Expert Systems with Applications 36 (2009) 6995–7001

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Study of genetic algorithm with reinforcement learning to solve the TSP

Fei Liu *, Guangzhou Zeng

School of Computer Science and Technology, Shandong University, Shandong, Jinan 250061, China

ARTICLE INFO

Keywords:
TSP (traveling salesman problem)
Genetic algorithm
Heterogeneous pairing selection
Reinforcement mutation

ABSTRACT

TSP (traveling salesman problem) is one of the typical NP-hard problems in combinatorial optimization problem. An improved genetic algorithm with reinforcement mutation, named RMGA, was proposed to solve the TSP in this paper. The core of RMGA lies in the use of heterogeneous pairing selection instead of random pairing selection in EAX and the construction of reinforcement mutation operator, named RL-M, by modifying the Q-learning algorithm and applying it to those individual generated from modified EAX. The experimental results on small and large size TSP instances in TSPLIB (traveling salesman problem library) have shown that RMGA could almost get optimal tour every time in reasonable time and thus outperformed the known EAX-GA and LKH in the quality of solutions and the running time.

1. Introduction

Given a set of cities and the distances between them, the traveling salesman problem (TSP) is to find a complete, minimal-cost tour visiting each city once. The TSP is a well-known NP-hard problem with many real-world applications, such as jobshop scheduling and VLSI routing (Gutin & Punnen, 2002). The TSP has often served as a touchstone for new problem-solving strategies and algorithms; and many well-known combinatorial algorithms were first developed for the TSP. In this paper, we consider the symmetric TSP, where the distance from a city to another is the same as the distance in the opposite direction.

A large number of methods have been developed for solving TSP. The complexity of exact algorithms is often exponential. In order to tackle larger TSP instances effectively and decrease the computational cost, it is necessary to develop approximate algorithms that do not always aim at finding optimal solutions but at finding quasi-optimal solutions in an acceptable running time. Johnson and McGeoch (1997) provide an excellent survey on approximate algorithms for the TSP. These methods can be roughly divided into local search and global search approaches. In general, the local search approaches, such as 2-opt, 3-opt and Lin-Kernigan (Johnson & McGeoch, 1997) are efficient and fast convergence, because the selection of reconnecting cities depends on geometric neighborhood information and the edges from other individuals in the population. But they may get struck at local minima because they do not deal with the diversity of feasible solutions.

Genetic Algorithm is a global search algorithm appropriate for problems with huge search spaces such as the TSP, in which the crossover realizes the construction of the offspring and the mutation operator maintains the diversity of the individuals. Because the object of the genetic operators is individual structure and these individual structures are relevant to intrinsic characters of the problem, the good operator is problem-specific. If certain features of problem domain are known in advance, problem-specific crossovers should be designed in consideration of these features to obtain good performance, and several problem-specific crossovers have been proposed so far.

To design a problem-specific crossover, it is intuitionally important that offspring inherit appropriate characteristics from parents. Characteristics are substructures of solution candidates and should be selected so as to have the following qualities: (1) Substructures whose existing probability in a solution candidate tend to be higher and higher with it fitness being higher. (2) An influence of the substructure on the fitness tend to be independent of other substructures in a solution candidate.

EAX (edge assembly crossover) shows best performance being compared with other crossovers from the viewpoints of inheritance of edges (Nagata & Kobayashi, 1997, 1999). HeSEA (Tsai, 2004) replaced the heterogeneous pairing selection (HpS) with the random pair selection (RpS), which was the original selection mechanism of the EAX genetic algorithm (Nagata & Kobayashi, 1997). It integrated edge assembly crossover (EAX) and Lin-Kernighan (LK) local search, through family competition and heterogeneous pairing selection. This algorithm is especially useful for evolutionary algorithms in solving large TSP, but the LK algorithm and L times mutation procedure are relatively complex. Also the family competition length L lacks the bases theoretically. We propose a good problem-specific crossover H-EAX, which makes use of the thought of heterogeneous pairing selection to improve the EAX algorithm.

* Corresponding author.
E-mail address: liufeir78@gmail.com (F. Liu).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.08.026

The current state-of-the-art in the TSP suggests that the best performing algorithms combine solution construction or modification with the subsequent application of fast and effective local search algorithms (Johnson & McGeoch, 1997). Reinforcement learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. It is a local optimal algorithm and its immediate reward is based on the feedback of the environment. Aiming at the solution of the TSP, the immediate reward is the reciprocal of the distance between the two neighboring cities. The shorter the distance of the two cities is, the bigger the value of the immediate reward is. Reinforcement learning is used to optimize the tour based on the accumulative reward between two cities. Because the calculation of immediate reward and accumulative reward does not consider the TSP structure except the tour, the RL operator is in fact problem-independent. In order to improve the global search ability of RL, the solution may be based on colony learning, such as Ant-Q and so on (Dorigo & Gambardella, 1997; Gambardella & Dorigo, 1995; Wu & Shi, 2001). Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. The more prefer for paths with a high pheromone level, the higher rate of growth of the amount of pheromone on shorter paths. The solution ability of Ant-Q precedes some local search algorithm, such as 2-opt, 3-opt and so on (Tsai, 2004). But its computational cost is $O(mn^2)$, that is not efficient for larger TSP, in which n is the city numbers and m is the number of ants.

The implementation mechanism of reinforcement learning under the framework of genetic algorithm is described by using gene space division and an algorithm RMGA is then proposed (Ben-nian et al., 2006). But this algorithm did not experiment on the TSP instances. An algorithm RMGA for solving the TSP is then proposed. This method, called reinforcement mutate genetic algorithm, integrates EAX and RL. The idea of improvement on EAX comes from the papers (Tsai, 2004; Xie & Liu, 2006). In RMGA, the adjustments of the individual obtained from EAX using Q-learning is named reinforcement mutation. Different from the traditional Q-learning, reinforcement mutation considers searching the next unvisited city s in the TSP space. But only when the distance $d(t,s)$ between s and the current city is less than the distance $d(t,q)$ between t and its neighboring city q in the optimized tour, the operation that replacing q with s is then performed. Otherwise the operation maintains the primary relations between cities.

The objective of our research is to develop an effective method capable of finding high-quality solutions by using reinforcement learning. Our method uses the improved edge assembly crossover (EAX) and reinforcement mutation. Our method is validated by comparing it with EAX-GA (Nagata & Kobayashi, 1997) and the LK-Helsgaun method (LKH) (Helsgaun, 2000), which are effective heuristics for the TSP. Experimental results of benchmarks having up to 3000 cities show good strategy and performance.

The outline of the paper is as follows. Section 2 describes the algorithm RMGA to solve the TSP. Section 3 discusses the effects and the role of implementing the algorithm in TSP instances and the analysis of data compared with EAX-GA. A conclusion and some directions for future research are given in the last section.

2. RMGA algorithm to solve the TSP

In the traveling salesman problem, or "TSP", we are given a set $\{c_1, c_2, \ldots, c_n\}$ of cities and for each pair $\{c_i, c_j\}$ of distinct cities a distance $d(c_i, c_j)$. Our goal is to find an ordering π of the cities that minimizes the quantity $\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)})$. This quantity is referred to as the tour length, since it is the length of the tour a salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city. We shall concentrate in this chapter on the symmetric TSP, in which the distances satisfy $d(c_i,c_j) = d(c_j,c_i)$, $1 \leqslant i, j \leqslant n$. In genetic algorithm, the individual is the tour in which all of the cities are visited exactly once and the fitness function is the reciprocal of the tour length $1/[\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)})]$. The shorter the tour length, the bigger the value of fitness function, the better the individual.

2.1. The improvement of EAX algorithm

Since the main search mechanism of edge-based crossovers is to generate offspring by preserving and adding edges, the characteristics of edge-based operators are analyzed by measuring their ability to add and preserve the "good edges" of EAX, DPX, and ERX operators. A "good edge" is defined as an edge in a known optimal tour. In summary, a good crossover operator for solving TSP problems should be edge-based and possess good mechanisms for adding and preserving "good edges". EAX is regarded as a good crossover operator because it meets these requirements.

EAX (Nagata & Kobayashi, 1997) has two important features – preserving parents' good edges in a novel technique and adding new edges in a greedy method. It considers the TSP search space as a weighted complete graph $G = \{V,E\}$, in which V is the set of vertices and E is a set of weighted edges. Suppose A (denote acceptor) and B (denote donor) are parent individual, E_A and E_B are their edge sets respectively. So $G_{AB} = (V, E_A \cup E_B)$, and AB cycle is a closed tour on G_{AB}. The basic process of EAX-GA is as follows:

I. Select two parent individuals A and B at random, generate the sub-graph G_{AB}.
II. Iterate the Steps (1)–(5) N_{cross} times:
  (1) Alternately select edges from A and B in G_{AB} to generate ABcycle. Let C to be ABcycle generated.
  (2) Select ABcycle at random. Let D to be a union of selected ABcycles.
  (3) Generate intermediate solution by $E_A \to (E_A \cup \overline{D}) \cup (E_B \cap D)$.
  (4) Modify intermediate solution to valid tour in greedy strategy.
  (5) Estimate and memorize the optimum: tour.
III.Select the two best individuals from A, B and tour to be the next generation.

Tsai (2004) has proposed a parent selection method, called heterogeneous selection evolutionary algorithm (HeSEA). HeSEA uses the concept of family competition and heterogeneous pairing selection (HpS). In HpS, the parent pairs are selected for individuals sharing edges less than a criterion, which is the average number of common edges in a population. HpS selection is called a family competition strategy. Each individual is the "family father". This "family father" and another solution denote as "family mother", selected by HpS act as parents to perform the EAX crossover operation. HpS can avoid crossover of highly similar parents, and retain the diversity of the population. According to the experiments on TSPLIB instances, HpS is more powerful than RpS in both maintaining the diversity of the population and generating better offspring. The computational complexity of edge similarity between any tour and others in population is $O(mn)$, m is the population size and n is the city numbers in TSP search space.

Xie and Liu (2006) have presented an extended multiagent optimization system, called $MAOS_E$ and an optimization algorithm using the thought of heterogeneous pairing selection. The clue of this optimization algorithm is that each agent performs EAX crossover between the best individual in its private memory and an individual randomly selected from socially shared memory, which serves as the blackboard for all agents by creating a shared past best individual.

We make use of the idea of heterogeneous pairing selection (Tsai, 2004; Xie & Liu, 2006) to improve the EAX algorithm. Firstly, the individual of the current population in order becomes the "family father", and another individual picked up from population except the "family father" is the "family mother". Secondly, when the offspring gained from this crossover is better than "family father", this offspring replace the "family father" in the population, as presented in Fig. 1. This substitute strategy makes the current population to be a mixed one that is composed of "crossover offspring" and the original individual. And then the next selected "family mother" may be the better individual when pairing with the next "family father". So, the advantage of mixed population ensures that the crossover operation often is made in an excellent population. It not only can avoid crossover of highly similar parents, but also retain the diversity of the population.

According to the above strategy, the improved EAX algorithm H-EAX is depicted as Algorithm 1, in which the input is the current population A, the output is the result of crossover A and m is the population size.

Algorithm 1. H-EAX(A,m)

Begin
For ($i = 1; i < $ m; i++) do
{select $A_i \in A$ as father individual, pick up A_j from $A - A_i$ as mother individual at random;
$X \leftarrow$ iterate N_{cross} times EAX(A_i, A_j) and generate the best individual;
If X is better than A_i, then $A_i \leftarrow X$; /* X substitutes A_i and retain A_i in A */
}
Return A;
End.

2.2. Reinforcement mutation algorithm

When the TSP tour is optimized by the reinforcement learning algorithm, the state denotes the city number, the action denotes the selection of the next city and the immediate reward $r(t,q)$ between the current city t and the next city q is the reciprocal of the distance between t and q, that is $r(t,q) = 1/d(t,q)$. The cumulated reward RQ(t,q) is updated according to the following function, $\mathrm{RQ}(t,q) = (1-\alpha)\mathrm{RQ}(t,q) + \alpha[r(t,q) + \gamma \max_{z \in J(t)} \mathrm{RQ}(q,z)]$, where α is the learning rate, γ is the discount rate ($0 < \alpha,\gamma < 1$) and $J(t)$ denotes the unvisited cities when the current city is t.

The main idea of the reinforcement mutation algorithm is represented in Fig. 2, where X is the mutated tour and Y is the mutation result tour. In X whether the connection relation between city t and w is mutated rests with whether the distance $d(t,w)$ is bigger than the distance $d(t,s)$, s is the next city selected according to reinforcement learning. After several times mutation, if the length of Y is less than that of X, then X is replaced with Y in original population, otherwise X is retained.

Suppose n is the city numbers of a TSP instance; edge is the weighted edge set, denoted as the distance matrix DIST; the mutated individual X is denoted as the matrix $X[0, n-1]$, $X[0]$ is the start city; Len X is the length of the tour X; the offspring of X is denoted as the matrix $Y[0, n-1]$, Len Y is the length of the tour Y.

Algorithm 2. RL-M(X)

Begin
Compute the Len X according to DIST;
$\forall(p,q) \in$ Edge, RQ(p,q) $\leftarrow 1/\mathrm{d}(p,q)$; /* *initialize the RQ value of Edge* */
Count $\leftarrow$ 0;
Repeat/* *repeat mutation operation* */
{$t \leftarrow Y[0] \leftarrow X[0]$; /* *initialize the mutation tour Y*, $Y[0]$ is the start city, t is the current city in Y */
$J(t) \leftarrow \{1,\ldots,n\} - t$; /* $J(t)$ is the unvisited city sets when the current city is t */
Len $Y \leftarrow 0$;
For ($k = 1$; $k < n$; k++)/* *once mutation process* */
{If ($k \neq n-1$) then
{$q \leftarrow$ select city $s \in J(t)$ according to the maximum of $\frac{e^{RQ(s,z)/T}}{\sum_{z \in J(t)} e^{RQ(s,z)/T}}$;
Let the reflected position of the current t in Y to X is $X[i]$ (as showed in Fig. 2)
If $X[i+1] \in J(t)$ and $d(X[i], X[i+1]) < d(t,s)$, $q \leftarrow X[i+1]$;
$Y[k] \leftarrow q$; Len $Y \leftarrow$ Len $Y + d(t,q)$;
If ($k < n-2$) then $J(q) \leftarrow J(t) - q$;
If ($k = n-2$) then $J(q) \leftarrow J(t) - q + Y[0]$;
}
Else
{$q \leftarrow Y[0]$; Len $Y \leftarrow$ Len $Y + d(t,q)$;}
EndIf
$\mathrm{RQ}(t,q) \leftarrow (1-\alpha)\mathrm{RQ}(t,q) + \alpha[1/d(t,q) + \gamma \max_{z \in J(q)} \mathrm{RQ}(q,z)]$; /* update RQ table */
$t \leftarrow q$;
} /* *complete once mutation process* */
Count++;
} **Until** (**Len Y < Len X or Count = N**); /* *N denotes the mutation times* */
If Len $Y <$ Len X then $X \leftarrow Y$, otherwise retain X; return(X, Len X);
End.

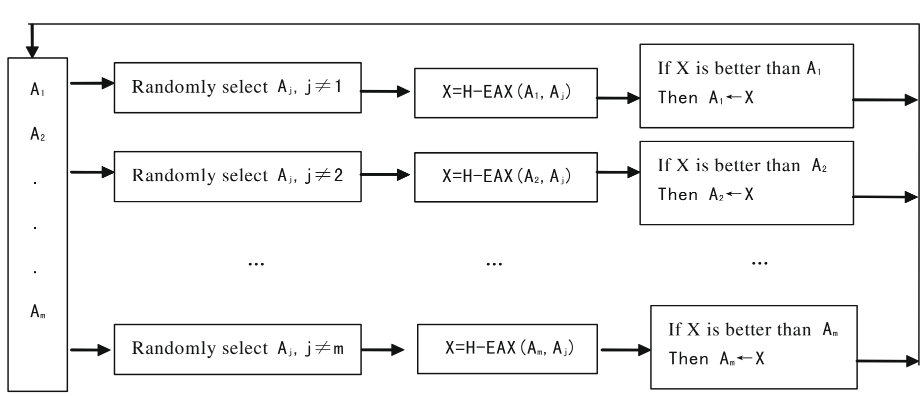

Fig. 1. The idea of EAX algorithm using mixed population.

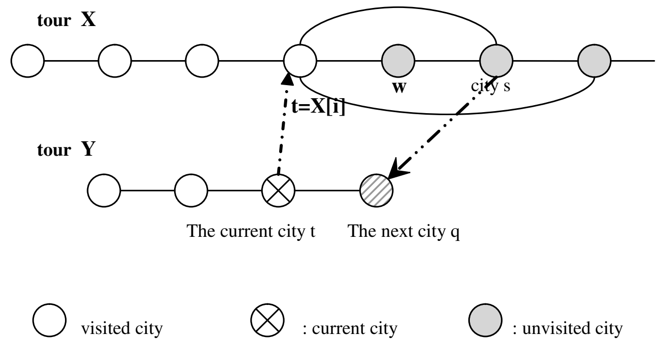

Fig. 2. The idea of reinforcement mutation algorithm.

2.3. RMGA algorithm

2.3.1. Genetic operators

(1) *Crossover*: It is commonly recognized in the GA community that crossover is the most important operator in GA. For this reason, many crossovers have been proposed for the TSP, including partially mapped crossover (PMX) (Goldberg & Lingle, 1985), cycle crossover (Oliver, Smith, & Holland, 1987), MPX (Gorges-Schleuter, 1997), edge recombination (ERX) (Dzubera & Whitley, 1994), EAX (Nagata & Kobayashi, 1997), distance preserving crossover (DPX) (Freisleben & Merz, 1996), generic crossover (GX) (Freisleben & Merz, 2001), NGA (Jung & Moon, 2001), and GSX (Nguyen, Yoshihara, Yamamori, & Yasunaga, 2002). These crossovers can be divided into two classes. Those in the first class use local problem-dependent information (e.g., edge lengths) in generating offspring, whereas crossovers in the second class do not (although the latter can use some global information, e.g., the fitness of a TSP chromosome relies on edges between loci and not the loci themselves). Variants of EAX, DPX, and GX belong to the first class, and variants of MPX, ERX, and GSX belong to the second class.Our strategy proposes an H-EAX operator, based on the EAX crossover. This operator has the advantage of quickly exploiting the diversity of the GA population that causes the method to converge very quickly. It works very well for problems with small-scale and medium-scale TSPs having up to several thousands of cities. However, for larger-scale TSPs it often causes the GA to become trapped in the local minima.

(2) *Mutation*: We propose a new mutation operator based on reinforcement learning. This mutation operator is followed by a call to the tour improvement heuristic that is applied to each individual being mutated. However, a number "num_iteration" of iterations are applied. For each iteration, the mutation operator performs on the individual a special kind of non-sequential move. The tour improvement heuristic is then applied. If the new tour produced by the tour improvement heuristic is not better than before mutation, it is discarded. Otherwise, the old tour is replaced by the new one.

2.3.2. General flow of RMGA

Based on the solution qualities and the abilities to add and preserve "good edges", H-EAX and RL-M are good methods for adding "good edges" in the early stage and preserving "good edges" in the late stage. The proposed approach integrates H-EAX and RL-M with family competition and heterogeneous pairing selection (HpS). These two mechanisms are added to maintain the diversity of the population and increase the probability of generating better children.

The main idea of the RMGA (reinforcement mutation genetic algorithm) is presented as Algorithm 3. Firstly, the entire individual in the current population A are in order operated by H-EAX crossover, and then the crossover result set C is gained. Secondly, each individual in the set C is in order operated using RL-M, and then the next population A_{i+1} is gained. The above process is repeated until the stopping criterion is satisfied.

Let $A_{\mathrm{curr}} = \{A_1, A_2, \ldots, A_{\mathrm{m}}\}$ denotes the current population, $C = \{C_1, C_2, \ldots, C_{\mathrm{m}}\}$ denotes the crossover result of A_{curr}, A_{next} denotes the next population after mutation, and m is the population size.

Algorithm 3. RMGA

Begin
$A_{\mathrm{curr}} \leftarrow$ generate random population of m individuals $\{A_1, A_2, \ldots, A_{\mathrm{m}}\}$ (suitable solutions for the problem);
Repeat
{$A_{\mathrm{next}} \leftarrow \Phi$;
$C \leftarrow$ H-EAX(A_{curr},m); /* *crossover* */
For ($i = 1$; $i < m$; i++)
{$A_i \leftarrow$ RL-M(C_i); /* *mutation* */
$A_{\mathrm{next}} \leftarrow A_{\mathrm{next}} \cup A_i$;
}
$A_{\mathrm{curr}} \leftarrow A_{\mathrm{next}}$;
}Until the stopping criterion is satisfied;
Return the optimal tour in A_{curr} and its length;
End.

There are several possible solutions to decide when to stop execution of Algorithm 3, such as the maximum number of generations, the maximum running time, no improvement for a certain number of generations and the difference between the best individual and the worst one etc.

3. Experimental results

The performance of algorithm to solve the TSP may be compared from two aspects: the running time and result quality. Because the running time is related with the environment condition, such as CPU performance, OS and so on, and the same running environment is very difficult, we only compare the algorithms through the result quality. The compared algorithm are EAX-GA (Nagata & Kobayashi, 1997) and LKH (Helsgaun, 2000). The TSP instances is derived from TSPLIB (Reinelt, 1991).

As stated before, the population size(m) critically determine the computation time and quality of the solution. Various values of the factor were tested on problems lin318, u724 and pr2392, to evaluate its effects in RMGA. Typically, RMGA yields similar curves for all test problems. Fig. 3 represents the relationship between the population size and the average error, defined by (1), obtained in solving problem lin318. For a particular number of evaluations of the function, the average declines as the population size increases. The improvement is insignificant when population size exceeds 100. Accordingly, the population size is set to 100. Fig. 4 shows the effect of the population size on the quality of the solution to problem u724. The population size is set to 300 because improvements in the quality of solution are limited when m exceeds 300.

The average error(%) and the best times required to solve the problem are used as measures of the performance of the compared methods. Table 1 shows the experimental results about the application of the two methods on small size TSP instances. In Table 1, the parameters and the experiment data of EAX-GA are all gained from the paper (Chan, Lee, & Kao, 2005). The parameters of RMGA

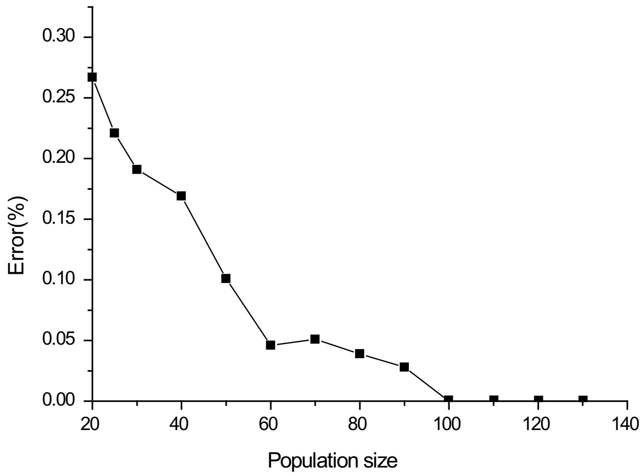

Fig. 3. The relation between population size and error of Lin318.

are setted as follows. The crossover time of iterations N_{cross} is 20; reinforcement mutation parameters: $\alpha = 0.1$, $\gamma = 0.9$, $T = 1$. The stopping criterion is that the optimal result is obtained or the number of the same length of individual is 50. In this study, the solution quality is given as an error percentage (%), defined as

$$\text{Error} = [(\text{average-optimum})/\text{optimum}] * 100\% \quad (1)$$

where average is the average value of the best solutions obtained by the method over 50 trials and optimum is the minimum length of tour so far. The best times is the number of the optimum obtained over 50 trials. From Table 1, RMGA outperforms EAX-GA in the error percentage and the best times.

RMGA is superior to EAX-GA in the result quality in two ways:

(1) EAX-GA does not have a separate mutation operation. RL-M and EAX may complement each other in performance by preserving "good edges" and inserting new edges into offspring. RL-M may generate mutation on the better edges in TSP search space. So, adding the complementary operator RL-M into the EAX-GA, not only optimizes the new individual that obtained from EAX but also maintains the population diversity.
(2) H-EAX generates the heterogeneous pairing selection in a mixed population. The paper (Tsai, 2004) proved that the heterogeneous pairing selection outperforms the random pair selection. The application of the mixed population may ensure that each crossover pairing selection is made in the best population.

Table 2 compared our RMGA with LKH using all 16 instances having from 1002 to 2392 cities in TSPLIB. For each instance, 10 runs were performed. The parameters in RMGA are setted as follows. The crossover number of iterations N_{cross} is 20; reinforcement mutation parameters: $\alpha = 0.1$, $\gamma = 0.9$. The stopping criterion is that the maximum number of generations N is 500 and the number of individual that is the same length is 50. RMGA and LKH were performed on an individual PC (1.82 GHz Pentium III, 512 MB memory). The code of RMGA was written in JAVA. We used the file LKH.WIN, which is the executable of LKH code (version 1.3) for the Windows operating system. All parameters were set as default.

In Table 2, the column "Name" indicates the instance name in TSPLIB; the columns "Best", "Average", and "Worst" show the best, average and worst tour lengths of 10 runs, respectively (the values in parentheses are the percentage above the optimal); the column "Time (s)" indicates the average running time in seconds; the column "Population size" is the initialized population number of RMGA.

Helsgaun Gorges-Schleuter (1997) reported that, except for two instances rl5915 and rl5934, LKH was able to find optimal tours in at least one out of 10 runs for all instances having less than 2392 cities. In our experiments, however, LKH failed to find optimal tours for two other instances in this size range, namely: u1817 and d2103.

In terms of running time, RMGA was slower than LKH on TSPs having up to 2392 cities. The reason is partly because the time RMGA spent for reinforcement mutation accounted for a large portion of the total running time. But the result quality of all instances of RMGA was better than that of LKH. A large portion of the results of RMGA can converge to the optimal.

Our method was validated by comparing with LKH, which is one of the most effective heuristics for the TSP. Experimental results of benchmarks having up to 2392 cities show that our method works more effectively on result quality, but the running time was slower.

The main contribution of our correspondence is to show that, when properly implemented, the combination of a GA with local search is very promising for the TSP, and that the effectiveness and efficiency of the local search play important roles in the performance of hybrid GAs. The design of the GA and the balance between local and global search also contribute to the improvement of hybrid GAs.

The incorporation of RL-M into the RMGA algorithm can thus enhance the performance and solution quality. One interesting question regards whether the RMGA algorithm (with RL-M mutation operator incorporated) can be further improved? We do not have definite answer to this question yet. However, from the analysis of EAX algorithm we can point out some possible steps in EAX that may be targets for improvements. For example, the formation of AB_cycles is random in current implementation of EAX; additional heuristics for better AB_cycle formation may provide fruitful results. Another possible step is the select of AB_cycles for construction of E-set. A simple and effective selection scheme is critical to EAX algorithm. Finally, the modification of intermediates into valid tours can also be refined. The selection of cities for joining two sub-tours, and selection of sub-tours to be joined can be considered for possible further enhancements.

Table 1
The comparison of RMGA and EAX-GA in small size TSP instances

TSP instance	RMGA		EAX-GA		Population size
	Error (%)	Best times/total times	Error (%)	Best times/total times	
eil101	0	50/50	0.006	48/50	100
lin318	0	50/50	0.233	12/50	100
pcb442	0	50/50	0.020	26/50	100
rat575	0.0018	44/50	0.031	3/50	300
u724	0	50/50	0.003	44/50	300
rat783	0	50/50	0.008	31/50	300

Table 2
Comparative results of RMGA and LKH for TSPLIB instances with $n > 1000$

Name (Optimal)	Method	Best (%)	Average (%)	Worst (%)	N_{opt}	Time (s)	Population size (of RMGA)
pr1002	RMGA	Optimal	Optimal	Optimal	10/10	1442.375	100
(259045)	LKH	Optimal	Optimal	Optimal	10/10	4.3	
u1060	RMGA	Optimal	Optimal	Optimal	10/10	1695.813	100
(224094)	LKH	Optimal	Optimal	Optimal	10/10	55	
vm1084	RMGA	Optimal	Optimal	Optimal	10/10	1305.282	100
(239297)	LKH	Optimal	239318.4(0.0089)	239407(0.0460)	7/10	24	
pcb1173	RMGA	Optimal	Optimal	Optimal	10/10	1668.016	100
(56892)	LKH	Optimal	56893(0.0018)	56897(0.0087)	8/10	15.6	
d1291	RMGA	Optimal	50823(0.0433)	50845(0.0866)	3/10	1084.62	100
(50801)	LKH	Optimal	50826.5(0.0502)	50886(0.167)	7/10	51.5	
rl1304	RMGA	Optimal	Optimal	Optimal	10/10	1310.45	100
(252948)	LKH	Optimal	253190.4(0.0958)	253435(0.1925)	4/10	13.7	
rl1323	RMGA	Optimal	Optimal	Optimal	10/10	1273.82	100
(270199)	LKH	Optimal	270241.6(0.0158)	270330(0.0485)	1/10	17.7	
nrw1379	RMGA	Optimal	56639.6(0.0028)	56643(0.0088)	6/10	2574.6	300
(56638)	LKH	Optimal	56638.5(0.00088)	56643(0.0088)	9/10	28.6	
u1432	RMGA	Optimal	152996.1(0.0171)	153035(0.0425)	5/10	2157.34	300
(152970)	LKH	Optimal	Optimal	Optimal	10/10	64.2	
d1655	RMGA	Optimal	Optimal	Optimal	10/10	3435.23	300
(62128)	LKH	Optimal	62128.1(0.0002)	62129(0.0016)	9/10	46.3	
vm1748	RMGA	Optimal	Optimal	Optimal	10/10	1994.38	300
(336556)	LKH	Optimal	336631.5(0.0224)	336701(0.0431)	3/10	44.9	
u1817	RMGA	Optimal	57206(0.0087)	57222(0.0367)	4/10	2144.69	300
(57201)	LKH	Optimal	57252.3(0.0897)	57272(0.124)	1/10	85.7	
rl1889	RMGA	Optimal	316539.5(0.0011)	316550(0.0044)	7/10	4584.98	300
(316536)	LKH	Optimal	316539.8(0.0012)	316549(0.0041)	5/10	57	
d2103	RMGA	Optimal	80451.3(0.0016)	80455(0.0062)	7/10	2919.703	300
(80450)	LKH	80455(0.006)	80474.7(0.031)	80490(0.0497)	0/10	404.2	
u2152	RMGA	Optimal	Optimal	Optimal	10/10	3198.7	300
(64253)	LKH	Optimal	64270.7(0.028)	64299(0.072)	4/10	147.6	
pr2392	RMGA	Optimal	Optimal	Optimal	10/10	4243.7	300
(378032)	LKH	Optimal	Optimal	Optimal	10/10	125.3	

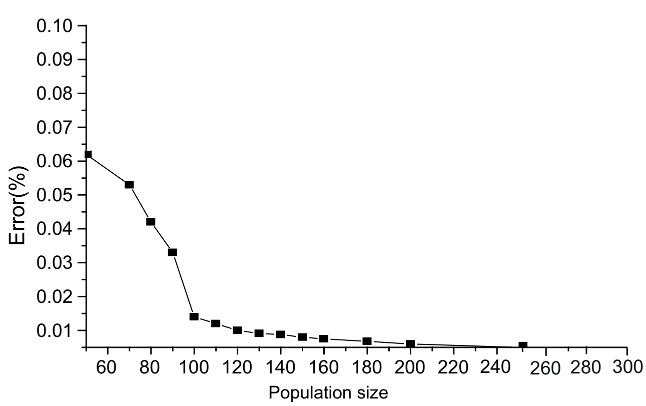

Fig. 4. The relation between population size and error of u724.

4. Conclusions and future work

In this paper a new hybrid GA algorithm RMGA with an improved H-EAX crossover operator and RL-M mutation operator has been presented for solving the TSP. This algorithm uses the genetic algorithm as the framework and uses reinforcement learning as mutation operator, so essentially it is an evolutionary algorithm. The searching space of H-EAX is big and its convergence rate is slow. RL-M performs local optimization over the individuals of H-EAX, so the convergence rate can be enhanced and the population diversity may be maintained. RMGA can seamlessly integrate H-EAX and RL-M to improve the overall search. Heterogeneous edge selection and family competition maintain the diversity of the population and help to generate good children with a high probability. Experiment on TSP instances shows that RMGA outperforms EAX-GA in result quality

The next steps for continuing this research are: (i) Test the largest TSP instances in order to validate the adaptability of RMGA. Because the largest TSP instances test consumes more running time, the experiment result when the problem size in over 3000 is not proposed. (ii) Study the approaches that may improve the running rate of RL-M. The searching range of the next city to be visited in Algorithm 2 is all of the unvisited cities in TSP search space. The working space of EAX is only a subspace that constructed by parent individuals in TSP space. So if the searching range is constrained into the working space of EAX, whether it can not only enhance the running rate but also ensure the result quality is a researchful problem. (iii) Study the better RL-M operator that may complement with H-EAX. EAX is a kind of crossover operator based on neighbored edges information. Only when RL-M maintains the more "good edges" in EAX tours, the complement of RL-M and H-EAX may be ensured and the optimization effect of EAX may be strengthened.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 60573169.

References

Ben-nian, Wang et al. (2006). RMGA: A reinforcement learning based genetic algorithm. *Acta Electronica Sinica, 34*(5), 856–860. 866.
Chan, C. H., Lee, S. A., & Kao, C. Y., et al. (2005). Improving EAX with restricted 2-opt. In *Genetic and evolutionary computation conference* (pp. 1471–1476). Washington DC, USA.
Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. *BioSystems, 43*(2), 73–81.
Dzubera, J., & Whitley, D. (1994). Advanced correlation analysis of operators for the traveling salesman problem. In Y. Davidor, H. P. Schwefel, & R. Manner (Eds.), *Parallel problem solving from nature-PPSN III* (pp. 68–77). New York: Springer-Verlag.

Freisleben, B., & Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 616–621).

Freisleben, B., & Merz, P. (2001). Memetic algorithms for the traveling salesman problem. *Complex Systems, 13*(4), 297–3345.

Gambardella, L. M., & Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of the ML-95, 12th international conference on machine learning* (pp. 252–260). Morgan Kaufmann.

Goldberg, D. E., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the international conference on genetic algorithms and their applications* (pp. 154–159).

Gorges-Schleuter, M. (1997). Asparagos96 and the traveling salesman problem. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 171–174).

Gutin, G., & Punnen, A. P. (Eds.). (2002). *The traveling salesman problem and its variations*. Kluwer.

Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research, 126*(1), 106–130.

Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts & J. K. Lenstra (Eds.), *Local search in combinatorial optimization* (pp. 215–310). Chichester, England: John Wiley and Sons.

Jung, S., & Moon, B. (2001). The natural crossover for the 2D Euclidean TSP. In *Proceedings of the Genetic and evolutionary computation conference* (pp. 1003–1010).

Nagata, Y., & Kobayashi, S. (1997). Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In *Proceedings of the 7th international conference on genetic Algorithms (ICGA)* (pp. 450–457).

Nagata, Y., & Kobayashi, S. (1999). An analysis of edge assembly crossover for the traveling salesman problem. *IEEE Transactions on Systems, Man and Cybernetics*, 628–633.

Nguyen, H. D., Yoshihara, I., Yamamori, K., & Yasunaga, M. (2002). Greedy genetic algorithms for symmetric and asymmetric TSPs. *IPSY transactions on mathematical modeling and applications* (Vol. 43, no. SIG10(TOM7), pp. 165–175).

Oliver, I., Smith, D., & Holland, J. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the second international conference on genetic algorithms* (pp. 224–230).

Reinelt, G. (1991). TSPLIB:A traveling salesman problem library. *ORSA Journal on Computing, 3*(3), 376–384.

Tsai, Huai-Kuang et al. (2004). An evolutionary algorithm for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34*(4), 1718–1729.

Wu, B., & Shi, Z. Z. (2001). An ant colony algorithm based partition algorithm for TSP. *Chinese Journal of Computers, 24*(12), 328–1333.

Xie, Xiao-Feng., & Liu, Jiming (2006). How autonomy oriented computing (AOC) tackles a computationally hard optimization problem. In *AAMAS'06, May 8–12, Hakodate, Hokkaido, Japan* (pp. 646–653).