singapore. (ISBN 981-02-2306-4).

Zhang Q., Sun J., Tsang E. P. K., and Ford J. A. (2004) “Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem,” In: *Proceedings of the Bird of a Feather Workshops, Genetic Assignment Computation Conference*, pp. 42-48.