International Journal of Automation and Computing 06(1), February 2009, 97-102
DOI: 10.1007/s11633-009-0097-4

A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Pickup and Delivery

Fang-Geng Zhao[1,2,*] Jiang-Sheng Sun[3] Su-Jian Li[2] Wei-Min Liu[2]
[1]Vehicle Management Institute, Bengbu 233011, PRC
[2]Department of Logistics Engineering, University of Science and Technology Beijing, Beijing 100083, PRC
[3]Ordnance Technology Research Institute, Shijiazhuang 050003, PRC

Abstract: In this paper, a hybrid genetic algorithm (GA) is proposed for the traveling salesman problem (TSP) with pickup and delivery (TSPPD). In our algorithm, a novel pheromone-based crossover operator is advanced that utilizes both local and global information to construct offspring. In addition, a local search procedure is integrated into the GA to accelerate convergence. The proposed GA has been tested on benchmark instances, and the computational results show that it gives better convergence than existing heuristics.

Keywords: Genetic algorithm (GA), pheromone-based crossover, local search, pickup and delivery, traveling salesman problem (TSP).

1 Introduction

The traveling salesman problem with pickup and delivery (TSPPD) is one extension of the classical traveling salesman problem (TSP) where each customer to be served is associated with two quantities of goods to be collected and delivered, respectively. The TSPPD can be described as follows: Let $G = (V_0, A)$ be a complete and undirected graph, where $V_0 = 0 \cup V$ is the customer set (with $V = 1, 2, \cdots, n$). Vertex 0 represents the depot, whereas the other vertices represent the customers. Each customer i requires both a non-zero pick-up demand $p(i)$ that should be sent to the depot and a non-zero delivery demand $d(i)$ that originates from the depot. For each edge $(i, j) \in A, (i, j \in V_0)$, a non-negative edge cost (or distance), d_{ij}, is also given. A vehicle with capacity Q is located at the depot. The TSPPD involves determining a tour starting and ending at the depot, and visiting each customer exactly once. In each point along its tour, the vehicle cannot carry a total load greater than Q. The objective is to minimize the overall length of the tour.

The TSPPD is an important extension to the TSP, since there are many practical applications, such as the distribution system of the soft drink industry where full bottles must be delivered and empty bottles have to be collected.

Mosheiov[1] proposed a mathematical model for the TSPPD and the first heuristic algorithms based on the extension of heuristics for the TSP. Anily and Mosheiov[2] proposed a new heuristic for TSPPD based on the solution of shortest spanning trees. Gendreau et al.[3] proposed two heuristic algorithms for the TSPPD. The first was based on a linear time algorithm for the optimal solution of a special case of TSPPD arising when the graph G is a cycle, whereas the second is based on the tabu search approach using a two-exchange neighborhood. Baldacci et al.[4] gave a new mathematical model for the TSPPD based on a two-commodity network flow approach, and proposed a branch-and-cut algorithm for the optimal solution of the TSPPD. Recently, Hernández-Pérez and Salazar-González[5] put forward two heuristic algorithms for the one-commodity pickup-and-delivery traveling salesman problem (1-PDTSP), and the algorithms are also used to solve the TSPPD. Their first method described a greedy algorithm with a k-optimality criterion, while the second heuristic algorithm was an incomplete optimization algorithm based on the branch-and-cut procedure.

A correlative problem to the TSPPD is the TSP with backhauls (TSPB, see [6] for details), where each customer requires a pure pickup service or a pure delivery service, and all delivery customers must be served before the pickup customers. In addition, when the TSPPD is generalized to a multiple vehicle version, it becomes the vehicle routing problem (VRP) with pickup and delivery (VRPPD).

The TSPPD is NP-hard because it coincides with TSP when the vehicle capacity is large enough. Therefore, the development of heuristic algorithms for this problem is of primary interest. In recent years, more and more meta-heuristics are successfully used to solve the VRPPD, such as tabu search[7] and ACO algorithm[8]. However, no application of genetic algorithm (GA) for both the VRPPD and TSPPD can be found to our knowledge. To remedy this gap, we utilized GA, which may be the most effective algorithm for TSP[9], to solve the TSPPD.

GA is a search algorithm based on an evolutionary principle. Since it was first introduced by Holland[10], GA has been successfully applied to several NP-hard combinatorial optimization problems, such as the TSP[9], the quadratic assignment problem (QAP)[11], the VRP[12], and the job-shop scheduling problem (JSP)[13], because of its simplicity, ease of operation, and global perspective.

In this paper, we applied GA to solve the TSPPD, and this is the first application of GA to solve this problem to our best knowledge. In our hybrid GA, a novel pheromone-based crossover operator, which utilizes both local and global information, is applied to construct the offspring. The local information used here includes edge lengths and

Manuscript received March 11, 2008; revised April 6, 2008
*Corresponding author. E-mail address: fgzhao@163.com

adjacency relations, whereas global information is stored as pheromone trails. The utilization of global information in crossover can increase the chance to find a better solution, and then improve the performance of GA. In addition, a local search procedure is embedded into GA to accelerate convergence. The proposed algorithm was tested on benchmark instances used in previous studies, and experimental results show that our algorithm outperformed existing heuristic algorithms in accuracy.

This paper is organized as follows. Section 2 describes the proposed GA, and computational results are presented in Section 3. Finally, Section 4 draws conclusions from this study.

2 The proposed GA for TSPPD

In our proposed GA, path representation is used. Algorithm 1 outlines the proposed GA for the TSPPD. After initializing the parameters used in the algorithm, the GA starts from a random population of individuals (feasible tours), followed by 2-opt local optimization, and iteration until some pre-defined stopping criterion is satisfied. Each individual is evaluated in each iteration (generation) based on its fitness function. In each iteration, individuals are probabilistically selected from the population according to the rule of roulette wheel selection. Then, the selected individuals are recombined and mutated to generate offspring. To accelerate the convergence of GA, a local search procedure, 2-opt, is used after recombination (crossover). After the operators mentioned above, half of the individuals with worse fitness in population are replaced by offspring, and the pheromone trails are updated.

Algorithm 1. The proposed GA for TSPPD.

```
begin
  initialize parameters and pheromone trails:
  generate initial population $P$;
  optimize initial solutions by 2-opt;
  evaluate individuals in $P$;
  while terminal condition is not satisfied, do
      for $i = 1$ to $pop\_size/2$ do
          select two parents $p_1$ and $p_2$ from $P$ according to
          roulette wheel selection rule;
          cross $p_1$ and $p_2$ with a predefined probability and
          attain an offspring $o$;
          $o \leftarrow$ 2-opt ($o$);
          mutate $o$ with a predefined probability;
      end for;
      replace half of individuals in $P$ with offspring;
      update pheromone trails;
  end while;
end.
```

2.1 The generation of initial population and fitness evaluation

In our algorithm, an adapted nearest neighbor heuristic[14] is used to generate initial population. Let r denote a TSPPD route, $r(i)$ represent the i-th customer visited by vehicle, and l_i denote the load of vehicle when the i-th customer is visited (just before it is served). Obviously, the load of the vehicle when it leaves the depot is $l_1 = \sum_{j=1}^{n} d(j)$. The adapted heuristic is described as follows:

Step 1. Let $k = 1, r(0) = 0$ (the depot), and select random customer t;

Step 2. If $l_k + (p(t) - d(t)) \leqslant Q$, then let $r(k) = t$, $k = k + 1$, $l_k = l_{k-1} + p(t) - d(t)$; otherwise, back to Step 1;

Step 3. Select the nearest one as the k-th customer, $r(k)$, of r from customers satisfying the following equation:

$$l_k + (p(x) - d(x)) \leqslant Q, \quad x \in S$$

where S is the set of unvisited customers;

Step 4. If $k = n$, STOP. Otherwise, let $k = k + 1$, $l_k = l_{k-1} + p(r(k-1)) - d(r(k-1))$, and back to Step 3.

This procedure is repeated until pop_size (the size of population) routes are generated. The fitness of a chromosome (route) is derived from its solution quality. In the proposed GA, we use the rank of each chromosome in the population to evaluate its fitness[15]. To evaluate the fitness of chromosomes, the chromosomes should be first ordered according to their total lengths, and the chromosome with the minimum length gets rank 1. With these ranks, the fitness of a chromosome of rank i is calculated as follows:

$$fitness_i = Max - \left[(Max - Min) \cdot \frac{(i-1)}{(k-1)}\right], \quad (i = 1, 2, \cdots, k)$$

where k is the number of chromosomes in the population, Max is the maximum value of fitness, and Min is the minimum value of fitness. It is easy to see that the best chromosome in population gets fitness value Max and the worst chromosome gets fitness value Min. In our implementation, Max and Min are set at 1.2 and 0.8, respectively.

2.2 The crossover operator of GA

Of the three operators of GA, crossover is well-known to be the most important one because it significantly affects the performance of GA. Previous crossover operators, such as PMX[16], OX[17], POS[18], and cycle crossover[19], only took into account the position of the node or its order when generating offspring. This is generally considered as "blind" recombination. Other crossover operators, such as ERX[9] and DPX[20], only utilize some local information, such as edge length, to construct offspring, and pay no or limited attention to the global information that may be helpful in search of better solutions. To overcome the drawbacks of previous crossover operators, we proposed a new pheromone-based method, which utilizes both local (edge length and adjacency relations) and global information to construct offspring. The global information used here is pheromone trails on the edges of 1-PDTSP, as is done by artificial ants in the ant colony optimization (ACO) algorithm[21].

When generating an offspring, the pheromone-based crossover operator first randomly selects a customer i as the first customer in offspring. Then, the operator will search customer i in the two parents, and identify the immediate predecessors and successors, named neighbors of customer i. The nearest feasible customer j among unvisited customers in the neighbors will be selected as the next customer in offspring. If all customers in neighbors have been visited or all of them cannot be feasibly added to the end of the route, the next customer will be selected according to the following pseudo-random-proportional rule.

$$j = \begin{cases} \arg\max_{k \in S} \{[\tau_{ik}(t)] \cdot [\eta_{ik}]^\alpha\}, & \text{if} \quad p \leqslant p_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where $\tau_{ik}(t)$ is the value of pheromone trail on edge (i, k) at time t; η_{ik} ($\eta_{ik} = 1/d_{ik}$) is the visibility of customer k from customer i; α is the parameter that determines the relative importance of pheromone trail and the visibility; S is the set of customers that have not been visited yet and can be feasibly added to the end of the route; p is a value chosen randomly with uniform probability in $[0, 1]$; p_0 ($0 \leqslant p_0 \leqslant 1$) is a parameter: the smaller the value of p_0 the higher the probability to make a random selection; and J is a random variable selected according to the distribution defined as follows:

$$p_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)] \cdot [\eta_{ij}]^\alpha}{\sum_{R \in S} [\tau_{ik}(t)] \cdot [\eta_{ik}]^\alpha}, & \text{if} \quad j \in S \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In light of above rule, the proposed crossover operator chooses the "fittest" next customer with probability p_0, and with a probability $(1 - p_0)$ selects the next customer randomly according to the probability distribution defined in formula 4 when there is no feasible customer in neighbors. This process is repeated until all customers Are visited, and then an offspring is generated. The procedure of the pheromone-based crossover operator is summarized in Algorithm 2, where o is offspring, and $o(k)$ is the k-th customer in o.

Algorithm 2. The pheromone crossover operator.

```
begin
  $k = 1$, $o(0) = 0$ (depot);
  select a random customer that can be feasible visited as
  the first customer, $o(k)$, of $o$;
  while ($k \leqslant n$) do
    search $o(k)$ in parents $p_1$ and $p_2$, respectively;
    if (all neighbors of $o(k)$ in $p_1$ and $p_2$ have appeared in $o$,
        or all neighbors cannot be feasibly added to the end
        of $o$)
        generate a random decimal fraction $p$;
        select the next customer $j$ according to (1);
    else
    select the nearest customer that has not appeared in $o$ from
    the feasible neighbors of $o(k)$ as the next customer $f$;
    end if;
  $k = k - 1$;
  $o(k) \leftarrow f$;
end while;
end.
```

2.3 The local search algorithm

To accelerate the convergence of GA further, we utilized a local search procedure to optimize the offspring generated by the crossover operator. The local search algorithm used in GA is a simple 2-opt[22]. In 2-opt, when an exchange of edges results in a feasible and better solution, it is accepted. Otherwise, the exchange is aborted. In order to shorten the search time, for each customer i, the 2-opt procedure only allows the edge (i, j) to be inserted into the tour, where j is one of the cl closest customers to i.

2.4 The mutation operator of GA

After being optimized by 2-opt, the offspring will be mutated with probability p_m to avoid stagnation. In the proposed GA, we used a simple 3-exchange mutation operator. The 3-exchange operator randomly selects 3 cities first, and then exchanges the places of them. Obviously, there are 5 possible combinations besides the offspring itself, and the best feasible one will be selected as the result of mutation. If all the 5 possible tours are infeasible, the offspring is not changed.

2.5 The initialization and update of pheromone trails

In the proposed GA, pheromone trails are used to store the global information utilized by the crossover operator. Like the Max-Min ant system (MMAS)[23], the pheromone trails are initialized to $\tau_{\max}$, and will be limited to an interval $[\tau_{\min}, \tau_{\max}]$ during generations. $\tau_{\min}$ and $\tau_{\max}$ are defined as follows:

$$\tau_{\max} = \frac{1}{1 - \rho} \cdot \frac{1}{f(S^{gb})} \tag{3}$$

$$\tau_{\min} = \frac{\tau_{\max}}{2n} \tag{4}$$

where $0 < \rho < 1$ is the pheromone trail persistence, $f(S^{gb})$ is the cost of the globally best solution S^{gb}, and n is total number of cities. At the end of each generation, the pheromone trails are updated on the basis of the globally best solution. The pheromone update is done as follows:

$$\tau_{ij}(t + 1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{gb} \tag{5}$$

where

$$\Delta\tau_{ij}^{gb} = \begin{cases} \dfrac{1}{f(s^{gb})}, & \text{if } (i, j) \in S^{gb} \\ 0, & \text{otherwise.} \end{cases}$$

Formula (5) dictates that only those edges belonging to the globally best solution will be reinforced.

3 Computational results

3.1 Implementation and parameters setting

The proposed GA is implemented in C under the Visual C++.net environment and runs on a Pentium M 1.6 GHz PC with 256 MB of RAM, and computing time is indicated in seconds.

Based upon working experience, the population size *pop_size* was set at 30, cl at 20, crossover and mutation probabilities at 0.8 and 0.1, respectively. The number of iteration steps was set at 300 for instances with less than 100 customers, and at 500 for large instances ($n \geqslant 100$). The parameters considered here are those that have a direct effect on the pheromone-based crossover operator, including $\alpha\{0, 1, 2, 3, 5\}$, $p_0\{0.5, 0.7, 0.9\}$, and $\rho\{0.8, 0.9, 0.95\}$. Simulation studies show that the following values $\alpha = 3$, $p_0 = 0.9$, $\rho = 0.95$ outperform other tested values.

To illustrate the role of the pheromone trails in crossover operator, we modified it by deleting the first multiplier

$[\tau_{ik}(t)]$ in (1) and (2). This modified version of the crossover operator was then used to benchmark the pheromone-based crossover operator.

3.2 Benchmark instances and results

All experiments were tested on the benchmark instances of the TSPPD in [3]. The first class of instances, 26 instances with customers from 6 to 261, is derived from the VRP instances, while the second class, with n=25, 50, 75, 100, 150, and 200, is randomly generated instances (see [3] for details).

Gendreau et al[3]. proposed H algorithm and tabu search (TS) algorithm for TSPPD, and the TS was proved to be better than H algorithm. In addition, the incomplete optimization algorithm proposed by Hernández-Pérez and Salazar-González[5] also outperformed their first heuristic. Hence, we compared our hybrid GA with the TS in [3] and the incomplete optimization algorithm in [5].

Tables 1 and 2 show the computational results for two classes of instances, respectively. In both tables, column "β" indicates the parameter used for determining the demands of customers; column "TS2/TSP" denotes the average over the 26 instances of the percentage ratios of the TS[3] solution value with respect to the optimal TSP solution value; column "UB2/TSP" gives the average percentage ratio of the incomplete optimization algorithm[5] solution value with respect to the optimal TSP solution value; columns "TS2-t" and "UB2-t" present the average computing time needed by the TS and the incomplete optimization algorithm, respectively; column "Best/TSP" indicates the average percentage ratio of the best result in 30 independent runs of our GA with respect to the optimal TSP result; column "Avg./TSP" denotes the corresponding average result in 30 runs of our GA; column "Std. Dev." indicates the standard deviation for each subset of instances; and column "GA-t" shows the average computing time needed by our algorithm. The last column gives the corresponding average result in 30 runs of GA using the modified version of the crossover operator mentioned above.

Table 1 Computational results on the first class of instances

β	TS2/TSP^a	TS2-t^b	UB2/TSP	UB2-t^b	Best/TSP	Avg./TSP	Std. Dev.	GA-t	GA without pheromone
0	100.51	3.10	100.05	0.93	100.05	100.24	1.24	1.41	100.72
0.05	103.45	2.18	100.6	0.87	100.04	100.20	1.34	1.41	103.75
0.10	104.34	2.31	100.72	1.07	100.08	100.25	1.10	1.42	104.85
0.20	106.16	2.26	100.90	1.02	100.06	100.23	1.24	1.43	105.15

^{a}Results taken from Hernández-Pérez and Salazar-González[5]; ^{b}CPU seconds in an AMD 1.333 GHz PC.

Table 2 Results on the second class of instances

β	n	TS2/TSP^a	TS2-t^b	UB2/TSP	UB2-t^b	Best/TSP	Avg./TSP	Std. Dev.	GA-t	GA without pheromone
0	25	100.00	0.16	100.00	0.07	100.00	100.00	0	0.25	100.00
	50	100.20	0.75	100.00	0.17	100.00	100.08	0	0.51	100.25
	75	100.78	1.82	100.00	0.44	100.00	100.30	2.12	0.90	101.82
	100	101.27	3.24	100.10	0.80	100.10	100.54	3.25	2.25	102.34
	150	102.37	8.35	100.34	2.55	100.33	100.90	4.16	4.25	103.25
	200	103.13	15.27	100.35	5.74	100.36	101.02	4.53	6.93	104.83
0.05	25	107.36	0.18	102.70	0.08	100.00	100.00	0	0.24	100.00
	50	103.95	0.60	100.17	0.19	100.00	100.08	0	0.50	101.44
	75	110.13	1.32	100.83	0.73	100.12	100.34	1.36	0.89	103.20
	100	107.23	2.37	100.39	0.97	100.13	100.63	3.83	2.28	103.55
	150	108.72	5.46	100.35	2.71	100.26	100.95	4.87	4.15	104.35
	200	108.57	11.15	100.69	5.74	100.57	101.56	5.84	6.71	105.90
0.10	25	108.21	0.17	101.18	0.10	100.00	100.00	0	0.24	100.00
	50	106.34	0.63	100.47	0.20	100.00	100.08	0	0.51	101.25
	75	113.28	1.42	101.32	1.00	100.15	100.40	1.54	0.95	102.90
	100	111.53	2.60	100.50	1.44	100.12	100.60	3.02	2.25	104.67
	150	110.90	6.19	100.68	3.52	100.46	100.93	3.59	4.18	105.30
	200	111.31	11.93	100.76	5.71	100.72	101.47	5.41	6.92	106.06
0.20	25	107.28	0.20	102.59	0.13	100.00	100.00	0	0.24	100.00
	50	106.53	0.72	100.79	0.25	100.00	100.08	0	0.50	101.80
	75	114.25	1.44	101.77	1.11	100.12	100.45	2.17	0.88	102.47
	100	113.09	2.61	100.96	1.94	100.20	100.72	4.13	2.26	104.01
	150	111.69	6.01	101.02	3.75	100.60	101.26	4.87	4.30	105.22
	200	113.28	11.85	100.99	8.54	100.85	101.82	5.31	6.94	106.77
∞	25	105.64	0.18	101.32	0.09	100.00	100.00	0	0.25	100.00
	50	110.86	0.73	102.47	0.58	100.00	100.03	0	0.51	100.92
	75	111.86	1.42	100.91	0.69	100.13	100.40	1.63	0.92	101.88
	100	110.34	2.59	100.87	1.45	100.15	100.45	2.59	2.29	103.16
	150	112.85	5.78	100.63	3.52	100.39	100.95	4.75	4.22	105.24
	200	113.03	11.21	100.83	10.50	100.71	101.60	5.64	6.98	105.90

^{a}Results taken from Hernández-Pérez and Salazar-González[5]; ^{b}CPU seconds in an AMD 1.333 GHz PC.

As shown in Tables 1 and 2, our hybrid GA gives better or the same results for all instances (Wilcoxon test, Best/TSP versus TS2/TSP (p=5.6 E-7), Best/TSP versus UB2/TSP (p=3.2 E-6)) except one subset of instances with ($\beta = 0, n = 200$). Moreover, our algorithm is quite robust under different trials, in which the average results in 30 runs for more than a half (18 out of 35) of subsets of instances are better than or the same as the results obtained by the TS[3] and the incomplete optimization algorithm[5]. However, the results obtained by the GA without pheromone are obviously worse than those found by the GA using pheromone-based crossover operator, and the GA only works well in some small instances with no more than 50 customers.

As for the comparison of the computing time, both the TS and the incomplete optimization algorithm were run on an AMD 1.333 GHz PC (>649 MFlops[24]), while the Pentium M 1.6 GHz PC running the GA has an estimated power of 352 MFlops[24]. The scaled time was computed as follows: if T_a is the duration and P_a the computer power for one algorithm a, its scaled time is $T_a \cdot (P_a/P_g)$, with P_g standing for the power of the computer that running Gendreau′s TS algorithm[3]. The scaled times of algorithms for instances with different problem sizes are given in Table 3. From these gross equivalences, we can see that our algorithm is obviously faster than the TS on all instances. When compared to the incomplete optimization algorithm, the GA seems to be slower when solving instances with 25 customers. However, the computing time of our GA increases slowly with the growth of problem size, and for instances with more than 50 customers, our algorithm needs less time than the incomplete optimization algorithm.

Table 3 Scaled times for instances with different problem sizes

n	TS2	UB2	GA
25	0.18	0.09	0.14
50	0.69	0.28	0.28
75	1.48	0.79	0.50
100	2.68	1.32	1.24
150	6.36	3.21	2.29
200	12.28	7.25	3.74

To sum up, the results clearly show the superiority of our algorithm in both accuracy and speed over previous heuristics, which should be attributed to the pheromone-based crossover operator that utilizes both local and global information to construct offspring.

4 Conclusions

In this paper, we described a new GA for the TSPPD. In our algorithm, a pheromone-based crossover operator for GA was proposed. The pheromone-based crossover operator utilizes both local information, including edge lengths and adjacency relations, and global information stored as pheromone trails. In addition, a local search procedure, 2-opt, is integrated into the GA to accelerate convergence. The proposed algorithm was tested on benchmark instances in literature, and computational results show that our hybrid GA outperforms any existing heuristics for TSPPD in both accuracy and speed.

For future research, it may be worthwhile to merge other solution improvement methods to improve the performance of the hybrid GA for the TSPPD. In addition, the pheromone-based crossover operator can be extended to solve other routing problems, such as the VRPPD. Currently, this issue is being investigated, and the authors consider this research avenue to be promising.

References

[1] G. Mosheiov. The Travelling Salesman Problem with Pickup and Delivery. *European Journal of Operational Research*, vol. 79, no. 2, pp. 299–310, 1994.

[2] S. Anily, G. Mosheiov. The Traveling Salesman Problem with Delivery and Backhauls. *Operations Research Letters*, vol. 16, no. 1, pp. 11–18, 1994.

[3] M. Gendreau, G. Laporte, D. Vigo. Heuristics for the Traveling Salesman Problem with Pickup and Delivery. *Computer and Operations Research*, vol. 26, no. 7, pp. 699–714, 1999.

[4] R. Baldacci, E. Hadjiconstantinou, A. Mingozzi. An Exact Algorithm for the Traveling Salesman Problem with Deliveries and Collections. *Networks*, vol. 42, no. 1, pp. 26–41, 2003.

[5] H. Hernández-Pérez, J. J. Salazar González. Heuristics for the One-commodity Pickup-and-delivery Traveling Salesman Problem. *Transportation Science*, vol. 38, no. 2, pp. 245–255, 2004.

[6] M. Gendreau, A. Hertz, G. Laporte. The Traveling Salesman Problem with Backhauls. *Computer and Operations Research*, vol. 23, no. 5, pp. 501–508, 1996.

[7] F. A. Tang, R. D. Galväo. A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery Service. *Computer and Operations Research*, vol. 33, no. 3, pp. 595–619, 2006.

[8] P. Chen, H. K. Huang, X. Y. Dong. An Ant Colony System Based Heuristic Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pickup. In *Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications*, IEEE Press, Harbin, China, pp. 136–141, 2007.

[9] H. D. Nguyen, I. Yoshihara, K. Yamamori, M. Yasunaga. Implementation of an Effective Hybrid GA for Large-scale Traveling Salesman Problems. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 37, no. 1, pp. 92–99, 2007.

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbour, USA, 1975.

[11] A. Misevicius. An Improved Hybrid Genetic Algorithm: New Results for the Quadratic Assignment Problem. *Knowledge-Based Systems*, vol. 17, no. 2-4, pp. 65–73, 2004.

[12] C. Prins. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. *Computer and Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.

[13] B. J. Park, H. R. Choi, H. S. Kim. A Hybrid Genetic Algorithm for the Job Shop Scheduling Problems. *Computer & Industrial Engineering*, vol. 45, no. 4, pp. 597–613, 2003.

[14] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, Berlin, Germany, 1994.

[15] D. Whitley, H. K. Huang, X. Y. Dong. The Genitor Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, USA, pp. 116–121, 1989.

[16] D. E. Goldberg, R. Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, Pittsburgh, USA, pp. 154–159, 1985.

[17] G. Syswerda. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, USA, pp. 332–349, 1991.

[18] I. M. Oliver, D. J. Smith, J. R. C. Holland. A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In *Proceedings of 2nd International Conference on Genetic Algorithms and their Application*, Lawrence Erlbaum Associates, Mahwah, USA, pp. 224–230, 1987.

[19] H. D. Nguyen. Hybrid Genetic Algorithms for Combinatorial Optimization Problems, Ph. D. dissertation, University of Miyazaki, Miyazaki, Japan, 2004.

[20] P. Merz, B. Freisleben. Genetic Local Search for the TSP: New Results. In *Proceedings of IEEE International Conference on Evolutionary Computation*, IEEE Press, Indianapolis, USA, pp. 159–164, 1997.

[21] S. D. Shtovba. Ant Algorithms: Theory and Applications. *Programming and Computer Software*, vol. 31, no. 4, pp. 167–178, 2005.

[22] S. Lin. Computer Solutions of the Traveling Salesman Problem. *Bell Labs Technical Journal*, vol. 44, no. 10, pp. 2245–2269, 1965.

[23] T. Stützle, H. H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[24] J. J. Dongarra. Performance of Various Computers Using Standard Linear Equations Software, Technical Report CS-89-85, Department of Computer Science Department, University of Tennessee, TN, USA, 2007.

Fang-Geng Zhao graduated from Vehicle Management Institute, PRC, in 1999, and received the M. Sc. degree from Military Traffic Institute, PRC, in 2003. He is currently a Ph. D. candidate in University of Science and Technology Beijing, PRC.

His research interests include logistics management and optimization.

Jiang-Sheng Sun graduated from Ordnance Engineering Institute (OEI), PRC, in 1998, and received the M. Sc. degree from OEI, PRC, in 2001. He is currently a Ph. D. candidate in University of Science and Technology Beijing, PRC, and an engineer in Ordnance Technology Research Institute, Shijiazhuang, PRC.

His research interests include logistics management and optimization.

Su-Jian Li graduated from University of Science & Technology Taiyuan, PRC, in 1984, and received the Ph. D. degree from University of Science & Technology Beijing, PRC, in 1993. He is currently a professor in University of Science and Technology Beijing, PRC.

His research interests include logistics management, optimization, and information system.

Wei-Min Liu graduated from Northeastern University, PRC, in 1996. He is currently a Ph. D. candidate in University of Science and Technology Beijing, PRC.

His research interests include logistics management and optimization.