Soft Comput (2009) 13:95–101
DOI 10.1007/s00500-008-0312-1

FOCUS

A new grouping genetic algorithm approach to the multiple traveling salesperson problem

Alok Singh · Anurag Singh Baghel

Published online: 21 May 2008

Abstract The multiple traveling salesperson problem (MTSP) is an extension of the well known traveling salesperson problem (TSP). Given $m > 1$ salespersons and $n > m$ cities to visit, the MTSP seeks a partition of cities into m groups as well as an ordering among cities in each group so that each group of cities is visited by exactly one salesperson in their specified order in such a way that each city is visited exactly once and sum of total distance traveled by all the salespersons is minimized. Apart from the objective of minimizing the total distance traveled by all the salespersons, we have also considered an alternate objective of minimizing the maximum distance traveled by any one salesperson, which is related with balancing the workload among salespersons. In this paper, we have proposed a new grouping genetic algorithm based approach for the MTSP and compared our results with other approaches available in the literature. Our approach outperformed the other approaches on both the objectives.

Keywords Constrained optimization · Heuristic · Grouping genetic algorithm · Multiple traveling salesperson problem

A. Singh (✉)
Department of Computer and Information Sciences,
University of Hyderabad, Hyderabad 500046,
Andhra Pradesh, India
e-mail: alokcs@uohyd.ernet.in

A. S. Baghel
Department of Electronics and Communication,
Banasthali Vidyapith Jaipur Campus, Sarojini Marg,
Jaipur 302001, Rajasthan, India
e-mail: anuragsbaghel@yahoo.com

1 Introduction

The multiple traveling salesperson problem (MTSP) is an extension of the well known traveling salesperson problem (TSP). Given a list of n cities to be visited by a salesperson, the TSP seeks an itinerary that visits each city exactly once and minimizes the total distance traveled by the salesperson. In MTSP, there are m salespersons instead of one to visit $n > m$ cities. The MTSP seeks a partition of n cities into m groups as well as an ordering among the cities in each group, so that each group of cities is visited by exactly one salesperson in their specified order in such a way that each city is visited once and only once and the total distance traveled by all the salespersons is minimized. Like two recent works on MTSP (Carter and Ragsdale 2006; Brown et al. 2007), we have also considered an alternate objective of minimizing the maximum distance traveled by any single salesperson. This objective is related with balancing the workload among salespersons. Clearly, both versions of the MTSP are NP-Hard as for $m = 1$ they reduce to the TSP, which is NP-Hard (Garey and Johnson 1979). The MTSP is harder than the TSP as it requires determining which cities to allot to each salesperson as well as optimal ordering among the cities within each salesperson's allotted cities.

A number of other problems can also be modeled as the MTSP. Carter and Ragsdale (2002) cited the application of the MTSP in a job scheduling problem, where there are multiple parallel production lines. The vehicle scheduling problem (VSP) can also be modeled as the MTSP (Carter and Ragsdale 2006). Carter and Ragsdale (2006) cited another application of the MTSP for a variant of the TSP, where a salesperson visits n cities over a period of time spanning m weeks but returns to the home city during weekends.

Carter and Ragsdale (2006) proposed a genetic algorithm (GA) for the MTSP that uses a new two-part chromosome

representation and related genetic operators. Brown et al. (2007) proposed a grouping genetic algorithm (Falkenauer 1998) that uses a chromosome representation and genetic operators derived directly from those proposed in Falkenauer (1998). Among the earlier works, Tang et al. (2000) proposed the one-chromosome representation for the MTSP problem and applied it to solve the hot rolling production scheduling problem. The two-chromosome representation used in Malmborg (1996) and Park (2001) for the VSP can also be used for the MTSP. In fact, both Carter and Ragsdale (2006) and Brown et al. (2007) have compared the performance of their genetic algorithms against the genetic algorithms using one-chromosome and two chromosome representations. All the aforementioned chromosome representations are discussed in detail in Sect. 2.

For a detailed survey of MTSP and its variations, its practical applications, exact and heuristic solution approaches proposed so far, the reader is referred to Bektas (2006).

In this paper, we have also proposed a grouping genetic algorithm (GGA) for the MTSP. However, our GGA uses different chromosome representation and genetic operators than those used in GGA of Brown et al. (2007). Moreover, our GGA uses steady-state population replacement model (Davis 1991), whereas GGA of Brown et al. (2007) uses generational model. We have compared the results of our approach with those reported in Carter and Ragsdale (2006) and Brown et al. (2007).

The rest of this paper is organized as follows: Sect. 2 discusses the different chromosome representations proposed so far for the MTSP problem along with their relative tradeoffs. Section 3 describes our grouping genetic algorithm. Computational results are presented in Sect. 4, whereas Sect. 5 provides some concluding remarks.

2 Chromosome representation techniques for the MTSP

This section surveys the various chromosome representations proposed so far for the MTSP problem. Figure 1 illustrates the

12	3	8	-1	14	1	7	11	4	9	6	-2	16	2	15	10	5	13

(a) One-Chromosome Representation

12	3	8	14	1	7	11	4	9	6	16	2	15	10	5	13
1	3	1	1	2	3	2	1	1	2	2	2	3	3	1	2

(b) Two-Chromosome Representation

12	3	8	14	1	7	11	4	9	6	16	2	15	10	5	13	3	7	6

(c) Two-Part Chromosome Representation

Fig. 1 Different chromosome representations for 16 cities MTSP with three salespersons. **a** One-chromosome representation. **b** Two-chromosome representation. **c** Two-part chromosome representation

one-chromosome (Tang et al. 2000), the two-chromosome (Malmborg 1996; Park 2001), the two-part chromosome (Carter and Ragsdale 2006) representation techniques on a MTSP instance with $n = 16$ cities and $m = 3$ salespersons.

The one-chromosome technique represents a MTSP solution by a single chromosome of length $n + m - 1$. A solution is represented as a permutation of n cities and $m - 1$ dummy values. These dummy values can be anything other than the city values and separate tour of one salesperson from another. In the example of Fig. 1a, negative numbers are used as dummy values. In this example, the first salesperson will visit cities 12, 3 and 8 in that order, the second salesperson will visit cities 14, 1, 7, 11, 4, 9, 6 in that order and the third salesman will visit cities 16, 2, 15, 10, 5, 13 in that order.

The two-chromosome technique employs two chromosomes of length n to represent a MTSP solution. One chromosome represents a permutation of n cities and the other chromosome tells which city is assigned to which salesperson. In the example given in Fig. 1b, cities 12, 8, 14, 4, 9, 5 will be visited by the first salesperson in that order, cities 1, 11, 6, 16, 2, 13 will be visited by second salesperson in that order and cities 3, 7, 15, 10 will be visited by the third salesperson in that order.

The two-part chromosome technique, as the name implies divides the chromosome into two parts. The first part of length n represents a permutation of n cities and the second part of length m gives the number of cities assigned to each salesperson. Therefore, the total length of the chromosome is $n + m$ in this representation. The m values present in the second part of the chromosome must sum to n in order to represent a valid solution. In the example of Fig. 1c, where shaded region shows the second part, the first salesperson will visit cities 12, 3 and 8 in that order, the second salesperson will visit cities 14, 1, 7, 11, 4, 9, 6 in that order and the third salesman will visit cities 16, 2, 15, 10, 5, 13 in that order.

All the above chromosome representations suffer from the problem of redundant solutions, i.e., many different chromosomes can represent the same MTSP solution. The disadvantage of the redundancy is that the representation space (space of all possible chromosomes) can be much larger than the problem space (space of all possible solutions to the problem). As GA works in representation space rather than problem space therefore GA has to search a larger space. This can impair the performance of GA severely.

Some redundancy is inherent in any permutation based representation for the MTSP because we are representing tours by linear permutations, whereas in reality tours are circular permutations, for example, all the three linear permutations 12, 3, 8; 3, 8, 12 and 8, 12, 3 correspond to the same circular permutation and hence represent the same tour. This inherent redundancy is reduced to its minimum for those MTSP instances where all the salespersons start and end their tour in the same city called home city. For these instances

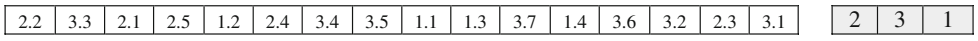

Fig. 2 Chromosome representation used in Brown et al. (2007)

the home city can be removed from the chromosome. In this case, if the instance is asymmetric then each tour is represented uniquely and if the instance is symmetric then each tour can be represented only in one of the two possible ways.

Though, Carter and Ragsdale (2006) showed that the size of the representation space is much less, specially for large m, for the two-part chromosome representation than one-chromosome or two-chromosome representations, all these representations, in fact, have much more redundancy than the inherent redundancy. In the example of Fig. 1a, the chromosome obtained after simply detaching the first four genes from the rest and reinserting these genes in the same order between the gene with value -2 and the gene with value 16 will represent exactly the same solution as the original chromosome. In the example of Fig. 1b, the two chromosomes obtained after swapping alleles 1 and 2 everywhere in the second chromosome and keeping the first chromosome unchanged will represent exactly the same solution as the original two chromosomes. In the example of Fig. 1c, detaching the first three genes from first part of the chromosome and reinserting these genes in the same order between the gene with value 6 and the gene with value 16 of the first part and then swapping the values of the first two genes in the second part will produce a chromosome that will represent exactly the same solution as the original.

Actually in the MTSP, only the grouping of cities, i.e., which city belongs to which group, and ordering among cities in each group matters. As long as groups and ordering among the cities in each group are same, it is immaterial which group is assigned to which salesperson. All these chromosome representation techniques fail to incorporate this basic property in their representation schemes thereby allowing more redundant solutions. Moreover, all these representations are used in a GA (Carter and Ragsdale 2006; Brown et al. 2007) that uses order crossover (Davis 1991) which was originally designed for the TSP. In case of the GA using two-part chromosome representation, order crossover was used for the first part of the chromosome and the asexual crossover was used for the second part (Chatterjee et al. 1996). In these representation schemes if two chromosomes, which represent the same solution, are crossed, they may produce a child that represents a solution quite different from the parents, i.e., crossover may fail to pass traits from parents to child.

Chromosome representation of Brown et al. (2007) is inspired from the chromosome representation described in Falkenauer (1998). This representation consists of two sections—main section and group section. The main section of the chromosome consists of n real-valued genes and group section consist of m integer-valued genes. The integer part of the real-valued gene i in the main section indicates the salesperson assigned to city i, whereas fractional part determines the order in which city i is visited. Group section simply shows the different groups present in the solution in the order in which they appear in the main section. This is a deviation from the representation described in Falkenauer (1998), where groups can appear in any order in the group section. Figure 2 illustrates this representation on the same TSP instance as used in Fig. 1. The shaded region in Fig. 2 shows the group section. This example represents a solution where cities 9, 5, 10, 12 will be visited by first salesperson in that order, cities 3, 1, 15, 6, 4 will be visited by second salesperson and cities 16, 14, 2, 7, 8, 13, 11 will be visited by the third salesperson in that order. The group section contains 2, 3, 1 because these groups appear in the main section in that order. This chromosome representation also suffer from the problem of redundant solutions, for example, if we interchange 1 and 2 in the integer part of every gene in the main section wherever these values occur and adjust the group part accordingly, i.e., change it to 1, 3, 2, then it represents exactly the same solution. However, Brown et al. (2007) used this representation in a GGA where genetic operators work on group section and therefore if two different chromosomes representing the same solution are ever crossed then they produce a child that also represents the same solution.

3 The grouping genetic algorithm

We have developed a steady-state grouping genetic algorithm (GGA-SS) for the MTSP. The GGA-SS is inspired by the steady-state grouping genetic algorithm proposed in Singh and Gupta (2007) for the one-dimensional bin-packing problem. The main features of GGA-SS are described below:

3.1 Chromosome representation

We have represented chromosome as a set of m tours, i.e., there is no ordering among the tours. With such a representation, there is no redundancy other than the inherent redundancy in representing individual tours. We have explicitly taken care of this inherent redundancy by disallowing such kind of redundant solutions in the population to coexist at any generation. Figure 3 shows our chromosome representation on the same MTSP instance as used in Fig. 1. There are three tours—12, 3, 8; 14, 1, 7, 11, 4, 9, 6 and 16, 2, 15, 10, 5, 13 in the solution. However, unlike all chromosome

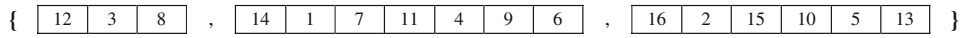

Fig. 3 Our chromosome representation

representations discussed in the previous section, these tours are not assigned to specific salespersons.

We have also designed our genetic operators in such a way that their results are independent of any ordering among tours. This is different from Brown et al. (2007), where result obtained after the crossover depends on the ordering in the group section of the chromosome, i.e., ordering among tours. Though, not used in Brown et al. (2007), Falkenauer (1998) proposed an inversion operator to mitigate the effect of this ordering. In our case, there is no need of inversion operator.

3.2 Fitness

Our fitness function is same as the objective function. As we are considering two different objectives for the MTSP therefore we have two different fitness functions. In both the cases we have to minimize the value of the fitness function.

3.3 Selection

We have used k-ary tournament selection for selecting chromosomes for crossover and mutation. For selecting the two parents for crossover, we have used $k = 2$ and selected more fit candidate with probability p_{better}. We have used $k = 3$ to select a chromosome for mutation, where the candidate with better fitness is always selected. The reason for using this aggressive selection strategy for mutation is that our mutation operator is designed in such a way that more fit chromosome has greater chance of generating a better chromosome upon mutation.

3.4 Crossover

Our crossover operator is derived from the crossover operator proposed in Singh and Gupta (2007). It works in two stages. The first stage iteratively constructs the child chromosome. During each iteration, it selects one of the two parents uniformly at random and copies the most promising tour from this parent to the child. Next it deletes all the cities belonging to this tour from the tours of both the parents by connecting the predecessor of each city to its successor and tour lengths are updated accordingly. This process is repeated $m-1$ times. The definition of the most promising tour differs for the two objectives. For the objective of minimizing the total distance traveled by all the salespersons, the most promising tour is the tour having the smallest ratio of tour length to the number of cities in that tour. For the objective of minimizing the maximum distance traveled by any one salesperson, the most promising tour is the shortest tour. Ties are broken arbitrarily in both the cases. In either case, while determining the most promising tour, only those tours are considered, which have at least two cities, i.e., have non zero tour length.

Clearly, after the end of the first stage some cities left unassigned, the second stage, one by one, reassigns these cities to the tours in an iterative fashion. The second stage does this in one of the two ways. The first method focuses on generating good solutions, whereas second method focuses on generating diverse solutions. Second method is used scarcely. During each iteration, the first method selects an unassigned city uniformly at random and inserts it into the tour whose length increases least by this insertion. To determine this, we have to check all possible insertion positions in every tour. For the objective of minimizing the maximum distance traveled by any one salesperson, we avoid assigning a city to the longest tour even if its cost increases least among all the tours by the insertion of the city. A city is assigned to the longest tour only when its cost increases least and assigning that city to the next least cost tour make that tour the longest tour. The second method, during each iteration, selects both a tour and an unassigned city uniformly at random and inserts the selected city into the selected tour at a position where insertion results in least increase in the tour length. The purpose of this method is to generate diverse solutions and hence maintain diversity in the population.

3.5 Mutation

The mutation operator tries to make a copy of the selected chromosome. However, each city from a tour of the selected chromosome is copied to the corresponding tour of the child with probability p_{cp}, otherwise the city is included in the list of unassigned cities. All these unassigned cities will be inserted into the tours by following the same process as used in the second stage of the crossover.

Similar to Singh and Gupta (2007), here also crossover and mutation are used in a mutually exclusive manner, i.e., at each generation either the crossover is used or the mutation is used but not the both. Crossover is applied with probability p_{c}, otherwise mutation is used.

3.6 Replacement policy

The generated child is first tested for uniqueness against the existing population members. If it is unique then it is included in the population in place of worst fit member. If the generated child is not unique then it is discarded.

3.7 Initial population generation

Each member of the initial population is generated by following an iterative procedure. During each iteration, our procedure selects a tour and an unassigned city both uniformly at random, and inserts the selected city into the selected tour at a position where insertion leads to smallest increase in tour length. Each newly generated chromosome is checked for uniqueness against the population members generated so far, and, if it is unique then it is included in the initial population, otherwise it is discarded.

4 Computational results

We have implemented the GGA-SS in C and executed it on a Pentium 4 system with 512 MB RAM, running at 3.0 GHz under Red Hat Linux 9.0. In all our experiments with GGA-SS, we have used a population of 100 chromosomes, $p_c = 0.8$, $p_{better} = 0.8$, $p_{cp} = 0.66$. During the second stage of the crossover, we have used second method with probability 0.05, i.e., around 5% offsprings are generated using this method. All these parameter values have been chosen empirically. We have tested the GGA-SS on the same test problems as used in Carter and Ragsdale (2006) and Brown et al. (2007). Test problems used in Carter and Ragsdale (2006) are Euclidean, two dimensional symmetric problems with 51, 100 and 150 cities. There are three test problems with 51 cities, one for each value of m in $\{3, 5, 10\}$, four test problems with 100 cities, one for each value of m in $(3, 5, 10, 20)$, 5 test problems with 150 cities, one for each value of m in $\{3, 5, 10, 20, 30\}$. So there are 12 problems altogether. Test problems used in Brown et al. (2007) are originally the test problems of TSP and are taken from TSPLIB. These problems also have 51, 100 and 150 cities and same values of m for each city. So there are twelve test problems in this data set also. The GGA-SS has been executed on each test problem ten times, each time with a different random seed. During each execution, the GGA-SS is allowed to run for 100,000 iterations. Similar to Carter and Ragsdale (2006) and Brown et al. (2007), we have also put two restrictions on the solutions - all salespersons start and end their tours in the same city called home city, every salesperson must visit at least one city in addition to the home city. The second constraint is a must for the objective of minimizing the total distance traveled by all the salespersons. Without this constraint, there is a possibility of getting solutions in which not all m salespersons are utilized or even a solution in which only a single salesperson is utilized. This is due to the fact that as we increase the number of salespersons the total distance traveled by all the salespersons also increases. As every salesperson has to start and end his tour in the home city so with the increase in the number of salespersons, the number of trips into and out of home city increases. This leads to the increase in the total distance traveled by all the salespersons (Carter and Ragsdale 2006). Therefore, without this constraint, a GA whose objective is to minimize the total distance traveled by all the salespersons can generate solutions in which less than m salespersons are utilized or a solution in which only a single salesperson is utilized, .i.e., the MTSP may reduce to the TSP. Though, Carter and Ragsdale (2006) and Brown et al. (2007) have put this constraint only for the objective of minimizing the total distance traveled by all the salespersons, we have put this constraint for both the objectives. In fact, this constraint helps GA in generating good solutions for the objective of minimizing the maximum distance traveled by any one salesperson. This is due to the fact that if there are empty tours in the solution then some cities may be transferred from the longest tour to an empty tour thereby reducing the length of the longest tour.

To test the performance of their proposed methods both Carter and Ragsdale (2006) and Brown et al. (2007) used genetic algorithms with one chromosome and two chromosome representations. Hereafter, the two-part chromosome representation based genetic algorithm of Carter and Ragsdale (2006) will be referred as GA2PC, the grouping genetic algorithm proposed in Brown et al. (2007) will be referred as GGA-BRC, the genetic algorithm with one chromosome representation will be referred as GA1C and the genetic algorithm with two chromosomes representation will be referred as GA2C.

4.1 Objective I: minimizing the total distance traveled by all the salespersons

Table 1 compares the performance of GGA-SS with GA2PC, GA1C and GA2C in terms of average total distance traveled by all the salespersons during ten runs on each of the 12 test problems used in Carter and Ragsdale (2006). Data for GA2PC, GA1C and GA2C are taken from Carter and Ragsdale (2006). Table 1 also reports the average execution time in seconds for GGA-SS under time column. Table 1 clearly shows the superiority of GGA-SS over GA2PC, GA1C and GA2C. On every instance GGA-SS finds the solution of least average cost. Carter and Ragsdale (2006) allowed GA2PC, GA1PC and GA2C to execute for 5, 10 and 15 min, respectively on 51, 100 and 150 cities. instances on a 1.7 GHz Pentium M system with 1GB RAM. As the termination criterion and the systems used in executing the algorithms are different therefore it is not possible to exactly compare the running times but we can safely say that GGA-SS finds better solutions much faster than any of GA2PC, GA1C and GA2C.

Table 2 compares GGA-SS with GGA-BRC, GA1C and GA2C on each of the 12 test problems used in Brown et al. (2007) using the same criterion as used in Table 1. Data for GGA-BRC, GA1C and GA2C are taken from Brown et al.

Table 1 Total distance travelled by all the salespersons averaged over ten runs for GA1C, GA2C, GA2PC and GGA-SS on the 12 test problems of Carter and Ragsdale (2006). Average execution time of GGA-SS is also shown

Instance			GA1C	GA2C	GA2PC	GGA-SS	Time (s)
num	n	m					
1	51	3	529	570	543	449	3.94
2	51	5	564	627	586	479	3.38
3	51	10	801	879	723	584	3.54
4	100	3	27, 036	30, 972	26, 653	22, 051	6.33
5	100	5	29, 753	44, 062	30, 408	23, 678	8.05
6	100	10	36, 890	65, 116	31, 227	28, 488	6.45
7	100	20	62, 471	95, 568	54, 700	40, 892	7.68
8	150	3	46, 111	48, 108	47, 418	38, 434	15.00
9	150	5	49, 443	51, 101	49, 947	39, 962	11.58
10	150	10	59, 341	64, 893	54, 958	44, 274	9.30
11	150	20	94, 291	100, 037	73, 934	56, 412	11.11
12	150	30	131, 503	143, 476	99, 547	72, 783	6.90

Table 2 Total distance travelled by all the salespersons averaged over ten runs for GA1C, GA2C, GGA-BRC and GGA-SS on the 12 test problems of Brown et al. (2007). Average execution time of GGA-SS is also shown

Instance			GA1C	GA2C	GGA-BRC	GGA-SS	Time (s)
num	n	m					
1	51	3	596	951	924	449	3.82
2	51	5	726	1, 180	882	476	3.35
3	51	10	1, 015	1, 403	1, 001	586	3.37
4	100	3	46, 929	80, 195	79, 347	22, 100	6.22
5	100	5	51, 509	97, 741	70, 871	23, 398	8.56
6	100	10	68, 664	129, 000	89, 778	28, 356	6.42
7	100	20	115, 210	151, 935	137, 805	41, 554	7.02
8	150	3	17, 754	37, 373	33, 888	6, 632	24.02
9	150	5	20, 558	38, 766	26, 851	6, 751	15.52
10	150	10	27, 621	39, 906	37, 771	7, 885	7.72
11	150	20	43, 279	41, 484	43, 699	10, 399	8.26
12	150	30	56, 006	45, 364	52, 564	14, 929	8.05

(2007). Brown et al. (2007) also allowed GGA-BRC, GA1C and GA2C to execute for 5, 10 and 15 min, respectively on 51, 100 and 150 cities instances. Conclusions analogous to Table 1 can be drawn here also. In fact, here GGA-SS performs much better than other approaches.

4.2 Objective II: minimizing the maximum distance traveled by any one salesperson

Table 3 compares GGA-SS with GA2PC, GA1C and GA2C on 12 instances used in Carter and Ragsdale (2006), whereas Table 4 compares GGA-SS with GGA-BRC, GA1C and GA2C on 12 instances used in Brown et al. (2007). Performance criterion on each instance is the average maximum distance traveled by any one salesperson during ten runs. For this objective also, GGA-SS completely outperforms all the other genetic algorithms. It always finds the solution of least average cost. Termination criterions used in all the algorithms for this objective are same as used for the previous objectives. Here also, with regard to time we can only say that GA-SS finds better solutions much faster.

5 Conclusions

In this paper we have developed a new steady-state grouping genetic algorithm (GGA-SS) for the multiple traveling salesperson problem (MTSP). We have proposed a chromosome representation scheme with least possible redundancy. Even this redundancy is removed from the population by our replacement policy. We have compared GGA-SS with the approaches presented in Carter and Ragsdale (2006) and Brown et al. (2007). Computational results clearly showed the superiority of our approach over all the other approaches.

Table 3 Maximum distance travelled by any one salesperson averaged over ten runs for GA1C, GA2C, GA2PC and GGA-SS on the 12 test problems of Carter and Ragsdale (2006). Average execution time of GGA-SS is also shown

Instance			GA1C	GA2C	GA2PC	GGA-SS	Time (s)
num	n	m					
1	51	3	234	275	203	161	1.39
2	51	5	173	220	164	119	1.37
3	51	10	140	165	123	112	2.13
4	100	3	14, 722	16, 229	13, 556	8, 542	4.27
5	100	5	11, 193	11, 606	10, 589	6, 852	3.54
6	100	10	9, 960	10, 200	9, 463	6, 370	4.25
7	100	20	9, 235	9, 470	8, 388	6, 359	8.19
8	150	3	19, 875	21, 067	19, 687	13, 268	8.52
9	150	5	15, 229	15, 450	14, 748	8, 660	8.73
10	150	10	12, 154	12, 382	11, 158	5, 875	10.65
11	150	20	10, 206	10, 338	10, 044	5, 252	12.02
12	150	30	9, 626	9, 683	8, 775	5, 247	15.74

Table 4 Maximum distance travelled by any one salesperson averaged over ten runs for GA1C, GA2C, GGA-BRC and GGA-SS on the 12 test problems of Brown et al. (2007). Average execution time of GGA-SS is also shown

Instance			GA1C	GA2C	GGA-BRC	GGA-SS	Time (s)
num	n	m					
1	51	3	329	374	343	160	1.42
2	51	5	277	276	224	119	1.53
3	51	10	208	183	137	112	2.36
4	100	3	30, 733	35, 203	31, 372	8, 555	4.51
5	100	5	26, 005	26, 764	19, 827	6, 826	3.74
6	100	10	17, 522	16, 525	13, 703	6, 370	4.29
7	100	20	12, 971	10, 920	10, 670	6, 359	8.18
8	150	3	11, 455	13, 882	13, 544	2, 425	8.49
9	150	5	8, 916	9, 456	8, 174	1, 761	9.40
10	150	10	5, 820	5, 506	5, 268	1, 581	10.33
11	150	20	4, 284	3, 473	3, 623	1, 554	15.57
12	150	30	3, 795	3, 482	2, 876	1, 554	19.96

In fact, on the test problems considered solution values found by our method are always much better than all the other methods.

Acknowledgments We are grateful to Prof. A. E. Carter and Prof. E. C. Brown for providing us their test problems and answering our queries regarding their approaches.

References

Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34:209–219

Brown EC, Ragsdale CT, Carter AE (2007) A grouping genetic algorithm for the multiple traveling salesperson problem. Int J Inf Technol Dec Mak 6:333–347

Carter AE, Ragsdale CT (2002) Scheduling pre-printed newspaper advertising inserts using genetic algorithms. Omega 30:415–421

Carter AE, Ragsdale CT (2006) A new approach to solving the multiple traveling salesperson problem using genetic algorithms. Eur J Oper Res 175:245–257

Chatterjee S, Carrera C, Lynch L (1996) Genetic algorithms and traveling salesperson problem. Eur J Oper Res 93:490–510

Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York

Falkenauer E (1998) Genetic algorithms and grouping problems. Wiley, Chicester

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, San Francisco

Malmborg C (1996) A genetic algorithm for service level based vehicle scheduling. Eur J Oper Res 93:121–134

Park YB (2001) A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. Int J Prod Econ 73:175–188

Singh A, Gupta AK (2007) Two heuristics for the one-dimensional bin-packing problem. OR Spectr 29:765–781

Tang L, Liu J, Rong A, Yang Z (2000) A multiple traveling salesman problem model for hot rolling schedule in Shanghai Baoshan Iron & Steel Complex. Eur J Oper Res 124:267–282