Flex Serv Manuf J (2011) 23:207–241
DOI 10.1007/s10696-011-9099-y

Evolutionary algorithms for solving multi-objective travelling salesman problem

Vui Ann Shim · Kay Chen Tan · Jun Yong Chia · Jin Kiat Chong

Published online: 1 June 2011

Abstract This paper studies the application of evolutionary algorithms for bi-objective travelling salesman problem. Two evolutionary algorithms, including estimation of distribution algorithm (EDA) and genetic algorithm (GA), are considered. The solution to this problem is a set of trade-off alternatives. The problem is solved by optimizing the order of the cities so as to simultaneously minimize the two objectives of travelling distance and travelling cost incurred by the travelling salesman. In this paper, binary-representation-based evolutionary algorithms are replaced with an integer-representation. Three existing EDAs are altered to use this integer-representation, namely restricted Boltzmann machine (RBM), univariate marginal distribution algorithm (UMDA), and population-based incremental learning (PBIL). Each city is associated with a representative integer, and the probability of any of this representative integer to be located in any position of the chromosome is constructed through the modeling approach of the EDAs. New sequences of cities are obtained by sampling from the probabilistic model. A refinement operator and a local search operator are proposed in this piece of work. The EDAs are subsequently hybridized with GA in order to complement the limitations of both algorithms. The effect that each of these operators has on the quality of the solutions are investigated. Empirical results show that the hybrid algorithms are capable of finding a set of good trade-off solutions.

Keywords Estimation of distribution algorithm · Evolutionary multi-objective optimization · Probabilistic model · Restricted Boltzmann machine · Travelling salesman problem

V. A. Shim (✉) · K. C. Tan · J. Y. Chia · J. K. Chong
Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, 117576 Singapore, Singapore
e-mail: g0800438@nus.edu.sg

1 Introduction

The travelling salesman problem (TSP) is a famous permutation-based combinatorial optimization problem which has been extensively studied over the past few decades. Despite its simplicity in understanding and general applicability to most scheduling problems, the TSP is notoriously difficult to solve. Researchers have recognized it as an NP-hard problem (Larrañaga and Lozano 2001; Aarts and Korst 1989). Some problems which can be described using the TSP formulation are the gene sequencing problem (Blazewicz et al. 2002), the dartboard design problem (Eiselt and Laporte 1991), the hole punching problem (Reinelt 1989), etc. Additional literature reviews of the development of TSP can be found in (Laporte 1992; Bagchi et al. 2006).

The TSP aims to minimize the total distance travelled, in which each city can only be visited once and the salesman must return to the starting depot after visiting the ordered cities. The coordinate of the cities are known in advance in order to find the pairwise distance of the cities. The adaptation of TSP into multi-objective framework (MOTSP) is a promising area which can be further explored (Jahne et al. 2009). In MOTSP, the aim is to simultaneously optimize several conflicting objectives, such as shortest travelling distance, minimum time, minimum cost, and lowest risk (Yang et al. 2007). Under multi-objective framework (Coello Coello 2006), no single point is considered as an optimal solution, because an improvement in one objective will cause at least another objective not being able to be optimized. Therefore, the optimal solution can only be a set of non-dominated and trade-off solutions.

Many practical problems are closely related to the MOTSP, such as logistic, planning, transportation, and gene sequencing problems. One recent application of the MOTSP is the study of berth allocation problem in container ports (Cheong et al. 2010). The problem aims to find the optimal berth schedule which simultaneously minimizes the three objectives of waiting time, makespan, and degree of deviation from a predetermined schedule. A fixed-length chromosome representation was proposed which represents a complete berth schedule. In a chromosome, the order of the ships that are allocated to the berth is determined by the genetic operators. In bioinformatics field, the concept of MOTSP is slightly modified and used to solve the deoxyribonucleic acid (DNA) sequencing problems (Shin et al. 2005). Given the spectrums that are extracted from DNA-related experiments, such as DNA microarrays, the problem aims to find a complete sequence of DNA. In this problem, the cities are replaced by the DNA spectrums and the distance between the cities is replaced by the criterion like a similarity measure between the spectrums.

Over the past few decades, various approaches have been proposed to solve the TSP. They include linear programming (Diaby 2007), dynamic programming (Malandraki and Dial 1996), branch and cut algorithm (Padberg and Rinaldi 1988), simulated annealing (Wu et al. 2007), tabu search (Wu and Zhou 2008), evolutionary algorithm (EA) (Yan et al. 2007), neural network (Li et al. 2009), clustering (Yang et al. 2009), etc. Among them, the use of EA to solve TSP and MOTSP has attracted considerable attention (Shi and Li 2009; Yugay et al. 2008;

Zhou and Jia 2008; Elaoud et al. 2010). This may be attributed to the fact that EAs, which are probabilistic-based algorithms, are less susceptible to be trapped at local optima. In addition, the population-based approach in EAs has the capability to generate a set of trade-off solutions in just a single simulation run. Selection and reproduction operators inspired from biological evolution enable the algorithms to deal with hard problems which of high dimensionality and having a large search space.

Genetic algorithm (GA) is a famous computing paradigm inspired by biological evolution. The variation operators of GA are based on crossover and mutation. Over the past few decades, there were several remarkable research efforts to drive GA to handle TSP. However, it is commonly known that the stochastic recombination operators of GA may disrupt the building of good schemas. An alternative to the use of GA is the estimation of distribution algorithms (EDAs). EDAs are motivated by the concept of using probability distribution of the candidate solutions to predict the movement in the search space. Another important feature of EDAs is the exploitation of the linkage information between the decision variables which will be used to guide in the search (Muhlenbein and Paass 1996; Larrañaga and Lozano 2001; Pelikan et al. 2002; Gonzales 2005; Lozano et al. 2006). More detailed information about linkage information can refer to Ting et al. (2010), Zhu et al. (2010), and Da San Martino and Sperduti (2010).

EDAs are similar to GA in the sense that they also mimic the principle of biological evolution, which is the survival of the fittest, to guide the search. However, the primary difference between EDAs and GA is that there is no implementation of genetic operators (crossover and mutation) in EDAs. Instead, the reproduction is carried out by the building of a representative probabilistic model of candidate solutions. Subsequently, offspring are generated through the sampling of the constructed model. The probabilistic-based approach in EDAs provides a strong search behavior through the consideration of the global probability distribution and linkage information (the probability of certain genes to be inherited by others) of the candidate solutions in the decision space. Over the past few years, several EDAs for multi-objective optimization (MOEDAs) (Pelikan et al. 2006) have been proposed. The probabilistic modeling in EDAs can be based on probability information (Bosman and Thierens 2002; Sastry et al. 2005; Neri and Mininno 2010), statistical techniques (Laumanns and Ocenasek 2002; Costa and Minisci 2003; Okabe et al. 2004), machine learning approaches (Zhang et al. 2008; Zhong and Li 2007; Arel et al. 2010), and neural net mechanisms (Marti et al. 2009; Tang et al 2010; Shim et al. 2010).

Solving scheduling problems (specifically the TSP) with the use of EDAs is a new area of research that has been explored recently. In (Larrañaga and Lozano 2001), the first adaptation of EDAs to solve the TSP was carried out. Several EDAs with discrete and continuous representations were utilized. Domain knowledge was used as a way to introduce a local search operator to facilitate the search. In Chen et al. (2010), the guidelines of implementing the use of EDAs in scheduling problems were being presented. The intensification and diversification effect of

EDAs can be enhanced by hybridizing EDAs with other meta-heuristic approaches. Next, an attempt was carried out by Jarboui et al. (2009) to study the permutation flowshop scheduling problem using one of the simplest EDAs which is known as univariate marginal distribution algorithm (UMDA) (Muhlenbein 1998). In the implementation, the order of the jobs to be performed in the sequences and the similar blocks of the jobs were modeled. A variable neighborhood search operator was also proposed to enhance the search. The operator is used to determine when and how the local search is to be performed. In Salhi et al. (2007), another research to study the flow shop scheduling was carried out. The authors studied the hybrid flow shop with sequence dependent setup times and uniform machines in parallel. The EDA mechanism was based on the population-based incremental learning (PBIL) (Baluja 1994), and the learning rate in PBIL was investigated. The algorithm adapted guided mutation to enhance the exploitation capability of the algorithm. In another study (Lin 2009), a new EDA based on maximum entropy was proposed to deal with the job shop scheduling problem. In Xu et al. (2009), EDA was used to estimate the distribution of the state transitions and the global updating rules of ant colony system. In this case, EDA is not applied directly to deal with the TSP; but acted as an optimization tool to update the parameter in the ant colony system. Most of the mentioned studies carried out the performance comparison between EDA and GA, and the experimental results indicated that the performance of EDA, when hybridized with local search, is better than GA in terms of solution quality and convergence speed.

While a great deal of efforts was being put in to study the single-objective permutation-based problems (specifically the TSP) using EDAs, there is still no research which studies multi-objective permutation-based problems (specifically the MOTSP) using EDAs. In this paper, three of the MOEDAs in binary-representation, namely multi-objective univariate marginal probability algorithm (MOUMDA) (Muhlenbein and Paass 1996), multi-objective population based incremental learning (MOPBIL) (Baluja 1994), and multi-objective restricted Boltzmann machine (MORBM) (Tang et al. 2010; Shim et al. 2010), are being adapted into a permutation-based integer-representation to solve the bi-objective TSP. These algorithms utilize the principles of Pareto optimality to deal with the multiple conflicting objectives. A permutation refinement operator is proposed to refine the cities in a chromosome to guarantee that no city is repeated. A local search operator is also presented to enhance the search capability of the algorithms. EDAs are subsequently hybridized with GA to improve the diversity of the trade-off solutions. The influences of each operator and parameter settings are also studied.

The rest of the paper is organized as follows. Section 2 describes the formulation of the travelling salesman problem in a multi-objective framework, as well as the principle behind the multi-objective optimization. Section 3 presents the algorithm's framework, together with their corresponding operators. The experimental results, investigations, and discussions are being outlined in Sect. 4. The paper ends off with the conclusions in last section. For the ease of reference, the notations used throughout the paper are being listed in Table 1.

Table 1 Notations

Notations	Description
F	Objective function, $F = F_1,\ldots F_m$
m	Number of objective functions
x_i	The representative integer of a city at ith position of a chromosome, $x = x_1,\ldots x_m$
D^m	The objective considered in the mth objective function
n	Number of cities or number of decision variables
$Pr_g(x_i)$	Probability matrix modeled by EDAs at generation g
$Pr_g(x_i = c_j)$	Probability of city j at the ith position of the chromosome at generation g
c_j	City j ($c_j = j$)
N	Population size
α	Learning parameter in PBIL
w_{ij}	Connection weight between the ith visible unit and the jth hidden unit of RBM
b_i	Bias for the ith visible unit of RBM
b_j	Bias for the jth hidden unit of RBM
E	Energy function of a RBM
v_i	State of the ith visible unit of RBM
h_j	State of the jth hidden unit of RBM
Z	Normalizing constant in RBM
L_i	Location of repeated cities
V_j	Unvisited cities
R	Number of repeated cities in a chromosome before refinement
Score 1 (j)	Score for the V_j as used in the refinement operator
κ	Number of cities to be relocated as used in the local search operator
ℓ	The ℓth position of a chromosome used in the local search operator
U_j	City to be relocated as used in the local search operator
Score 2 (j)	Score for the U_j used in the local search operator
H	Number of hidden units in RBM
LS	Local search rate
fr	Frequency of alternation in hybrid algorithms

2 Multi-objective travelling salesman problem

In this section, the mathematical formulation of the MOTSP and the principle behind the multi-objective optimization are presented.

2.1 Problem formulation

The TSP is a combinatorial optimization problem which is applicable to many scheduling problems. The aim of the problem is to find the route that has shortest travelling distance or lowest travelling cost for visiting all the cities. Each city is only can be visited once and the salesman must return to the starting depot after visiting the ordered cities. Mathematically, the problem in multi-objective

framework (Peng et al. 2009) can be formulated for the minimization case as:Minimize:

$$\begin{aligned} F(x) &= (F_1(x), F_2(x), \ldots, F_m(x)), \quad \text{where} \\ F(x) &= \sum_{i=1}^{n-1} D^1(x_i, x_{i+1}) + D^1(x_n, x_1) \\ &\vdots \\ F_m(x) &= \sum_{i=1}^{n-1} D^m(x_i, x_{i+1}) + D^m(x_n, x_1) \end{aligned} \tag{1}$$

where m is the number of objective functions, n is the number of cities, x_i is the representative integer of the city at ith position of a chromosome, $D^m(x_i, x_{i+1})$ can be considered as the travelling distance, travelling cost, or travelling risk (for the mth objective function) between cities at ith and $i + 1$th positions of a chromosome. The constraint in the problem is that all the cities must be visited exactly once in a route. In other words, there should be no repeated visit on any city in the route of the salesman. Readers can refer to García-Martínez et al. (2004), Herrera et al. (2007), Peng et al. (2009), Manuel and Thomas (2010) for a more detailed study of the MOTSP. In this paper, only two objectives (travelling distance and travelling cost) are being considered.

2.2 Multi-objective optimization

In multi-objective optimization, the aim is to simultaneously optimize m objective functions, which has n decision variables. Without loss of generality, the multi-objective optimization problem can be formulated for minimization case as follows:Minimize:

$$F(x) = (F_1(x), F_2(x), \ldots, F_m(x)) \tag{2}$$

where $x = (x_1, \ldots, x_n), x \in R^n, R^n$ is the decision space, $F(x) \in R^m$, and R^m is the objective space. In order to determine which solution is better than another, the principle of domination is utilized. Under this principle, let a and b be two decision vectors, a is said to dominate b ($a < b$) if and only if

$$F_i(a) \leq (F_i(b), \forall i \in \{1, 2, \ldots, m) \tag{3}$$

and a and b are incomparable ($a \sim b$) if and only if

$$F_i(a) = (F_i(b) \exists i \in \{1, 2, \ldots, m\} \quad \text{and} \quad F_j(a) < (F_j(b) \exists j \in \{1, 2, \ldots, m\} \tag{4}$$

A solution is non-dominated and Pareto optimal if there are no other solutions which dominate it. The set of Pareto optimal solutions in decision space is called Pareto optimal set (PS) and the corresponding objective vectors form the Pareto optimal front (PF). The concept of Pareto dominance is further illustrated in Fig. 1. In the figure, F1 is the first objective and F2 is the second objective. Solutions A, B, C, and D are mutually non-dominated and solutions B and C dominate solution E.

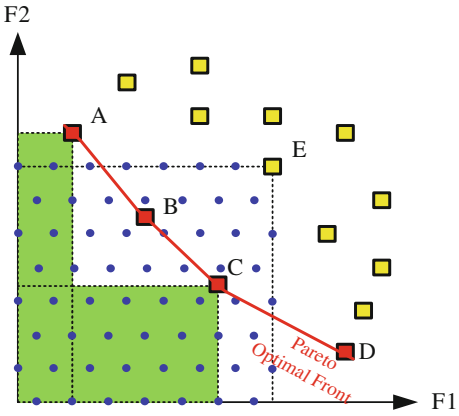

Fig. 1 The concept of domination

A set of non-dominated solutions (A, B, C, and D) will form the Pareto optimal front. Further description of multi-objective optimization can be found in Zitzler (1999).

3 Algorithms' framework

In this section, the overall framework of the algorithms and the corresponding operators are being presented. Firstly, the integer-representation of a chromosome will be described. Next, the fitness assignment operator based on Pareto ranking and crowding distance is outlined. Then, the modeling and sampling approaches of EDAs and recombination of GA are presented. In the original version of EDAs, binary-representation is being used. However, since integer-representation is used in this study, the modeling and sampling approaches are modified accordingly. Subsequently, the description of the enhanced operators, namely permutation refinement operator and local search operator, are presented. The overall framework of the algorithms is outlined in the final sub-section.

3.1 Permutation-based representation

The chromosome is represented by a set of sequenced cities, as shown in Fig. 2. Each chromosome (ch1 and ch2) encodes a complete solution. Integer value is used to represent the cities, where 1 means the first city. The gene of a chromosome will therefore consist of different integer values ranging from 1 to n (where n is the maximum number of city to be visited). The sequence of the cities to be visited is denoted by the sequence which it appears in the chromosome.

3.2 Pareto-based ranking

In multi-objective optimization, no single point is an optimal solution. Instead, the optimal solution is made up of a set of non-dominated individuals. Under this

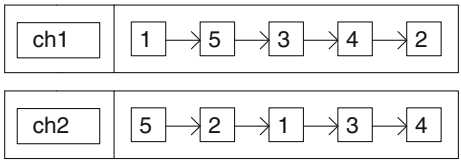

Fig. 2 Integer number representation

condition, the objective values derived from the objective functions cannot be directly used as the fitness values.

Therefore, a way to reassign the fitness to the candidate solutions is required. Through literature review, it is seen that the Pareto-based ranking and crowding distance is one of the favorite fitness assignment operators (Deb et al. 2002). Under this scheme, solution A is fitter than solution B if and only if all the objective values of the solution A are smaller than those of the solution B (for the case of minimization problems). Solution A and solution B are incomparable if and only if the objective values of the solution A are not all smaller than those of the solution B. By applying this concept, the solutions are ranked according to their rank of domination. Solutions that are not being dominated by any other solutions are ranked as one, while solutions which are only dominated by the solutions in rank one are ranked as two, and so forth. In this case, the solutions in smaller rank are fitter than those in higher rank. This ranking mechanism is illustrated in Fig. 3. By applying natural selection pressure, the chances of selecting solutions in smaller rank are higher than solutions in higher rank. In the implementation, binary tournament selection (Miller and Goldberg 1996) is being used. Under this selection scheme, two chromosomes are randomly selected to undergo tournament selection by comparing their rank of domination. The solution with the smaller rank is preferred, and thus it will survive to the next generation. However, if two solutions are located at the same rank, then the one that has the greater crowding distance is preferred. Crowding distance of a solution is calculated by summing over its two nearest neighbors. The procedure, illustrated in Fig. 4, calculates the crowding distance of solution A by the summation of L1 to L4. The crowding distance for solutions at the boundary location (solutions B and C in Fig. 4) is set to an infinite value, and this leads to the increase in the possibility of these solutions to survive a tournament selection infinitely. This procedure aims to increase the capability of the algorithm to evolve a set of diverse solutions. The Pareto-based ranking and crowding distance serve as the fitness assignment operator in all algorithms. This fitness assignment operator may produce a set of trade-off solutions with good proximity and diversity.

3.3 Modeling and reproduction

This section describes the three different types of EDAs as well as the genetic algorithm (GA). EDAs apply probabilistic modeling approaches to learn the probability distribution of the candidate solutions. Three EDAs, including univariate marginal distribution algorithm (UMDA), population-based incremental learning (PBIL), and restricted Boltzmann machine (RBM), are considered in this paper. The

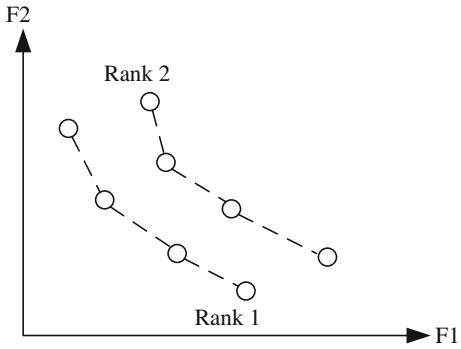

Fig. 3 Pareto-based ranking. F1 is the first objective and F2 is the second objective

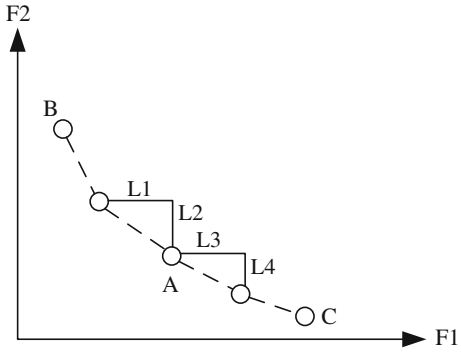

Fig. 4 Crowding distance measurement. F1 is the first objective and F2 is the second objective

original versions of the EDAs are in binary-representation. In order to deal with the MOTSP, the probabilistic modeling and sampling approaches are modified to handle the integer-representation.

3.3.1 Univariate marginal distribution algorithm (UMDA)

UMDA (Muhlenbein 1998) uses univariate modeling for the learning of the probability distribution of the cities in each position of the chromosome without consideration of the linkage dependencies between the cities. In the modeling, a $n \times n$ probability matrix, as shown in Eq. 5, which models the probability of the cities, is constructed

$$Pr_g(x_i) = \begin{bmatrix} Pr_g(x_1 = c_1) & \cdots & Pr_g(x_n = c_1) \\ \vdots & \ddots & \vdots \\ Pr_g(x_1 = c_n) & \cdots & Pr_g(x_n = c_n) \end{bmatrix} \quad (5)$$

where $Pr_g(x_i)$ is the probability distribution of the cities at generation g, $Pr_g(x_i = c_j)$ is the probability of city j to be located at the ith position of the chromosome, c_j is the city j ($c_j = j$), and n is the number of cities. The modeling considers the frequency of existence of the cities in each location of the chromosome.

The probability of the cities in each position of the chromosome is calculated according to Eq. 6

$$Pr_g(x_i = c_j) = \frac{\sum_{p=1}^{n} \delta_p(x_i = c_j) + 1/n}{N + N/n}$$
$$\delta_p(x_i = c_j) = \begin{cases} 1 & \text{if } x_i = c_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where N is the population size. The term $1/n$ is added to set the upper and lower bounds to the probability of each city. This is important as the probability of 0.0 and 1.0 will make no progress in future evolutions since a probability of 0.0 means that there will never be a generation of this particular city in the position of the chromosome. Similarly, a probability of 1.0 will mean that there will always be the generation the same city in the same position of the chromosome.

3.3.2 Population-based incremental learning (PBIL)

PBIL (Baluja 1994) is another earlier version of EDAs which uses the same modeling approach as UMDA. The primary difference of the PBIL with the UMDA is in terms of its probability updating rule. In PBIL, the probability of the cities in each position of the chromosome is calculated by considering the probability of the cities in current and previous generations. The updating rule is presented below:

$$P_g(x_i = c_j) = \alpha Pr_g(x_i = c_j) + (1 - \alpha) Pr_{g-1}(x_i = c_j) \quad (7)$$

where $\alpha \in [0, 1)$ is the learning parameter of the algorithm, $P_g(x_i = c_j)$ is the final probability of city j at the ith position of the chromosome at generation g, $Pr_g(x_i = c_j)$ is obtained according to Eq. 6. α is set to 1.0 at first generation since there is no prior probability distribution from any previous generation. In this situation, PBIL is similar to UMDA. In order to differentiate PBIL from UMDA, α would never be set to 1.0 over course of the evolution process.

3.3.3 Restricted Boltzmann machine (RBM)

RBM is an energy-based neural network (Hinton 2005; Salakhutdinov et al. 2007). The network learns the probability distribution of the input stimuli through unsupervised learning. The network consists of two layers of neurons, one visible layer and one hidden layer. Visible layer is the input frame to the network, while the capability of the network in modeling the distribution of the input data is determined by the number of hidden neurons. The architecture of the RBM is illustrated in the left portion of Fig. 5. The indices of the visible and hidden units are denoted by i and j, respectively. w_{ij} is the weight of the connection between the ith visible unit and the jth hidden unit, b_i is the bias for the ith visible unit, and b_j is the bias for the jth hidden unit. The energy function of the RBM is given as follows:

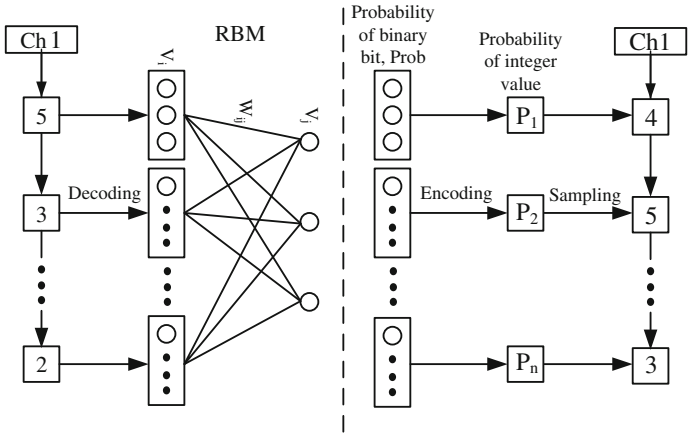

Fig. 5 RBM framework in integer number representation

$$E(v,h) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j \tag{8}$$

where v_i is the state of the ith visible unit and h_j is the state of the jth hidden unit.

In RBM modeling, the network is trained using contrastive divergence (CD) learning (Hinton 2002, 2005; Tieleman 2008) in order to obtain the trained weights and biases. The training of the RBM undergoes two phases (positive and negative phases). During the positive phase, the conditional probability of the hidden units given the input data is computed according to Eq. 9

$$P(h_j|v) = \varphi\left(\sum_i w_{ij} v_i - b_j\right) \tag{9}$$

where $\varphi(x) = \frac{1}{1+e^{-x}}$ is the logistic function, b_i is the bias for visible unit, b_j is the bias for hidden unit, and $P(h_j|v)$ is the conditional probability of h_j given v. Subsequently, the states of the hidden neurons $\langle h_j \rangle_0$ are sampled. During the negative phase, the hidden layer is stochastically fired to reconstruct the visible data $\langle v_i \rangle_1$by sampling according to Eq. 10.

$$P(h_j|v) = \varphi\left(\sum_j w_{ij} h_j - b_i\right) \tag{10}$$

The hidden layer $\langle h_j \rangle_1$ is, subsequently, recomputed from the visible layer using Eq. 9. After which, the weights and biases are updated as follows:

$$w'_{ij} = w_{ij} + \epsilon\left(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_1\right) \tag{11}$$

$$b'_i = b_i + \epsilon\left(\langle v_i \rangle_0 - \langle v_i \rangle_1\right) \tag{12}$$

$$b'_j = b_j + \epsilon\left(\langle h_j \rangle_0 - \langle h_j \rangle_1\right) \tag{13}$$

$\langle\rangle_0$ denotes the states of the neurons before reconstruction, and $\langle\rangle_1$ denotes the states of the neurons after a single step reconstruction. The same procedure, including positive phase, negative phases, and weight updating, are repeated until the stopping criterion is reached. The summary of the overall training procedure using CD can be found in Fig. 6.

The adaptation of RBM into EDA under multi-objective framework (MORBM) has been performed by Tang et al. (2010), and its sampling investigation has been carried out by Shim et al. (2010). For this network, the probabilistic model is constructed by considering the energy value of the network according to the following equation:

$$Pr_g(x_i = c_j) = \frac{\sum_{p=1}^{n} \delta_p(x_i = c_j) + Z_i/(N \times n)}{Z_i + Z_i/N}$$
$$\delta_p(x_i = c_j) = \begin{cases} \sum_{h=1}^{H} e^{-E(v=c_j,h)} & \text{if } x_i = c_j \\ 0 & \text{otherwise} \end{cases} \quad (14)$$
$$Z_i = \sum_{x,y} e^{-E(x,y)}$$

where v is the input state, h is the hidden state, Z is the normalizing constant, E is the energy value of the network as presented in Eq. 8, and N is the population size. Binary-representation is used in the original MORBM. Since integer-representation is utilized in this implementation, a decoding scheme from integer number to binary number is needed. 7 bits binary number is used to decode 100 cities, 8 for 200 cities, and 10 for 500 cities. In fact, any number of bits can be used as long as the possible states of the binary bits are enough to represent the number of cities. For example, 7 bits have 128 possible states, thus, is possible to decode 100 cities. After training, an encoding scheme is performed to obtain the probability of the cities in each position of the chromosome. Figure 5 shows the architecture of RBM and it adaptation in integer-representation. The overall modeling procedure is summarized in Fig. 7.

Begin
%%Network Training
Do While ("*maximum number of training epochs is not reached*")
%%Positive Phase
1. Construct the conditional probability of the hidden units given the input values $< v_i >_0$ according to eqn (9).
2. From $P(h_j|v)$, sample the states of the hidden units $< h_j >_0$.

%%Negative Phase
3. Reconstruct the states of the input units $< v_i >_1$ by sampling according to eqn (10).
4. Reconstruct again the states of the hidden units $< h_j >_1$ according to eqn (9).

%%Updating of weights
5. Update the weights and biases according to eqns (11)-(13).

End Do
End

Fig. 6 Pseudo code of CD training in RBM

Begin

1. Fetch the candidate solutions as input vectors into the visible units of the RBM
2. Decode the integer-representation of the cities into the binary-representation
3. Train the network using CD training mechanism to obtain the trained weights and biases according to the step described in Fig. 6.
4. Compute the energy value of the solutions according to eqn (8).
5. Compute the probability of the cities in binary-representation according to eqn (14).
6. Encode the binary-representation into integer-representation. Obtain the final probability distribution of the cities by taking average of the probability computed in step 5.

End

Fig. 7 Probabilistic modeling using RBM

3.3.4 Sampling

All EDAs considered in this paper will output the probability distribution in the similar form, which is the probability distribution of the cities to be located in each position of the chromosome. Therefore, the same sampling procedure can be applied to all EDAs. Offspring are generated by sampling the computed probability distribution, according to the following equation:

$$X_j = \begin{cases} C_1 & \text{if } random(0,1) \le Pr_g(x_j = c_1) \\ C_2 & \text{if } Pr_g(x_j = c_1) < random(0,1) \le \sum_{i=1}^{2} Pr_g(x_j = c_i) \\ \vdots & \\ C_n & \text{if } \sum_{i=1}^{n-1} Pr_g(x_j = c_i) < random(0,1) \le \sum_{i=1}^{n} Pr_g(x_j = c_i) \end{cases} \tag{15}$$

where X_j is a newly generated city at jth position of a chromosome, $random(0,1)$ is a randomly generated value between 0 and 1, and C_n is the city n.

3.3.5 Genetic algorithm (GA)

For genetic algorithm (Deb 2001), its variation operators are crossover and mutation. Single point crossover is used to create the offspring. This operator randomly selects the position to cut the chromosomes for crossing over between two parents. This single point crossover is equivalent to route inter-crossing. Through this biological process, the offspring will inherit some properties of the parent solutions. After which, mutation is carried out by swapping between two randomly selected alleles within the chromosome. This genetic perturbation provides exploitation capability to the optimizer to search within fitter region.

3.4 Feasibility correction

One of the problems after reproduction is that some cities may not be visited at all, while others are visited more than once. For EDAs, this is the result of the sampling mechanism which is based on the probability information of the cities. However, this sampling mechanism does not consider which city has or has not been included in the route. The same problem occurs in the GA, where the recombination of the

alleles between two parents is done indiscriminately. To overcome this problem, the simple permutation correction can be used. First of all, this correction identifies the repeated cities and the unvisited cities. The repeated cities are subsequently deleted and randomly replaced by the unvisited cities. However, this methodology is only able to satisfy the constraints of the problem, but it does not provide any extra information to guide the search. To fully utilize the available information from the database (travelling distance and travelling cost between cities), a refinement operator is proposed. Firstly, the repeated cities in a chromosome are detected, which will be followed by the recording of the unvisited cities. In this case, the number of repeated and unvisited cities should be the same. An insertion step is then carried out by inserting the unvisited cities to the position of the repeated cities. The average distance and cost (*Score* 1 as depicted in step 3 of Fig. 8) between the adjacent cities in the permutation are calculated and they will serve as the main criterion for insertion. For example, if the first unvisited city has a smaller average value of the travelling distance and cost compared to those of the second unvisited city at the first position of the repeated cities, then this city is inserted to the first position and the second unvisited city is inserted to another position. The pseudo code of the refinement operator is presented in Fig. 8.

3.5 Heuristic local search operator

As seen from literature, much research has been carried out to prove the advantages of using local search operator to enhance the exploitation capability of the EAs (Larrañaga and Lozano 2001; Tan et al. 2007; Ong et al. 2010). Their attempts rely largely on the available knowledge of the database (distance, cost, time, etc). With n cities, the TSP will have $(n - 1)/2$ routes. The huge search space of a TSP poses a challenge for optimizers in finding the shortest distance. However, the difficulty of the problem is alleviated when the domain knowledge is taken into consideration during optimization process. In this implementation, a local search operator is proposed. It makes use of the shortest distance and lowest cost among the cities. The process flow of this operator is as follows. Firstly, the number of cities to be

Begin
For $\boldsymbol{p = 1:N}$
1. Identify the location of the repeated cities in chromosome p, $L_i = L_1, \ldots, L_R$.
2. Identify the unvisited cities in chromosome p, $V_j = V_1, \ldots, V_R$.
For $\boldsymbol{i = 1:R}$
3. Calculate the score for all unvisited cities V_j in location L_i according to the following equation:

$$Score1\ (j) = \frac{\sum_{q=1}^{m} D^q\left(V_j,\ x_{L_i+1}\right) + D^q\left(V_j,\ x_{L_i-1}\right)}{m}$$

4. Insert a city in V_j which has the smallest score, to the $L_i{}^{th}$ position of the chromosome. Discard that city from V_j.
End For $\boldsymbol{i}$
End For $\boldsymbol{p}$
End

Fig. 8 Pseudo code of the refinement operator

relocated (k) is pre-defined. Then, a position ℓ is randomly determined. From this position, a sequence of k cities is being selected. A score based on the distances and costs (*Score* 2 as depicted in step 3 of Fig. 9) among all the selected cities are then calculated. The permutation of the cities is re-determined according to the *Score* 2. The pseudo code of the local search operator is presented in Fig. 9.

3.6 Algorithms' framework

The algorithmic process flow of the proposed algorithm is shown in Fig. 10. Firstly, the optimizers read the database which contains the coordinates of the cities and the traveling cost between the cities. A $n \times n$ distance matrix from city i to city j is constructed by computing the Euclidean distance between the cities. Subsequently, population initialization is performed by generating the permutations of the cities randomly. All solutions in the population are evaluated according to Eq. 1 to obtain the objective values. Fitness is then assigned to all the solutions in the population based on Pareto-based ranking and crowding distance. Next, binary tournament selection is applied to determine a set of N promising candidate solutions. In the selection process, two chromosomes are randomly picked into tournament. A fitter solution in terms of smallest rank or greatest crowding distance is selected. This selection procedure is repeated until the population size is reached. Subsequently, the probabilistic model of the candidate solution set is built. The probabilistic model is constructed using Eqs. 6, 7, or 14 according to which algorithm is performed. Based on the constructed model, sampling is carried out to produce N offspring according to Eq. 15. For GA, crossover and mutation are performed instead of probabilistic modeling and sampling to produce offspring. After reproduction, some cities may be visited more than once while other are never visited. Therefore, the refinement operator is implemented to correct the reproductive law. After this process, the chromosomes should satisfy the constraint of the MOTSP, where each city is strictly visited once. To further improve the routing, a local search operator is incorporated. The local search will only be performed if the generated random value

Begin

1. Pre-define the number of cities to be relocated, k .

For $\boldsymbol{p = 1: N}$

%If local search is performed

2. Start at the position ℓ (randomly determined) of a chromosome, and then select a sequence of k cities U_j, where $\ell + k - 1 \leq n$.

For $\boldsymbol{q = 1: k}$

3. Calculate the score for all cities in the selected sequence, $U_j = U_1, \ldots, U_k$

$$Score2\ (j) = \frac{\sum_{h=1}^{m} D^h\left(U_j,\ x_{\ell-2+q}\right)}{m}$$

4. Insert a city in U_j which has the smallest score to the $(\ell - 1 + q)^{th}$ position of the chromosome. Discard that city from U_j.

End For $\boldsymbol{q}$

End For $\boldsymbol{p}$

End

Fig. 9 Pseudo code of the local search operator

is smaller than a predefined local search rate *LS*. Subsequently, all offspring are evaluated to obtain the objective values. After this, an archive is created to store the parent and offspring found during the evolution. The solutions in the archive are re-ranked and the crowding distance of all solutions is re-computed. Following the ranking criteria, *N* solutions with smallest rank or greatest crowding distance are chosen. This is an elitism mechanism which may enhance the convergence speed and generate better solutions. This marks the completion of one generation. The same procedure is iterated over generations until the stopping criterion is met. The algorithmic flow of GA is similar to EDAs, except that the modeling and sampling mechanisms in EDAs are replaced by crossover and mutation. For hybrid algorithms, they start with EDA and alternates with GA every 500 generations.

4 Experimental results and discussions

All algorithms were written in C ++ and the experimental studies were run on an Intel(R) Core(TM)2 Duo CPU, 3.0 GHz. The experimental settings are shown in Table 2. In this work, TSP with two objectives is studied. The coordinate of the

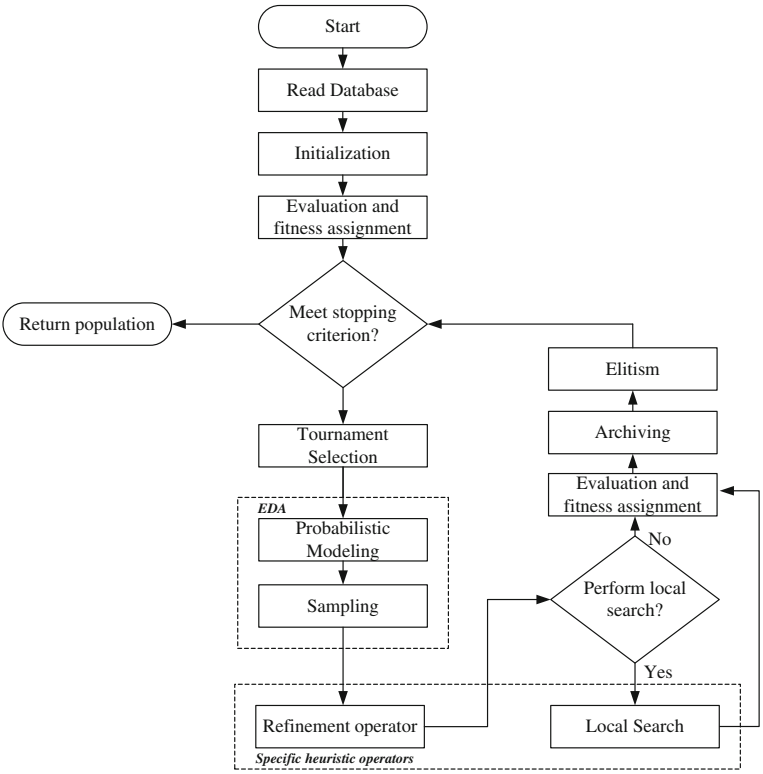

Fig. 10 Process flow of the MOEDAs

Table 2 Parameter settings for experiments

Parameter	Value
Population size (N)	Number of cities
Archive size (2N)	2 × Number of cities
Number of cities	100, 200, 500
Fitness evaluations	2000N
Local search rate (*LS*)	0.5
Frequency of alternation (*fr*)	500
Crossover rate in GA	0.8
Mutation rate in GA	0.05
Independent runs	10
α in PBIL	0.9
Hidden unit in RBM (*H*)	10
Training epoch in RBM	10
k	10

cities and the travelling cost between the cities are randomly generated for each objective in the range of [0, 1000] as done in Peng et al. (2009).

For experimental studies, the results are compared based on the performance metrics of inverter generational distance (IGD) (Villalobos-Arias et al. 2005), generational distance (GD) (Velhuizen and Lamont 1998), maximum spread (Zitzler et al. 2000), and non-dominance ratio (NR) (Goh and Tan 2009). IGD measures the average Euclidean distance between each solution on the approximate Pareto optimal front to the nearest solution on the evolvable front. This measurement takes into consideration the proximity as well as diversity information. A smaller IGD value implies better performance in terms of closer proximity of the evolvable front to the approximate Pareto optimal front and a wider distribution of the evolved front along the approximate Pareto optimal front. GD is the original version of IGD where only proximity is considered. Different from IGD, GD measures the average Euclidean distance between each solution on the evolvable front to the nearest solution on the approximate Pareto optimal front. Therefore, a smaller GD value shows that the evolvable front is closer to the approximate Pareto optimal front. On the other hand, MS only considers the diversity measurement. This metric measures the distribution of the evolvable front on the approximate Pareto optimal front. In this case, a higher value of MS will mean that the evolvable front has better coverage along the approximate Pareto optimal front. Since the optimal solution set to the problem is unknown, the approximate optimal solution set is formed using the non-dominated solutions found from all algorithms and all experimental runs, as has been done in (Peng et al. 2009). Lastly, NR measures the ratio of non-dominated solutions among all the solutions generated by all the algorithms. A higher value of NR means that the algorithm produces more non-dominated solutions.

In this section, the empirical comparison of the EDAs, GA, and the hybrid algorithms are carried out. The hybridization of EDA and GA start off with EDAs, and each algorithm is being alternated every 500 generations. Next, the effect of the permutation refinement operator, local search operator, and number of alternations

in the hybrid algorithms are investigated. Lastly, computation time analysis is presented. In total, seven algorithms are put into comparison. For easier readability of the paper, the abbreviations of the algorithms are illustrated in Table 3.

4.1 Comparison result

Figure 11 shows the performance metric of IGD, GD, MS, and NR after 200,000 fitness evaluations for MOTSP with 100 cities. From the IGD performance indicator, it is observed that EDAs (RBM, UMDA, and PBIL) outperform GA. Among the different EDAs, the performance of UMDA and PBIL is better than RBM. When hybridization is carried out, the performance of RBM (RBM-GA) is improved. However, no significant improvement is observed in UM-GA and PB-GA. In order to understand this observation, the results in terms of GD and MS are plotted. The GD results show the parallel observation as IGD. These observations suggest that the EDAs are able to generate a set of closer evolvable solution set to the approximate Pareto optimal front. Since the EDAs and GA implemented in this experimental study utilize similar operators except for the reproduction operators, the good performance of the EDAs is most probably attributed to the incorporation of probabilistic information in guiding the search.

In terms of MS performance indicator, the performance of GA is far better than EDAs. This implies that the diversity preservation in EDAs is poor. This may be attributed to the fact that EDAs only model the certain regions of the promising solutions. Throughout the evolution process, the modeled region may be reduced which results in poor solution diversity preservation. Furthermore, EDAs do not utilize any location information. This may cause the algorithms to be unable to explore the search regions which are further away from the modeled regions. When EDAs is hybridized with GA, the performance of EDAs in terms of MS is greatly improved, as shown by a higher value of MS obtained by RBM-GA, UM-GA, and PB-GA. This is the main reason why the hybridization of the EDAs and GA is done. EDAs are able to obtain a set of solutions which is closer to the approximate Pareto optimal front; whereas GA proves to be good in terms of maintaining the diversity of the solution set. Thus, the hybridization is expected to complement the strong points of both algorithms so as to overcome their limitations. As for NR

Table 3 Algorithms' abbreviation

Algorithms' abbreviation	Description
RBM-GA	A hybrid algorithm of RBM and GA
UM-GA	A hybrid algorithm of UMDA and GA
PB-GA	A hybrid algorithm of PBIL and GA
RBM	Restricted Boltzmann machine
UMDA	Univariate marginal distribution algorithm
PBIL	Population-based incremental learning
GA	Genetic algorithm

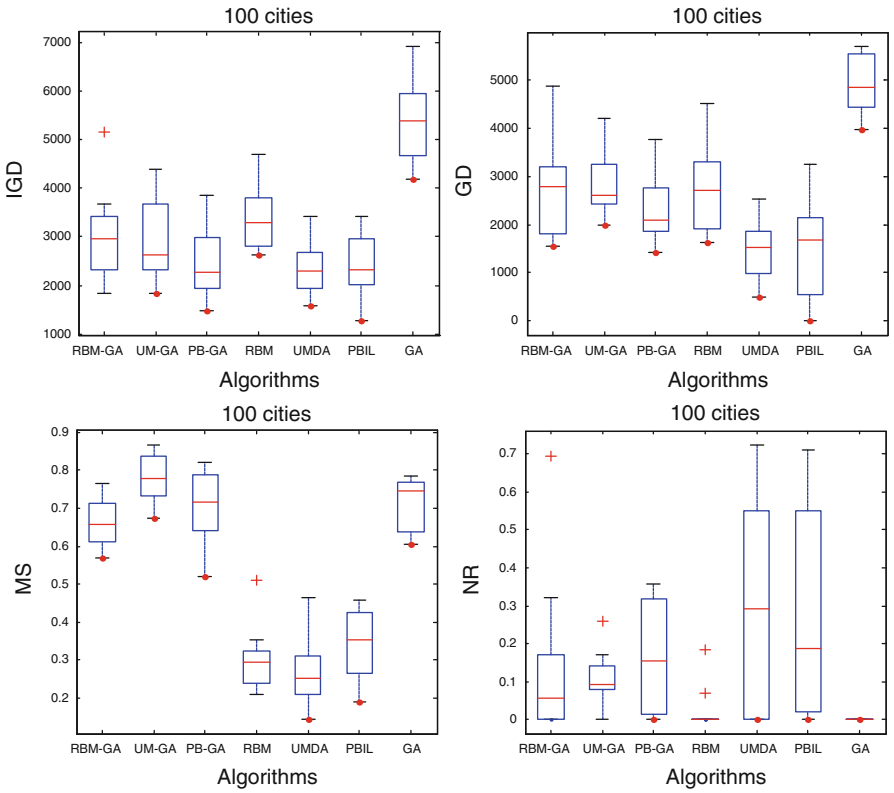

Fig. 11 Performance metric of IGD, GD, MS, and NR after 200,000 fitness evaluations for MOTSP with 100 cities. *x*-axis is the algorithms and *y*-axis is the value of the performance indicators

performance indicator, it is observed that most of the non-dominated solutions are generated by UMDA and PBIL, and then followed by PB-GA, RBM-GA, UM-GA, and RBM. The performance of GA is the worst.

In order to visualize how the evolvable solutions are distributed in the objective space, Fig. 12 plots the non-dominated solutions generated by each of the algorithm. For clearer visualization, only 50 randomly picked points of non-dominated solutions are plotted. It is observed that GA is able to produce a set of diverse solutions but they are inferior in terms of proximity. On the other hand, EDAs is able to evolve a set of solutions which is closer to the approximate Pareto optimal front but the diversity of the solution set is poor. The hybridization of EDAs and GA seems to improve the performance of the algorithms (RBM-GA, UM-GA, and PB-GA) in terms of proximity of the evolvable front to the approximate Pareto optimal front and distribution of the solutions on the approximate Pareto optimal front.

Figure 13 shows the evolution trace curves for IGD, GD, MS and NR performance indicators from the early stages of evolution up to the 200,000 fitness evaluations for MOTSP with 100 cities. It is observed that RBM has the best

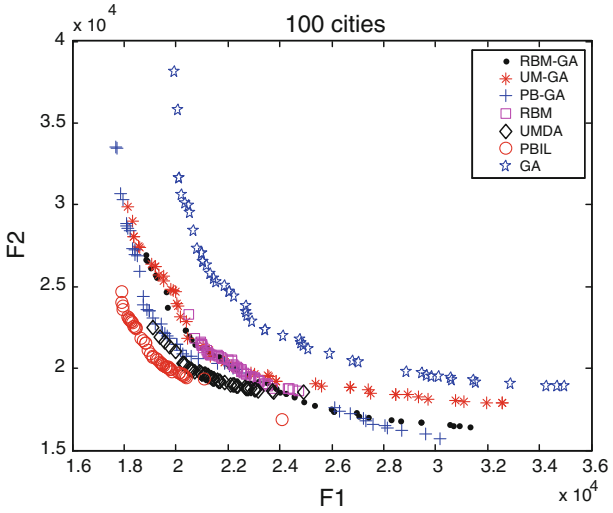

Fig. 12 Final evolvable front generated by the various algorithms for MOTSP with 100 cities. F1 is the first objective or travelling distance and F2 is the second objective or travelling cost

convergence speed at the early stages of evolution (in terms of IGD and GD performance indicators). However, the convergence speed of RBM is decreased after about 500 generations. This is most probably caused by the modeling mechanism of RBM in whereby only certain promising regions are being modeled. RBM constructs the probability distribution of the candidate solutions by considering their linkage dependencies. This information is useful in estimating of the search direction, which therefore promotes the convergence speed. Throughout the evolution process, the modeled regions will become smaller. Since there is no presence of any enhanced diversity preservation mechanism in RBM, therefore it is hard for the RBM to further explore other promising regions. Eventually, the convergence speed is decreased during the later evolutions. For the other EDAs, the building block of a good schema is harder to build than RBM since only independent probability is considered in their probabilistic modeling. The stochastic recombination of GA also provides fast convergence speed at the early stages of evolution. However, the convergence speed is decreased when it reached to about 100 generations.

In terms of MS performance indicators, the hybrid algorithms exhibit significant change after 500 generations, which is the condition that the EDAs are alternated to GA. The MS values gradually improve throughout the evolution process. A higher value of MS indicates that the evolvable solution set is highly distribution along the approximate Pareto optimal front. This is important as poor solution diversity will mean that most of the optimal trade-off solutions are not being found. In real world conditions, a manager will have more options in their decision making when a set of diverse solutions are provided. In terms of NR performance indicators, it is observed that most of the non-dominated solutions are generated by UMDA and PBIL. This is because both of the algorithms only focus their search within a particular region.

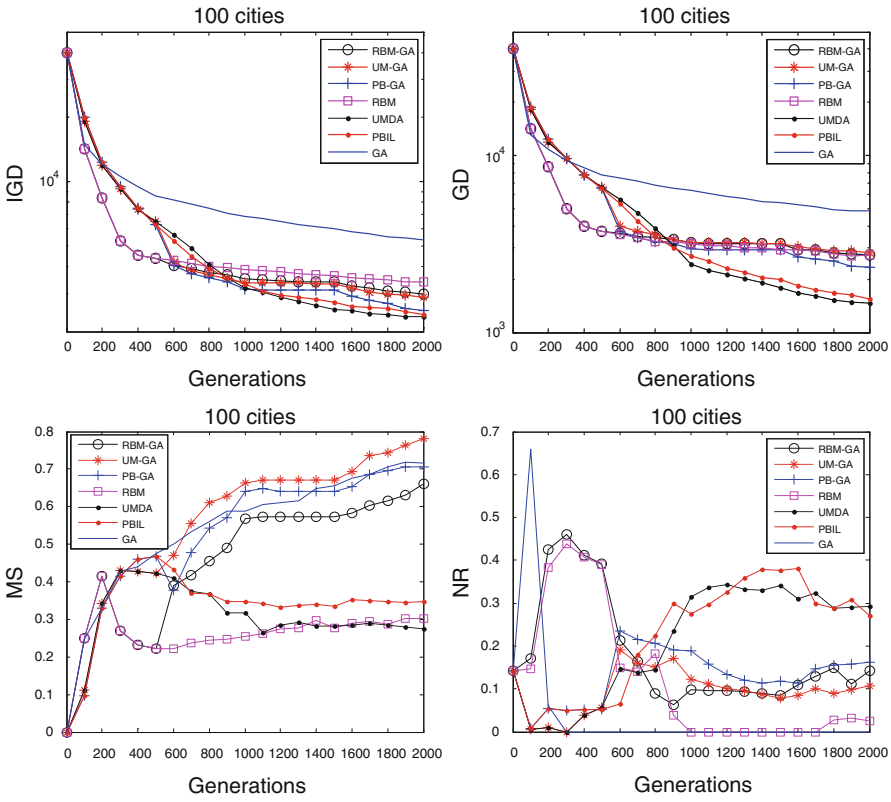

Fig. 13 Evolution trace of IGD, GD, MS, and NR performance indicators for MOTSP with 100 cities. *x*-axis is the number of generations and *y*-axis is the value of the performance indicators

Thus, they could find more non-dominated solutions, but poor solution diversity. The hybridization of EDAs and GA improve the ability of the algorithms to further explore and exploit the search space, rectifies the limitation of both of the algorithms.

In order to test the search ability of the algorithms in problem with larger search space, experimental studies were carried out to solve the MOTSP with 200 cities. Performance metric of IGD, GD, MS, and NR at 400,000 fitness evaluations are presented in Fig. 14. From the performance indicators of IGD and GD, it is observed that the performances of RBM and hybrid algorithms are better as compared to the PBIL, UMDA, and GA. The linkage information is beneficial to the RBM as it helps in the algorithm in the estimation of the probability distribution of the cities in the position of the chromosome, and hence this leads to the achievement of better results. This is its main difference with the UMDA and PBIL whereby independent probability distribution of the cities is being considered. As for GA, the stochastic behaviors of the variation operators are dependent on the chance of recombination, thus it is not possible to predict the direction of the motion. The hybridization of EDAs and GA complement the strong points of each other, and this

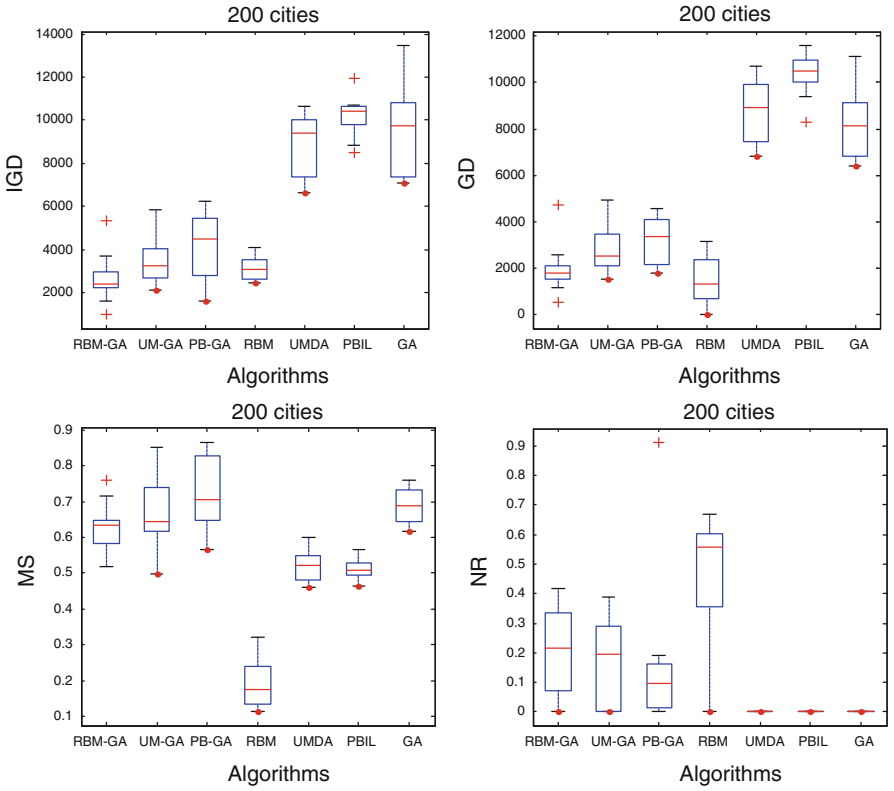

Fig. 14 Performance metric of IGD, GD, MS and NR after 400 000 fitness evaluations for MOTSP with 200 cities. *x*-axis is the algorithms and *y*-axis is the value of the performance indicators

helps to overcome the limitations of both algorithms, which results in the hybrid being able to outperform the original algorithms. In terms of NR performance indicators, it is observed that RBM evolves most of the non-dominated solutions, and followed by RBM-GA, UM-GA, and PB-GA. However, no non-dominated solution is being generated by UMDA, PBIL, and GA. Overall, RBM performed well in terms of proximity results, but poor solution diversity. This can be further visualized in Fig. 15, where the final non-dominated evolvable solutions obtained by the various algorithms are plotted.

Figure 16 plots the evolution trace of IGD, GD, MS, and NR performance indicators for MOTSP with 200 cities. Similar observation can be made for RBM, in terms of IGD and MS performance indicators, that it has the fastest convergence speed at the early stages of evolution. Furthermore, the convergence speed of the RBM is enhanced when it is hybridized with GA, as shown in IGD, GD, and MS performance indicators. Generally, in all of the performance indicators, the performance of the hybrid algorithms and RBM are better when compared to the UMDA, PBIL, and GA. In terms of NR performance indicator, GA is only able to maintain a set of non-dominated solutions at the early stages of evolution. However,

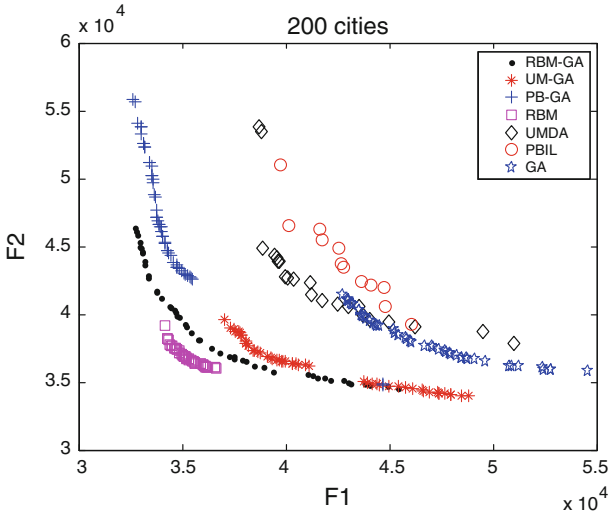

Fig. 15 Final evolvable front generated by the various algorithms for MOTSP with 200 cities. F1 is the first objective or travelling distance and F2 is the second objective or travelling cost

the NR for GA on the declining trend over the generations. At the end of the evolution, most of the solutions evolved by the RBM dominate the solutions evolved by the other six algorithms.

Performance metric of IGD, GD, MS, and NR for MOTSP with 500 cities are presented in Fig. 17. It is observed that the performance of RBM-GA, UM-GA, and PB-GA, in terms of IGD, GD, and MS, are better than RBM, UMDA, PBIL, and GA. Among the different hybrid algorithms, RBM-GA and UM-GA have almost similar performance. Other observations that can be noted are that the performance of EDAs is inferior when compared to the GA; and also the performance of PBIL is the worst. In this problem, the search space is huge. When the modeling of EDAs only emphasize on certain regions, the diversity of the maintained solution set will be poor. Therefore, this leads to the poor performance of EDAs.

One way to cope with this problem is to divide the promising regions into several clusters and then a probabilistic model is built in every cluster. The clustering can be carried out in decision (Pelikan et al. 2005) or objective (Tang et al. 2010) domain. Sampling is subsequently carried out to sample equal number of solutions from each cluster. The generated solutions are then combined to form the new set of offspring. In this paper, another option to cope with the problem is proposed, which is through hybridization of EDAs and GA. These results show that the hybridization can enhance the performance of the original algorithms. In terms of MS performance indicator, it is observed that the diversity of the evolvable solution sets on the approximate Pareto optimal front is greatly improved. This is because the EDAs and GA have their own strengths and weaknesses, in which the performance of EDAs is superior in exploring the promising search regions, while GA is superior in searching for a set of diverse solutions. With these features, the hybridization is able

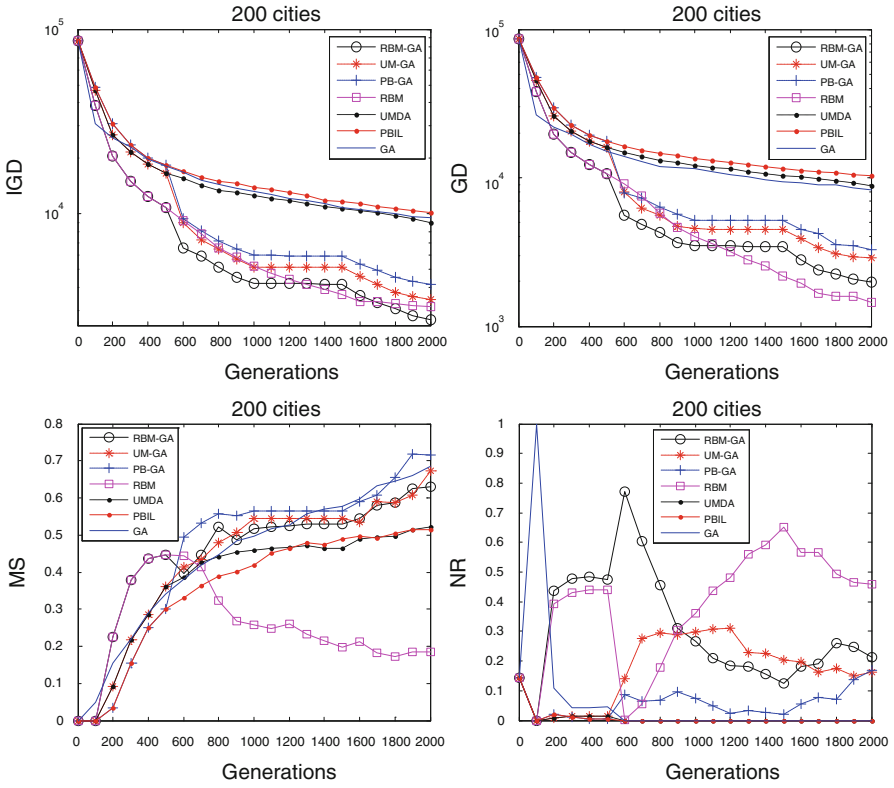

Fig. 16 Evolution trace of IGD, GD, MS and NR performance indicators for MOTSP with 200 cities. *x*-axis is the number of generations and *y*-axis is the value of the performance indicators

to complement each other so as to overcome their individual limitations, and thus the hybrid algorithms are able to outperform other algorithms. At the end of the evolution, most of the non-dominated solutions are evolved by UM-GA and followed by RBM-GA. As for the other algorithms, they fail to obtain any number of non-dominated solutions. The distributions of the non-dominated solutions evolved by the various algorithms are shown in Fig. 18.

Figure 19 shows the evolution trace of IGD, GD, MS and NR performance indicators for MOTSP with 500 cities. The results obtained in this problem are quite similar to those obtained in problem with 100 and 200 cities. GA has the fastest convergence speed at the early stages of evolution, but the speed is decreased after the algorithm is run up to 100 generations. As for the RBM, its convergence speed outperforms GA when it is run up to 200 generations. Generally, hybrid algorithms are able to achieve better performance, in terms of proximity, diversity, and convergence speed, when compared to the other algorithms. Specifically, RBM-GA and UM-GA display the best performance while the performance of the PBIL is the worst.

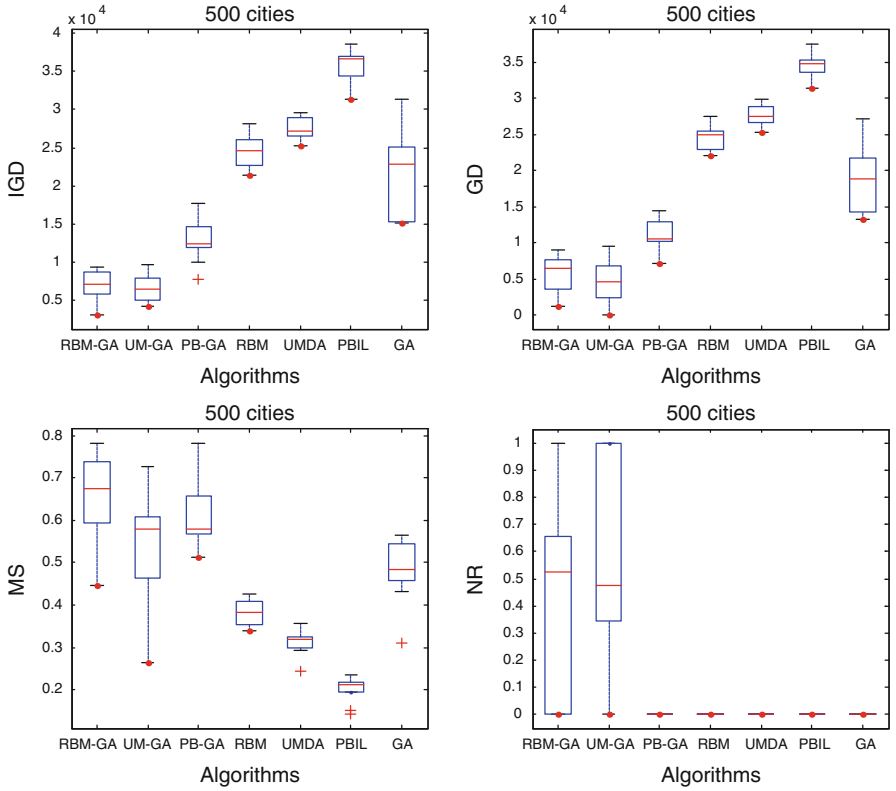

Fig. 17 Performance metric of IGD, GD, MS and NR after 1,000,000 fitness evaluations for MOTSP with 500 cities. *x*-axis is the algorithms and *y*-axis is the value of the performance indicators

4.2 Effect of feasibility correction operator

The aim of the feasibility correction operator is to refine the sequence in a chromosome in order to guarantee that no repeated visit on any city in the route of the salesman. Two feasibility correction operators are described in Sect. 3.4, namely permutation correction operator and permutation refinement operator. In permutation correction operator, the unvisited cities are randomly inserted into the positions of the repeated cities. On the other hand, permutation refinement operator makes use of the available information from the database (travelling distance and travelling cost between the cities) to define the insertion and deletion rules.

In order to investigate the benefit of using the permutation refinement operator instead of the permutation correction operator, the experimental results in terms of IGD performance indicators after running the various algorithms with permutation refinement operator or permutation correction operator on MOTSP with 100 cities and 200 cities are presented in Table 4.

It is observed that, in all algorithms, the performance of the algorithms with permutation refinement operator outperforms the algorithms with permutation correction operator. This set of results suggests that the utilization of problem

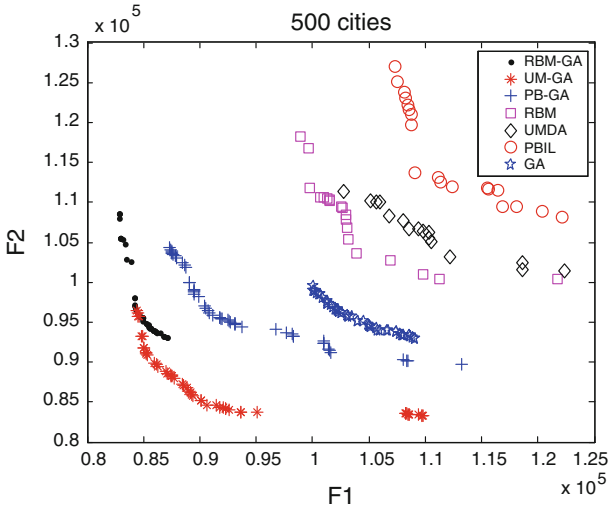

Fig. 18 Final evolvable front generated by the various algorithms for MOTSP with 500 cities. F1 is the first objective or travelling distance and F2 is the second objective or travelling cost

domain knowledge is able to enhance the searching task. In the case of EDAs, the diversity of the solution set is exhausted. Therefore, the permutation refinement operator is importance as it serves as a technique to introduce some diverse solutions to the modeled region.

4.3 Effect of local search operator

In order to improve the ability of the algorithm to search for better solutions, a local search operator is proposed. This operator will randomly select a sequence of cities, and then relocating them according to the problem domain knowledge.

Figure 20 shows the IGD performance indicator obtained by RBM, UMDA, PBIL, and GA for MOTSP with 100 cities under various settings of local search rate ($LS = 0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$). It is observed that, for RBM, a LS of 0.5 gives the best performance. A lower local search rate ($LS = 0.1$ and 0.3) or a higher setting of LS slightly improve the performance of the algorithm compared to the one without local search ($LS = 0$). For UMDA, the best performance is obtained when $LS = 0.5$. Other settings of LS give random performance. No significant improvement is observed in PBIL. For GA, a larger value of LS gives better result, and the best performance was observed at $LS = 0.9$. Similar to permutation refinement operator, the proposed local search operator also utilizes the problem domain knowledge to guide the search, hence can further enhance the search.

4.4 Effect of frequency of alternation between EDAs and GA

EDAs which model the global distribution of the candidate solutions may be able to drive the search towards certain promising search regions. Since only global

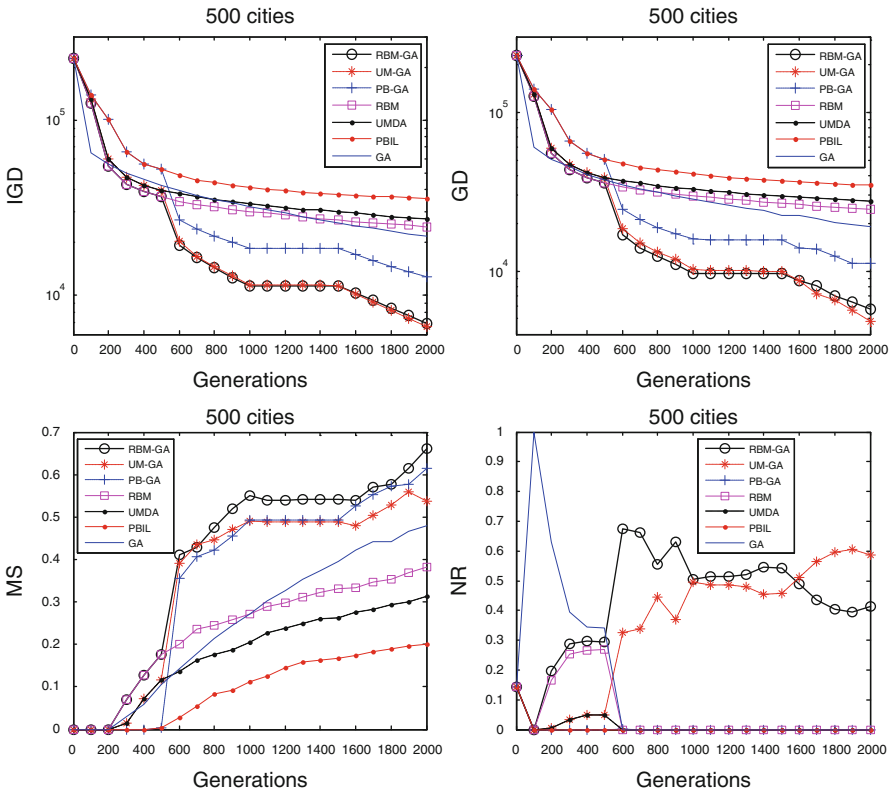

Fig. 19 Evolution trace of IGD, GD, MS and NR performance indicators for MOTSP with 500 cities. *x*-axis is the number of generations and *y*-axis is the value of the performance indicators

Table 4 Performance indicator of IGD after running the various algorithms with permutation refinement operator or permutation correction operator on MOTSP with 100 and 200 cities

Algorithms	100 cities		200 cities	
	P. Refinement	P. Correction	P. Refinement	P. Correction
RBM-GA	2998.6 ± 964.40	6878.7 ± 1086.4	2689.0 ± 1193.9	19560 ± 1455.9
UM-GA	2888.1 ± 831.30	7993.2 ± 1458.4	3456.0 ± 1101.2	21448 ± 1648.8
PB-GA	2504.0 ± 734.50	7881.7 ± 792.76	4161.0 ± 1571.9	20936 ± 4019.7
RBM	3418.8 ± 718.10	8391.7 ± 1212.6	3141.0 ± 587.00	30030 ± 10673
UMDA	2349.8 ± 552.70	8949.1 ± 3862.7	8955.0 ± 1459.1	67774 ± 5459.7
PBIL	2397.5 ± 696.40	8890.5 ± 3875.4	10205 ± 982.6	65984 ± 4161.8
GA	5371.0 ± 848.26	7493.6 ± 926.75	9530.0 ± 2098.5	15917 ± 2818.2

information is taken into consideration, the weakness of the algorithms is mainly on its poor exploration of the other search regions which are further away from the modeled regions. Thus, EDAs demonstrate poor solution diversity as indicated in

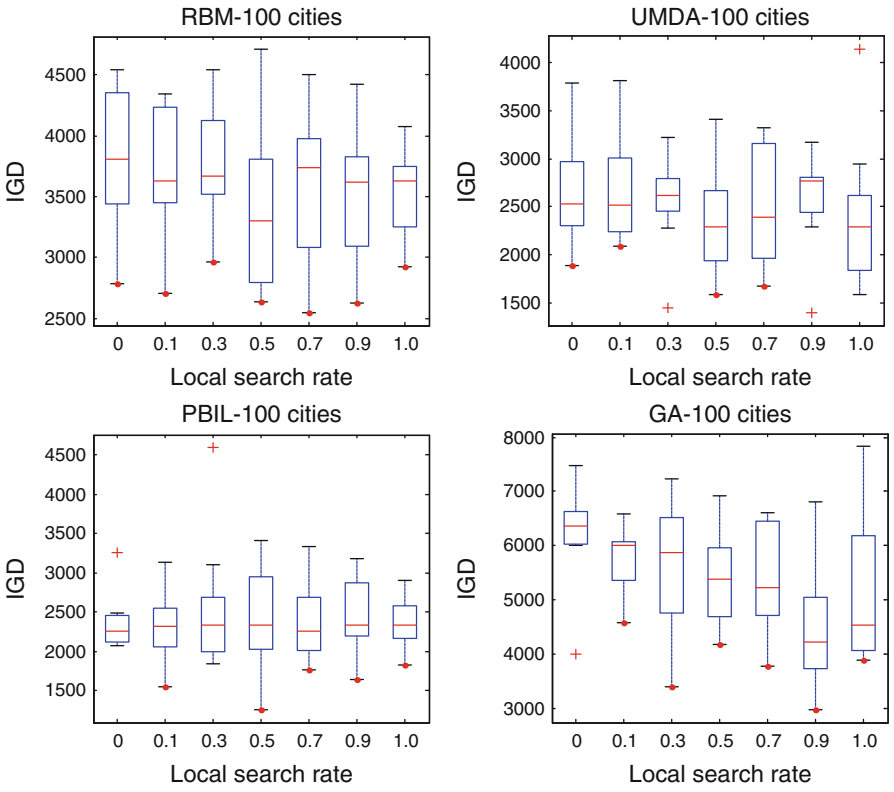

Fig. 20 Performance indicators of IGD obtained by RBM, UMDA, PBIL, and GA in MOTSP with 100 cities under different settings of local search rate. *x*-axis is the local search rate and *y*-axis is the IGD value

Sect. 4.1. In literature review, several approaches have been proposed to cope with this limitation. They include the introduction of the mutation operator (Zhong and Li 2007), clustering (Pelikan et al. 2005), and Parzen estimator (Costa and Minisci 2003) into EDAs.

In the preliminary experimental results, it was observed that GA is able to produce a set of diverse solutions in most of the experimental runs. However, the performance of GA, in terms of proximity of the evolvable solutions to the approximate Pareto optimal front, is inferior when compared to EDAs. This observation sparked that the hybridization of EDAs and GA can create a synergy that can ameliorate the limitation of both algorithms.

In this study, the hybridization is carried out by firstly running the EDAs and then alternates to GA every pre-defined number of generations. Alternation with frequency of every 500 generations is implemented in the above experimental studied. In this section, experimental studies are carried out with various frequency of alternation (*fr*) including $fr = 1$, 10, 100, 200, 500, 1000, 1500, and 2000. Since the computing budget is set to 2000 generations, which is equal to 2000 N fitness

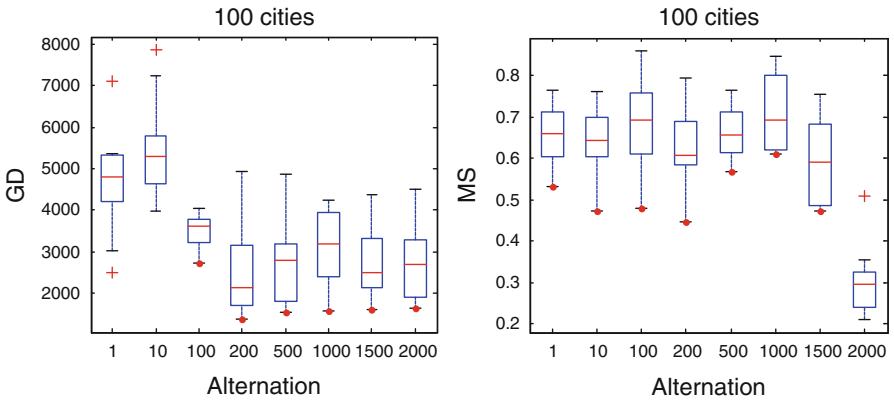

Fig. 21 Performance indicator of GD and MS obtained by RBM-GA for MOTSP with 100 cities under different settings of the frequency of alternation *fr*. *x*-axis is the frequency of alternation and *y*-axis is the value of the performance indicators

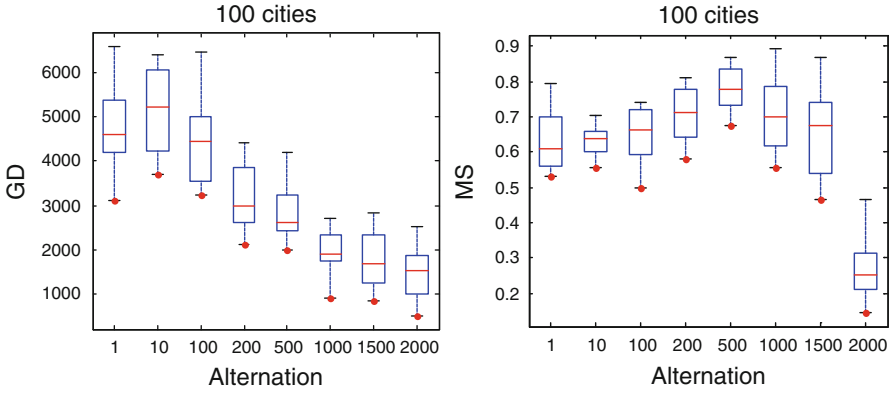

Fig. 22 Performance indicator of GD and MS obtained by UM-GA for MOTSP with 100 cities under different settings of the frequency of alternation *fr*. *x*-axis is the frequency of alternation and *y*-axis is the value of the performance indicators

evaluations, the *fr* of 2000 means that no alternation from EDAs to GA is occur. Since the aim of the hybridization is to improve the poor solution diversity in EDAs, only results in terms of GD and MS will be looked into and presented.

Figures 21, 22, 23 show the experimental results of GD and MS performance indicators obtained by RBM-GA, UM-GA, and PB-GA, respectively, with respect to the various frequency of alternation *fr*. In terms of GD performance indicators, it is observed that the performance of the hybrid algorithms are inferior when the fast alternation are performed ($fr = 1$, 10, and 100). For a larger value of *fr* ($fr > 100$), the GD performance is better, as indicated by a lower value of GD. This may be attributed to the fact that EDAs and GA may build different building block of schemas during the course of evolution. The fast alternation may prevent the

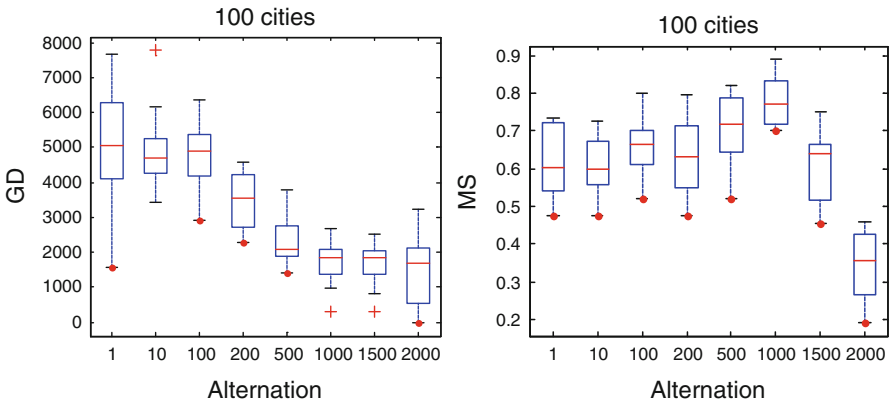

Fig. 23 Performance indicator of GD and MS obtained by PB-GA for MOTSP with 100 cities under different settings of the frequency of alternation *fr*. *x*-axis is the frequency of alternation and *y*-axis is the value of the performance indicators

building up of the good genetic schemas; thus, the failure to generate better solutions.

In terms of MS performance indicator, a common observation was made for all sets of results. It is observed that the diversity of the solution set obtained by all algorithms is poor when no alternation is applied ($fr = 2000$). When alternation between EDAs and GA are carried out, great improvements are observed in all settings of *fr*. This observation strengthens the idea that the diversity of the solution set evolved by EDAs can be improved by hybridizing EDAs and GA in the proposed alternating manner. Similar results are observed for MOTSP with 200 cities, hence the results will not be presented.

4.5 Computational time analysis

Comparing between GA and EDAs, GA applies genetic operators which only involves cut and insert procedures while EDAs implement the probabilistic modeling approaches to estimate the set of approximate Pareto optimal solutions. Thus, the different optimization schemes adapted in GA and EDAs will result in different computational time needed. Table 5 presents the computational time required by one experimental run of the various algorithms in terms of solving MOTSP with settings of 100, 200, and 500 cities. From the table, the computational time required by the EDAs is generally higher than GA. This is due to the higher complexity in construction of the probabilistic models as compared to the genetic operators used in GA. Among the different EDAs, the RBM surfaces as the most time consuming one. This is attributed to the training stages in RBM which is performed in every generation. As for UMDA and PBIL, they consume lesser computational time than RBM because there is no training in them, and their probabilistic model is directly built from the raw data of the candidate solutions. When the problem size is increased from 100 to 500 cities, all algorithms require additional computational time to complete one experimental run.

Table 5 Computational time (in second) used by the various algorithms for solving MOTSP with 100, 200, and 500 cities

Algorithms	100 cities	200 cities	500 cities
RBM-GA	628.33 ± 14.460	2994.5 ± 7.3088	35486 ± 511.92
UM-GA	27.805 ± 1.0947	194.28 ± 3.1202	2987.4 ± 136.29
PB-GA	27.810 ± 0.8897	199.71 ± 3.7931	3141.1 ± 145.50
RBM	2689.7 ± 91.703	10481 ± 159.82	68792 ± 355.56
UMDA	28.714 ± 0.9036	332.55 ± 8.1735	4648.8 ± 30.824
PBIL	29.167 ± 1.3288	337.84 ± 6.6643	4836.5 ± 25.803
GA	19.005 ± 0.1993	124.98 ± 4.2990	1665.6 ± 43.812

5 Conclusions

This paper has studied the potential of EDAs and GA in solving MOTSP with different number of cities. It is among the first attempts to employ EDAs in the study of permutation-based multi-objective optimization problems, specifically the MOTSP. Three EDAs, including univariate marginal distribution algorithm, population-based incremental learning, and restricted Boltzmann machine, have been considered. The EDAs are altered to handle the problem using integer-representation. A permutation refinement operator has been proposed to make sure the constraint of the problem is fulfilled. In addition, a heuristic-based approach of local search was defined to enhance the search ability of the algorithms. As the limitation of EDAs in evolving a set of solutions with good diversity, EDAs have been hybridized with GA in order to complement their weaknesses. The effectiveness of the seven algorithms has been experimentally studied under TSP with two objective and different number of cities.

From the results obtained in the comparative studies, the EDAs was able to achieve better proximity results and GA was success in maintaining a set of diverse solutions. The hybridization of EDAs and GA mutually complements each other's limitation, thus yielding better performance. Among the different hybrid algorithms, RBM-GA performed the best, and followed by UM-GA and PB-GA. However, RBM-GA incurred additional computational time in its modeling mechanism. Therefore, if the computational time is not the primary concern, RBM-GA is considered the best algorithms. However, if the computational time is taken into consideration, UM-GA is preferred.

Since the global Pareto optimal solutions for the problems are unknown, as in the case of most real world optimization problems, the solution's quality is compared in relation to other solutions obtained from the various algorithms. GA is served as the benchmark algorithm for comparison since the performance of GA in MOTSP has been proven to be acceptable, if not superb. Therefore, it is concluded that the performance of the EDAs is superior to GA in terms of proximity and convergence speed, and the hybrid algorithms outperform the original algorithms.

References

Aarts EHL, Korst JHM (1989) Boltzmann machines for travelling salesman problems. Eur J Oper Res 39(1):79–95

Arel I, Rose DC, Karnowski TP (2010) Deep machine learning–a new frontier in artificial intelligence research. IEEE Comput Intell Mag 5(4):13–18

Bagchi TP, Gupta JND, Sriskandarajah C (2006) A review of TSP based approaches for flowshop scheduling. Eur J Oper Res 169(3):816–854

Baluja S (1994) Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Technical Report: CS-94-163

Blazewicz J, Kasprzak M, Kuroczycki W (2002) Hybrid genetic algorithm for DNA sequencing with errors. J Heuristics 8(5):495–502

Bosman PAN, Thierens D (2002) Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. Int J Approx Reason 31(3):259–289

Chen S, Chen M, Zhang P, Zhang Q, Chen Y (2010) Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. Expert Syst Appl 37(9):6441–6451

Cheong CY, Tan KC, Liu DK, Lin CJ (2010) Multi-objective and prioritized berth allocation in container ports. Ann Oper Res 180(1):63–103

Coello Coello CA (2006) Evolutionary multi-objective optimization: a historical view of the field. IEEE Comput Intell Mag 1(1):28–36

Costa M, Minisci E (2003) MOPED: a multi-objective Parzen-based estimation of distribution algorithm for continuous problems. In: Second international conference evolutionary multi-criterion optimization, pp 282–294

Da San Martino G, Sperduti A (2010) Mining structured data. IEEE Comput Intell Mag 5(1):42–49

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE T Evolut Comput 6(2):182–197

Diaby M (2007) The traveling salesman problem: a linear programming formulation. T Math 6(6):745–754

Eiselt HA, Laporte G (1991) A combinatorial optimization problem arising in dartboard design. J Oper Res Soc 42(2):113–118

Elaoud S, Teghem J, Loukil T (2010) Multiple crossover genetic algorithm for the multi-objective traveling salesman problem. Electron Notes Discrete Math 36:939–946

García-Martínez C, Cordón O, Herrera F (2004) An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), pp 61–72

Goh CK, Tan KC (2009) A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE T Evolut Comput 13(1):103–127

Gonzales C (2005) Contributions on theoretical aspects of estimation of distribution algorithms. Dissertation, University of the Basque Country

Herrera F, Garcia-Martinez C, Cordon O (2007) A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Eur J Oper Res 80(1):116–148

Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14(8):1771–1800

Hinton GE (2005) What kind of a graphical model is the brain? In: Proceedings of the 19th international joint conference on artificial intelligence pp 1765–1775

Jahne M, Li X, Branke J (2009) Evolutionary algorithms and multi-objectivization for the travelling salesman problem. In: Proceedings of the 11th annual genetic and evolutionary computation conference, pp 595–602

Jarboui B, Eddaly M, Siarry P (2009) An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Comput Oper Res 36(9):2638–2646

Laporte G (1992) The traveling salesman problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(2):231–247

Larrañaga P, Lozano JA (2001) Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer, Norwell

Laumanns M, Ocenasek J (2002) Bayesian optimization algorithms for multi-objective optimization. Parallel Problem Solving from Nature seventh International Conference pp 298–307
Li M, Yi Z, Zhu M (2009) Solving TSP by using Lotka-Volterra neural networks. Neurocomputing 72(16–18):3873–3880
Lin L (2009) Maximum entropy estimation of distribution algorithm for jssp under uncertain information based on rough programming. International workshop on intelligent systems and application, pp 1–4
Lozano JA, Larraga P, Bengoetxea E (2006) Towards a new evolutionary computation: advances on estimation of distribution algorithms (Studies in Fuzziness and Soft Computing). Springer, New York
Malandraki C, Dial RB (1996) A restricted dynamic programming heuristic algorithm for time dependent traveling salesman problem. Eur J Oper Res 90(1):45–55
Manuel L, Thomas S (2010) The impact of design choices of multiobjective antcolony optimization algorithms on performance: an experimental study on the biobjective TSP. In: Proceedings of the 12th annual conference on genetic and evolutionary computation, pp 71–78
Marti L, Garcia J, Berlanga A et al (2009) On the model-building issue of multi-objective estimation of distribution algorithms. In: Proceedings of the fourth international conference on hybrid artificial intelligence systems, pp 293–300
Miller BL, Goldberg DE (1996) Genetic algorithms, selection schemes, and the varying effects of noise. Evol Comput 4(2):113–131
Muhlenbein H (1998) The equation for response to selection and its use for prediction. Evol Comput 5(3):303–346
Muhlenbein H, Paass G (1996) From recombination of genes to the estimation of distributions I. binary parameters. In: Proceedings of the fourth international conference on parallel problem solving from nature, pp 178–187
Neri F, Mininno E (2010) Memetic compact differential evolution for cartesian robot control. IEEE Comput Intell Mag 5(2):54–65
Okabe T, Jin Y, et al (2004) Voronoi-based estimation of distribution algorithm for multi-objective optimization. In: IEEE congress on evolutionary computation, pp 1594–1601
Ong YS, Lim M, Chen XS (2010) Memetic computation–past, present and future. IEEE Comput Intell Mag 5(2):24–31
Padberg M, Rinaldi G (1988) Branch-and-cut approach to a variant of the traveling salesman problem. J Guid Control Dynam 11(5):436–440
Pelikan M, Goldberg DE, Lobo FG (2002) A survey of optimization by building and using probabilistic models. Comput Optim Appl 21(1):5–20
Pelikan M, Sastry K, Goldberg DE (2005) Multi-objective hBOA, clustering, and scalability. In: Conference on genetic and evolutionary computation, pp 663–670
Pelikan M, Sastry K, Goldberg DE (2006) Multiobjective estimation of distribution algorithm. Scalable optimization via probabilistic modeling 33 Springer pp 223–248
Peng W, Zhang Q, Li H (2009) Comparison between MOEA/D and NSGAII on the multi-objective travelling salesman problem. Stud Comput Intell 171:309–324
Reinelt G (1989) Fast heuristics for large geometric traveling salesman problems. ORSA J Comput 4(2):206–217
Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. In: ACM international conference proceeding series, pp 791–798
Salhi A, Rodriguez JAV, Zhang Q (2007) An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. Genetic and evolutionary computation conference, pp 570–576
Sastry K, Goldberg DE, Pelikan M (2005) Limits of scalability of multiobjective estimation of distribution algorithms. In: IEEE Congress on evolutionary computation, pp 2217–2224
Shi L, Li Z (2009) An improved pareto genetic algorithm for multi-objective TSP. International conference on natural computation, pp 585–588
Shin SY, Lee IH, Zhang BT (2005) Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE T Evol Comput 9(2):143–158
Shim VA, Tan KC, Chia JY (2010) An investigation on sampling technique for multi-objective restricted boltzmann machine. IEEE congress on evolutionary computation, pp 1081–1088
Tan KC, Cheong CY, Goh CK (2007) Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. Eur J Oper Res 177(2):813–839

Tang HJ, Shim VA, Tan KC, Chia JY (2010) Restricted Boltzmann machine based algorithm for multi-objective optimization. IEEE congress on evolutionary computation, pp 3958–3965
Tieleman T (2008) Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on machine learning, pp 1064–1071
Ting CK, Zeng WM, Lin TC (2010) Linkage discovery through data mining. IEEE Comput Intell Mag 5(1):10–13
Velhuizen DA, Lamont GB (1998) Evolutionary computation and convergence to a Pareto front. Late breaking papers at the genetic programming conference, pp 221–228
Villalobos-Arias MA, Pulido GT, Coello Coello CA (2005) A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. IEEE swarm intelligence symposium, pp 23–30
Wellman MA, Gemmill DD (1995) A genetic algorithm approach to optimization of asynchronous automatic assembly systems. Int J Flex Manuf Syst 7(1):27–46
Wu Y, Zhou X (2008) Meliorative Tabu search algorithm for TSP problem. J Comput Eng Appl 44(1):57–59
Wu JB, Xiong SW, Xu N (2007) Simulated annealing algorithm based on controllable temperature for solving TSP. Appl Res Comput 24(5):66–89
Xu C, Xu J, Chang H (2009) Ant colony optimization based on estimation of distribution for the traveling salesman problem. Fifth international conference on natural computation, pp 19–23
Yan XS, Liu HM, Yan J, et al (2007) A fast evolutionary algorithm for traveling salesman problem. Third international conference on natural computation, pp 85–90
Yang M, Kang L, Guan J (2007) An evolutionary algorithm for dynamic multi-objective TSP. In: Proceedings of the second international conference on advances in computation and intelligence, pp 62–71
Yang JQ, Yang JG, Chen GL (2009) Solving large scale TSP using adaptive clustering method. Second international symposium on computational intelligence and design, pp 44–51
Yugay O, Kim I, Kim B, et al (2008) Hybrid genetic algorithm for solving traveling salesman problem with sorted population. Third international conference on convergence and hybrid information technology, pp 1024–1028
Zhang Q, Zhou A, Jin Y (2008) RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE T Evol Comput 12(1):41–63
Zhong X, Li W (2007) A decision-tree-based multi-objective estimation of distribution algorithm. International conference on computational intelligence and security, pp 114–118
Zhou G, Jia L (2008) A novel evolutionary algorithm for bi-objective symmetric traveling salesman problem. J Comput Inf Syst 4(5):2051–2056
Zhu ZX, Jia S, Ji Z (2010) Towards a memetic feature selection paradigm. IEEE Comput Intell Mag 5(2):41–53
Zitzler E (1999) Evolutionary algorithms for multi-objective optimization: methods and applications. Dissertation, Swiss Federal Institute of Technology, Zurich
Zitzler E, Deb K, Thiele L (2000) Comparison of multi-objective algorithms: empirical results. Evol Comput 18(2):173–195

Author Biographies

Vui Ann Shim received the Bachelor of Engineering (Hons) in Electrical and Electronic Engineering from Universiti Teknologi Malaysia in 2008. He is currently working towards the Ph.D. degree at the Centre for Intelligent Control in the National University of Singapore. His current research interests include evolutionary computation and neural networks, specifically in the application of neural network based estimation of distribution algorithm for multi-objective optimization.

Kay Chen Tan is currently an Associate Professor at the Department of Electrical and Computer Engineering, National University of Singapore (NUS). He has published over 200 journal and conference papers and co-authored 5 books. He has been invited to be a keynote/invited speaker for 21 international conferences, and served in the international program committee for over 100 conferences and involved in the organizing committee for over 30 international conferences. Dr. Tan is currently a Distinguished Lecturer of IEEE Computational Intelligence Society and the Editor-in-Chief of IEEE Computational Intelligence Magazine (CIM). He also serves as an Associate Editor/Editorial Board member of 15

international journals, such as IEEE Transactions on Evolutionary Computation, IEEE Transactions on Computational Intelligence and AI in Games, Evolutionary Computation (MIT Press), European Journal of Operational Research, Journal of Scheduling, and International Journal of Systems Science. Dr. Tan is currently a Fellow of the NUS Teaching Academic.

Jun Yong Chia received the Bachelor of Engineering (Hons) in Electrical Engineering from the National University of Singapore in 2010. He is currently working towards a Masters of Engineering degree at the Center for Intelligent Control in the National University of Singapore and a Diplôme d'Ingénieure from Ecole Supérieure d'Electricité. His current research includes evolutionary optimization and data mining.

Jin Kiat Chong received the Bachelor of Engineering (Hons) in Electrical Engineering and the Master of Science (Electrical Engineering) majoring in Automation and Control Engineering from the National University of Singapore in 2002 and 2008 respectively. He is currently working towards the Ph.D. degree at the Centre for Intelligent Control in the National University of Singapore. His current research interests include evolutionary computation and multiobjective optimization, specifically in the areas of differential evolution and evolutionary gradient search techniques.