Journal of Economic and Social Research 4 (2), 1-26

A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem

M. Fatih Tasgetiren[1]

Abstract. This paper presents a genetic algorithm to solve the orienteering problem, which is concerned with finding a path between a given set of control points, among which a start and an end point are specified, so as to maximize the total score collected subject to a prescribed distance constraint. Employing several sets of test problems from the literature, the performance of the genetic algorithm is evaluated against problem specific heuristics and an artificial neural network optimizer.

JEL Classification Codes: C60, C61.

Key Words: Orienteering problem, genetic algorithm, near feasibility threshold, penalty function.

1. Introduction

Given a set of control points with associated scores along with the start and end points, the orienteering problem (OP) deals with finding a path between the start and end points in order to maximize the total score subject to a given distance budget, denoted by DMAX. Due to the fact that distance is limited, tours may not include all points. It should be noted that the OP is equivalent to the Traveling Salesman Problem (TSP) when the time is relaxed just enough to cover all points and the start and end points are not specified.

To model the orienteering problem, we denote V as the set of control points and, E as the set of edges between points in V. Then, the complete graph can be defined by $G = \{V, E\}$. Each control point, i, in V has an associated score $S_i \geq 0$ whereas the start point 1 and the end point n

[1] Management Department, Fatih University, 34500 Buyukcekmece, Istanbul, Turkey. Email: ftasgetiren@fatih.edu.tr

have no scores. The distance, d_{ij} between the points i and j is the nonnegative cost for each edge in E or the cost of traveling between points i and j. So, the objective is to find a path from the start point 1 to the end point n through a subset of control points so that the total score collected from the visited points will be maximized without violating the given distance constraint. The mathematical model of the OP is given as follows:

$$Max \quad \sum_{i=1}^{n}\sum_{j=1}^{n} S_i x_{ij}$$

$$Subject \quad to:$$

$$\sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1,$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \le 1, \qquad k = 2,\dots,n-1$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} x_{ij} \le DMAX$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 1,\dots,n$$

The OP has applications in vehicle routing and production scheduling. Golden, Assad, and Dahl (1984) discussed certain applications of the OP to customer/vehicle assignment and inventory/routing problems. Golden, Levy, and Vohra (1987) also applied the OP to a vehicle routing problem in which oil tankers are routed to service stations in different locations. The total score of the route is maximized while minimizing the route distance without violating the DMAX constraint. Balas (1989) modeled certain types of production scheduling problems as the OP. These problems are concerned with product-mix planning of production to maximize the total profit without violating the production time constraints. Keller (1989) modified his multi-objective vending problem as the OP in which there is a trade off between maximizing reward potential and minimizing the travel cost. Kantor and Rosenwein (1992) presented the OP with time windows in which a point can only be visited within a specified time interval. This approach seems to be promising for potential applications such as bank and postal delivery, industrial refuse collection, dial-a-ride services and school bus routing. Golden, Levy and Vohra (1987) have shown that the OP is NP-hard.

2. Background

Most research concerning the OP falls into two categories: heuristic approaches or exact solution methods. Tsiligirides (1984) introduced two heuristic approaches to solve the OP. The first approach is based on the Monte Carlo method where a large number of solutions are generated to select the best one among them as a final solution. Each route is constructed in such a way that every point not included in the route is assigned a desirability measure, which is given by

$$A_j = \left(\frac{S_j}{C_{last,j}} \right)^{4.0}$$

Where S_j is the associated score of point *j* and $C_{last,j}$ is the distance between last point and point *j*. Then, four points are selected according to their higher desirability measure in order to normalize them as follows:

$$P_j = \frac{A_j}{\sum_{t=1}^{4} A_t} \quad j=1, 2, 3, 4$$

Point *j* is determined by random sampling so that it is the new last point inserted onto the current route. This insertion procedure is repeated until no additional point can be included in the route without violating the DMAX constraint. 3000 routes are generated using the above procedure and the final solution is the one with the highest score among them.

Tsiligirides's deterministic algorithm is based on the procedure developed by Wren and Holiday (1972). In this approach, the search space is divided into a number of sectors determined by two concentric circles and an arc of length. Sectors are varied by changing the two radii of the circles and by rotating the axes four times with $\pi/2$ differences in order to look into different possibilities. The route construction process is stopped when all the nodes in that particular sector have been visited or if there is no way to visit another node of the same sector without violating the DMAX constraint. Tsiligirides (1984) examined 48 cases in each run and the final solution is the one with the highest score.

Golden, Levy and Vohra (1987) presented a heuristic algorithm to solve the OP in three steps: route construction, route improvement, and center of gravity improvement. In the first step, the initial route constructed is based on a weighted ranking including a score rank S_j, a distance to center-of-gravity rank C_j, and a sum of the distances to the two foci of an ellipse rank E_j. Each node not in the route is assigned a weighted measure to determine the next node to be inserted. The weighted measure is given by

$$W_j = \alpha S_j + \beta C_j + \gamma E_j$$

where $\alpha + \beta + \gamma = 1$. In the second step, the current solution is improved by applying a 2-opt heuristic, which is followed by a cheapest insertion procedure in which the maximum possible numbers of nodes are inserted onto the route without violating the DMAX constraint. In the third step, the center of gravity of the route obtained in the second step is computed and these three steps are repeated until the two successive routes are identical to each other. Different routes are constructed in this way and the final solution is the one with the highest score.

Golden, Wang and Liu (1988) designed a new and improved heuristic for the OP. Tsiligrides's randomization concept is embedded in the center of gravity procedure together with learning capabilities. For any node j not in the route, its score S_j, center-of-gravity distance C_j, and ellipse distance E_j are the basic factors to determine which point to be inserted onto the route. The weighted measure is given by

$$W_j = \alpha S_j + \beta C_j + \gamma E_j$$

where $\alpha + \beta + \gamma = 1$. Based on the previous work in Golden B.L., Levy L., and Vohra R., (1987), it is recommended that the score measure is the most important factor, center-of-gravity measure is the intermediate and the ellipse measure is the least important factor. So they used $\alpha=0.7$, $\beta=0.2$ and $\gamma=0.1$. At each step, the five nodes with the largest weighted averages are examined and a node is chosen randomly to be inserted onto the route. Then a feasibility check is performed to see whether or not the resulting solution is feasible. If feasible, another insertion is made, otherwise, a node is removed from the resulting route to gain feasibility again. This process is repeated until the route cannot be improved any more by addition of further nodes. The route generated by the insertion procedure is conveyed to the improvement step to apply a 2-opt local search routine in which attempts are made to reduce the total distance of the route by swapping some nodes. If there is a decrease in the total distance of the route through the 2-opt routine, insertions are again performed by using the same procedure summarized above. A candidate route is finally obtained after the 2-opt routine. On the other hand, a new route is established by recomputing the center of gravity, updating the weighted average and repeating the selection and insertion procedures. For each problem, five centers of gravity that are located at the centers of five squares are initially used. For each center, they generated 20 solutions among which the one with the highest score is the final solution.

Keller (1989) modified his algorithm for a multi-objective vending problem to solve the OP. His modified algorithm consists of two stages: path construction and improvement. In the path construction stage, two alternative approaches are evaluated to determine the node to be inserted next onto the path. The first approach is deterministic in nature and computes the simple ratio of reward potential of the node j to the penalty of $P_{last,j}$. The second approach is a stochastic approach in which the scores of all points that are both not in the current solution and also can be feasibly inserted are normalized. Then random sampling is used to determine the next node that will be inserted onto the route. Having the starting solution constructed, two routines are applied to reduce the total penalty of the path by altering the route sequence. In the second stage, three strategies, namely; 'one in – zero out', 'one in – one out', and 'one in – two out' are applied to obtain possible increases in the total score while keeping the DMAX constraint satisfied. For further improvement, cluster exchange is performed by removing one cluster and inserting another one to increase the total score.

Ramesh and Brown (1991) developed an efficient four-phase heuristic for the OP. Their heuristic consists of vertex insertion, cost improvement, vertex deletion and maximal insertions. In the first phase, single insertion and double insertion rules are employed to construct an initial solution. Then this solution is improved by 2-opt and 3-opt local search routines respectively. The third phase attempts to achieve a decrease in the length of the path in such a way that one point is removed and another is inserted. The final phase deals with a systematic attempt to include each unvisited node in the path. The last three phases are repeated to find a very good solution.

Chao (1993) and Chao et al. (1996) developed a fast and effective heuristic for the orienteering problem for which the results are the best so far in the literature. Chao's heuristic basically consists of initialization and improvement steps. In the initialization step, he generated *L* solutions by a greedy method, among which the one with the highest score is selected as the initial solution. In the improvement step, first, two-point exchange is applied to the initial solution on a record-to-record improvement basis. Then, one-point movement is applied to the current solution generated by two-point exchange procedure. The movement is made whenever it is feasible and it increases the total score. Otherwise, a feasible movement with the least decrease in the total score is performed. Then, the 2-opt procedure is applied to the current solution to decrease the length of the current solution. This procedure explained above is repeated until the ***K*** and ***I*** loop values are achieved.

It should be noted that Chao (1996) extended the single member OP to the multi-member OP in which there is a team of salesmen and they compete to maximize the total score of their route while minimizing the distance of their route. The extension is made for 2-member, 3 member and 4-member teams and 358 problem instances were generated and solved by his heuristic.

In addition to the heuristic algorithms, there exists a number of exact solution methods for the OP in the literature. Two of them are developed for a variant of the OP in which the start and end points are the same. Laporte and Martello (1990) developed a branch and bound method to solve test problems with at most 20 points. Ramesh, Yoon, and Karwan (1992) developed an optimal algorithm to solve a variant of the OP. Their algorithm is based on Lagrangian relaxation with an improvement procedure within a branch and bound framework and solves test problems with at most 150 points. Sokkappa (1990) presented two exact algorithms and a heuristic to solve the OP. While the heuristic is based on the method of Golden, Wang and Liu (1988), the exact methods are based on a branch and bound for the knapsack problem and the TSP, respectively. Leifer and Rosenwein (1993) introduced 0-1 integer programming and a cutting plane method by which a tight upper bound on the optimal objective function value to each of 49 benchmark test problems is provided. Pillai (1992) developed an exact procedure in which the OP is treated as a special case of the TSSP+1 problem. The exact procedure is based on a branching and cutting plane method. First, the relaxed LP is solved to examine the violated constraints and then the violated constraints are added to the relaxed LP to be resolved. These procedures are performed repeatedly until no violated constraint is found. Pillai used Tsiligirides's 49-benchmark problems to test her exact procedure. Fishetti, Gonzales and Toth (1998) recently presented a branch-

and-cut algorithm to find optimal solutions for the OP. They claim their algorithm can solve problem instances with 500 nodes optimally within acceptable CPU time. As an artificial neural network application by Wang, Sun, Golden and Jia (1995) presented a continuous Hopfield neural network to solve the OP. Their results are similar to those by Chao (1996), but never better in any problem instance. Tasgetiren and Smith (2000) presented a genetic algorithm for the OP. Their application was computationally expensive, different representation, simple penalty function and crossover operator were used. In this study, we present a variable length permutation representation, injection crossover and the adaptive penalty function which provided computationally less expensive and robust solutions.

3. Genetic Algorithm

Genetic algorithms (GA) are a family of parallel search heuristics inspired by the biological process of natural selection and evolution (Gen and Cheng, 1997). In GA optimization, solutions are encoded into chromosomes to establish a population being evolved through generations. At each generation, parents are selected and mated from the population to carry out the crossover operator leading to new solutions called children. Then, some of the individuals are mutated or perturbed. Finally, they are pooled together to select new individuals for next generation.

Figure 1 shows the overall scheme of the genetic algorithm (GA) for the OP, and this is detailed in the subsections of this section. First, the initial population of size λ is constructed probabilistically based on the distance and DMAX information for each problem. At each generation, two parents are determined by tournament selection of size 2 and random selection, respectively, to produce an offspring through order crossover. This process is conducted in a loop until μ offspring (population size *x* crossover probability) are produced. Hence, the size of the population is increased to $(\lambda + \mu)$ at the end of each generation. Then, local searches consisting of add, omit, replace and swap operators are applied to individuals selected randomly from the offspring generated by the crossover operator. For the population of the next generation, the tournament selection with size of 2 is used to establish the population, again among $(\lambda + \mu)$ individuals, thus maintaining a λ size of population. This procedure is repeated until the stopping criterion is achieved.

```
Initialize population P of size $\lambda$
Evaluate $\lambda$ individuals in P
While not termination do {
        Select 2* $\mu$ individuals from P
        Crossover individuals to produce
         $\mu$ offspring
        Mutate some individuals in $\mu$
        Add $\mu$ offspring to $\lambda$ individuals in P
         Evaluate $(\lambda + \mu)$ individuals in P
        Select $\lambda$ individuals from $(\lambda + \mu)$ individuals in P }
End While
End Algorithm
```

Figure 1. Pseudo code for the genetic algorithm

3.1 Variable Length Permutation Representation

Variable length permutation representation (Hinterding 1994) is used for each chromosome. In an OP tour, points are listed in the same order they are visited by keeping the start and end points the same in every chromosome. This is also called a *path* or *order* representation. An example is given below.

1, 3, 13, 11, 15, 12, 2, 5, 8, 10, 20, **21**

It should be noted that each chromosome in the population has different length of size based on the point insertion probability.

3.2. Initial Population

In order to construct the initial population, we used a point insertion probability based on the DMAX constraint and distance information for each problem as shown in Figure 2. This is done so that the initial population generally follows the DMAX constraint, i.e. not too few and not too many points are included in each tour. A list of the *n* points is generated in random order. Then, each point is assigned a random number [0,1] and, if this is less than the point insertion probability, the point is inserted in the tour, otherwise it is not included in the tour.

```
Define DMAX
Define maxloop
Set count=0 and loop=0;
Do{
    Produce a random number, $R_N$, between [1,N]
    Produce a list, $R_L$, between [1,N] randomly
    Generate a sub list, $R_S$, by taking the first $R_N$ part of the random list $R_L$
    Compute the total distance, $d_{RS}$, of the sub list $R_S$
    If $d_{RS}$ <=DMAX, then count=count+1
    loop=loop+1
    }while (loop<=maxloop)
  point insertion probability=(1-count)/maxloop
```

Figure 2. Point insertion probability

3.3 Injection Crossover Operator

A modified injection crossover operator (Falkenauer and Delchambre, 1992) is used in this study to manipulate the chromosome of different lengths. In injection operator, an insertion point from first parent and a sub list from the second parent are chosen. Then, the sub list is injected into the first parent at the insertion point. Duplicate points are deleted outside the sub list to get a proto-child. Then the proto child is fitted to the size of the first parent. The following is an example (remember that points 1 and 21 are fixed).

Insertion point

Parent 1:	**1, 9, 18, 16, 15,** $\downarrow$ **13, 2, 21**
Parent 2:	**1, 3, 13, 11, 10, 12, 2, 5, 8, 15, 20, 21**
P-Child:	**1, 9, 18, 16, 15, 2, 5, *8, 15, 20*, 21**
Offspring:	**1, 9, 18, 16, 15, 2, 5, 21**

3.4 Local Search Methods

Regarding the mutation scheme, local searches are employed to enrich and diversify the population by the use of add, omit, replace and swap operators, which are applied to only newly created offspring. Through these local search operators, solutions near the feasibility border (i.e., where DMAX is active) are improved while solutions far from the feasibility border are worsened. For example, an offspring is randomly selected, and add operator is applied 10 times. The best one is then replaced into the population of μ offspring. This procedure is repeated for omit, swap and, replace operators respectively. This entire procedure of four local search mechanisms is repeated to generate the specified number of mutants (i.e., population size *x* mutation probability). An example for add, omit, replace, and swap operators is given in Figure 3 to 6 respectively.

Offspring:
1, 3, 11, 18, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21
- Find a point that is not in the offspring, say **14.**
- Add it to the offspring in a position selected randomly

Modified Offspring:
1, 3, 11, 18, 16, 15, 13, 2, 5, 14, 6, 12, 8, 10, 20, 21
- Evaluate the offspring.
- Repeat ten times.
- Select the best and replace into offspring population.

Figure 3. Pseudo code for the add operator

Offspring:
1, 3, 11, 18, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21
- Find a point that is in the offspring, say **18.**
- Omit it from the offspring.

Modified Offspring:
1, 3, 11, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21
- Evaluate the offspring.
- Repeat ten times.
- Select the best and replace into offspring population.

Figure 4. Pseudo code for the omit operator

Offspring:

1, 3, 11, 18, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21

- Find a point that is not in the offspring, say point **4.**
- Select a point randomly in the offspring, say point **15**, and replace it with the randomly selected point not in the offspring, that is, point 4.

Modified Offspring:

1, 3, 11, 18, 16, 4, 13, 2, 5, 6, 12, 8, 10, 20, 21

- Evaluate the offspring.
- Repeat ten times.
- Select the best and replace into the offspring population.

Figure 5. Pseudo code for the replace operator

Offspring:

1, 3, 11, 18, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21

- Find two points in the offspring randomly, say points **15** and **12**
- Swap two points.

Modified Offspring:

1, 3, 11, 18, 16, 12, 13, 2, 5, 6, 15, 8, 10, 20, 21

- Evaluate the offspring.
- Repeat ten times.
- Select the best and replace into the offspring population.

Figure 6. Pseudo code for the swap operator

Local search methods explained above significantly improved the performance of the GA. This is illustrated in Figure 7. For example, the solution ***x*** might result in the solution ***y*** by adding a point which is not included in the solution, the solution ***z*** might result in the solution ***k*** by either replacing a point which is not included in the solution or simply deleting a point from it, finally the solutions ***x*** or ***z*** might result in the solutions ***y*** or ***k*** by swapping two points in them.

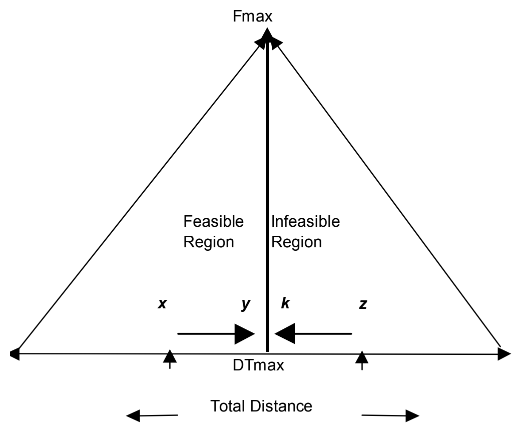

Figure 7. The effect of local search

3.5 Adaptive Penalty Function

Since the OP is a constrained problem and search can benefit from considering infeasible solutions (Coit and Smith 1996b). An adaptive penalty function is used to penalize the infeasible solutions, which employs the notion of a "Near-Feasibility Threshold" (NFT) for the constraints as proposed in Coit and Smith (1996a).The NFT defines a threshold distance from the feasible region to infeasible region as being close to feasibility. The orienteering problem has only one constraint where the total distance is restricted to a DMAX value. So the following penalty function is employed in this study.

$$F_p = F - (F_{all} - F_{feas})\left(\frac{TD - DMAX}{NFT}\right)^{\alpha}$$

where F_p is the penalized fitness value of the objective function, F is the unpenalized value of the objective function, F_{all} represents the unpenalized value of the best solution found, and F_{feas} denotes the value of the best feasible solution found, and TD is the total distance of the tour. α is a predefined severity parameter. NFT is the “Near-Feasible Thresholds” for the constraint and take adaptive form as follows:

$$NFT = \frac{NFT_0}{1+\lambda \times N_g}$$

where NFT_0 is the initial value of the NFT, which is set to 10% of the constraint’s value in this study. N_g represents the number of generations, and λ denotes a constant which assures that the entire region between NFT_0 and zero is searched. Thus, solutions are adaptively penalized according to their distance from feasibility.

3.6 Parameters

The GA parameters have been determined experimentally. The population size, crossover and mutation probabilities are taken as 100, 0.70 and 0.40 respectively. The severity parameter α is set to 4. GA is terminated after the best feasible solution remains unchanged for δ generations. δ varies with DMAX since a larger DMAX allows a longer string of visited points and needs a larger δ. δ increases from 5 to 100 as DMAX increases.

4. Computational Results

The GA is coded in Borland C and implemented on an Intel P4 1.33GHz PC with 256 MB memory. All computations are conducted using real precision without rounding or truncating values. The precision of the final solution length has been rounded to one decimal point as in Chao (1993 and 1996). We examined three sets of test problems with 32, 21 and 33 points provided by Tsiligirides (1984) as well as the corrected problem set 1 by Chao (1993 and 1996). GA is applied on 67 problems from the four test sets and compared to Tsiligirides’s stochastic algorithm T , Chao’s heuristic C and an artificial neural network NN.

Computational results are given in Tables A1-A4 in Appendix A. In addition, distance and sequence information for the best solutions that GA generated, minimum, average and the standard deviation of ten replications for each instance to all the test problems are given in tables B1-B4 in Appendix B. In tables A1-A4, the better results that GA produced against previous methods are designated by "+", the results that GA was not outperformed are designated by "-", and the results that GA and the previous methods produced the same scores have no sign.

From tables A1-A4, it is clear that GA outperforms Tsiligirides's stochastic algorithm. For the 67 problems, GA produced better scores than T in 46 of 67 while the rest was the same. In comparison to the artificial NN, GA produced better scores than the artificial NN in 8 of 67 while in one case the artificial NN was better and the rest was the same. In comparison to the Chao's heuristic, GA produced the same results in 64 of 67 cases. In three cases, GA produced better results. These comparisons are summarized and depicted in Table 1 and Figure 8. However, it should be noted that in two of the five cases (DMAX=30 and DMAX=40) in the second data set, the distances are rounded to one decimal point, as in Chao. We also included alternative solutions for these two cases without rounding to one decimal point, in which the first reported solutions slightly violate DMAX.

Table 1. The number of better results that the GA produced

	GA vs. T	GA vs. C	GA vs. NN
PSET1	11	0	0
PSET2	8	2	3
PSET3	20	0	3
PSET4	7	1	2

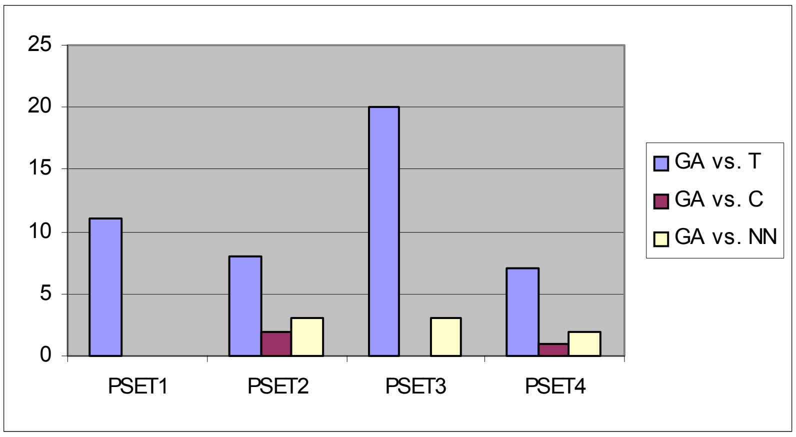

Figure 8. The number of better results that the GA produced

5. Conclusions

In this paper, a GA with an adaptive penalty function is presented to solve the orienteering problem which has variety of industrial applications. The advantage of GA is due to the fact that search is conducted from a population of solutions whereas heuristic approaches start from an initial solution and rely on a point-to-point improvement on the initial solution.

The GA is applied to 67-benchmark test problems and compared to the problem specific heuristics and the artificial neural network optimizer. It should be noted that GA was able to find better results for three problem instances than the current results in the literature. From the computational results, it can be seen that the GA performed very well across a wide variety of problem instances and degrees of constraint.

References

Balas E, (1989), "The prize collecting traveling salesman problem", *Networks* Vol.19, 621-636.

Chao I-M., (1993), "Algorithms and solutions to multi-level vehicle routing problems", *Ph.D Dissertation*, Applied Mathematics Program, University of Maryland, College Park, MD.

Chao I-M., Golden B.L., and Wasil E. A., (1996), "A fast and effective heuristic for the orienteering problem", *European Journal of Operational Research*, Vol. 88, 475-489.

Chao I-M., Golden B.L., and Wasil E. A., (1996), "The team orienteering problem", *European Journal of Operational Research*, No. 88, 464-474.

Coit D.W. and Smith A. E., (1996), "Penalty guided genetic search for reliability design optimization", *Computers and Industrial Engineering*, Vol. 30, No. 4, 895-904.

Coit D.W., Smith A. E., Tate M D, (1996), "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems", *INFORMS Journal on Computing*, vol. 8, No. 2, Spring, pp. 173-182.

Falkenauer E, and Delchambre, A, (1992), "A genetic algorithm for bin packing and line balancing", *Proceeding of the IEEE International Conference on Robotics and Automation*, pp. 1186-1193

Fischetti M., Gonzales J.S., and Toth P., (1998), "Solving the orienteering problem through branch-and-cut", *INFORMS Journal on Computing*, Vol.10, No.2, 133-148.

Gen M. and Cheng R., (1997), "Genetic Algorithms and Engineering design", *John Wiley & Sons*, Wiley Series in Engineering Design and Automation.

Golden B. L., Assad A. and Dahl R., (1984), "Analysis of a large-scale vehicle routing problem with inventory component", *Large Scale Systems*, Vol. 7, 181-190

Golden B.L., Levy L., and Vohra R., (1987), "The orienteering problem", *Naval Research Logistics*, Vol. 34, 307-318

Golden B.L., Wang Q, and Liu L., (1988), A multifaceted heuristic for the orienteering problem, *Naval Research Logistics*, Vol. 35, 359-366

Hinterding R, 1994, "Mapping, order-independent-genes and the knapsack problem", *Proceeding of the first IEEE conference on evolutionary computation*, IEEE Press, Orlando, FL

Kantor, M., and Rosenwein, M, (1992), "The orienteering problem with time windows", *Journal of Operational Research Society*, Vol. 43, No. 6, 629-635.

Keller P., (1989), "Algorithms to solve the orienteering problem: a comparison", *European Journal of Operational of Research*, Vol. 41, 224-231.

Laporte G. and Martello S., (1990), "The selective traveling salesman problem", *Discrete Applied Mathematics*, Vol. 26, 193-207

Leifer A.C, and Rosenwein M.S., (1993), "Strong linear programming relaxations for the orienteering problem", *European Journal of Operational Research*, Vol. 73, 517-523.

Pillai R.S., (1992), "The traveling salesman subset-tour problem with one additional constraint (TSSP+1)", *Ph.D. Dissertation*, The University of Tennessee, Knoxville, TN.

Ramesh R., and Brown K. M., (1991), "An efficient four-phase heuristic for the generalized orienteering problem", *Computers and operations Research*. Vol. 18. No.2, 151-165.

Ramesh R., Yoon Y-S., and Karwan M. H., (1992), "An optimal algorithm for the orienteering tour problem", *ORSA Journal on Computing*, Vol.4, No.2, 155-165.

Sokkappa P.R., (1990), "The cost-constrained traveling salesman problem", *Ph.D. Dissertation*, The University of California, Livermore, CA

Tasgetiren M. F, and Smith A. E, (2000), “A genetic algorithm for the orienteering problem, Proceedings of the 2000 Congress on evolutionary Computation”, CEC2000, Vol.2, p.910-915, IEEE, New York

Tsiligirides T., (1984), “Heuristic methods applied to orienteering”. *Journal of Operational Research Society*, Vol. 35, No. 9, 797-809.

Wang Q., Sun X., Golden B. L., and Jia J., (1995), “Using artificial neural networks to solve the orienteering problem”, *Annals of Operations Research*, Vol. 61, 111-120.

Wren, A and Holliday, A, (1972), “Computer scheduling of vehicles for one or more depots to a number of delivery points”, *Operational Research Quarterly*, Vol. 23, 333-344.

Appendix A

Table A1. Comparison of results on data set 1

DMAX	**T**	**C**	**NN**	**GA**	**GA vs. T**	**GA vs. C**	**GA vs. NN**
5	10	10	10	10			
10	15	15	15	15			
15	45	45	45	45			
20	65	65	65	65			
25	90	90	90	90			
30	110	110	110	110			
35	135	135	135	135			
40	150	155	155	155	+		
46	170	175	175	175	+		
50	185	190	190	190	+		
55	195	205	205	205	+		
60	220	225	225	225	+		
65	235	240	240	240	+		
70	255	260	260	260	+		
73	260	265	265	265	+		
75	265	270	270	270	+		
80	270	280	280	280	+		
85	280	285	285	285	+		
Summary of GA vs. previous					**11 +**	**0 +**	**0 +**
					0 -	0 -	0 -

Table A2. Comparison of results on data set 2

TMAX	T	C	NN	GA	GA vs. T	GA vs. C	GA vs. NN
15	120	120	120	120			
20	190	200	200	200	+		
23	205	210	205	**210**	+		+
25	230	230	230	230			
27	230	230	230	230			
30 a	250	265	265	**275 ***	+	+	+
30	250	265	265	**265**			
32	275	300	300	300	+		
35	315	320	320	320	+		
38	355	360	360	360	+		
40 *	395	395	395	**400 ***	+	+	+
40	395	395	395	**395**			
45	430	450	450	450	+		
Summary of GA vs. Previous					**8 +**	**2 +**	**3 +**
					0 -	**0 -**	**0 -**

a. Rounded to one decimal point.
*Best results found so far in the literature

Table A3. Comparison of results on data set 3

TMAX	T	C	NN	GA	GA vs. T	GA vs. C	GA vs. NN
15	100	170	170	170	+		
20	140	200	200	200	+		
25	190	260	250	260	+		+
30	240	320	320	320	+		
35	290	390	390	390	+		
40	330	430	420	430	+		+
45	370	470	470	470	+		
50	410	520	520	520	+		
55	450	550	550	550	+		
60	500	580	580	580	+		
65	530	610	610	610	+		
70	560	640	640	640	+		
75	590	670	670	670	+		
80	640	710	700	710	+		+
85	670	740	740	740	+		
90	690	770	770	770	+		
95	720	790	790	790	+		
100	760	800	800	800	+		
105	770	800	800	800	+		
110	790	800	800	800	+		
Summary of GA vs. previous					20+	0+	3+
					0-	0-	0-

Table A4. Comparison of results on data set 4

DMAX	T	C	NN	GA	GA vs. T	GA vs. C	GA vs. NN
5	10	10	10	10			
10	15	15	15	15			
15	45	45	45	45			
20	65	65	65	65			
25	90	90	90	90			
30	110	110	110	110			
35	135	135	130	135			+
40	150	155	155	155	+		
46	175	175	175	175			
50	190	190	190	190			
55	205	205	205	205			
60	220	220	220	**225***	+	+	+
65	240	240	240	240			
70	255	260	260	260	+		
73	260	265	265	265	+		
75	270	275	275	275	+		
80	275	280	280	280	+		
85	280	285	285	285	+		
	Summary of GA vs. Previous				7+	1+	2+
					0-	0-	0-

* Best result found so far in the literature

Appendix B

Table B1. Sequence Found by the GA for Problem Set 1

DMAX	CPU(s)	Max Score	Min Score	Std	TD	Tour
5	2.2	10	10	0.00	4.14	1, 28, 32
10	4.32	15	15	0.00	6.87	1, 28, 18, 32
15	6.55	45	45	0.00	14.26	1, 27, 31, 26, 20, 19, 32
20	9.23	65	65	0.00	19.60	1 ,27 ,31 ,26 ,22 ,21 ,20 ,19 ,32
25	11.09	90	90	0.00	24.82	1, 27 ,31 ,26 ,22 ,21 ,12 ,11 ,10 , 9 ,32
30	13.72	110	110	0.00	29.57	1 ,28 ,27 ,31 ,26 ,22 ,21 ,12 ,11 , 10 ,9 ,8 ,13 ,32
35	16.03	135	130	1.58	34.08	1 ,28 ,27 ,31 ,26 ,25 ,23 ,22 ,21 , 12 ,11 ,10 ,8 ,9 ,13 ,32
40	18.03	155	150	1.58	38.97	1 ,28 ,27 ,31 ,26 ,25 ,23 ,22 ,21 , 12 ,11 ,10 ,8 ,2 ,3 ,7 ,6 ,32
46	21.73	175	175	0.00	44.61	1 ,28 ,27 ,31 ,26 ,25 ,24 ,23 ,22 , 21 ,12 ,11 ,10 ,8 ,2 ,3 ,7 ,6 ,13 ,32
50	23.16	190	185	1.58	49.79	1 ,28 ,27 ,31 ,26 ,25 ,24 ,23 ,22 , 21 ,12 ,11 ,10 ,8 ,2 ,3 ,7 ,4 ,5 ,6 ,32
55	25.34	205	200	2.58	54.80	1 ,28 ,27 ,31 ,26 ,25 ,23 ,22 ,21 ,12 , 11 ,10 ,8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,18 ,32
60	27.87	220	215	2.11	58.73	1 ,28 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 , 12 ,11 ,10 ,8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,18 ,32
65	33.83	240	235	2.11	64.20	1 ,28 ,17 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 ,8 ,9 ,10 , 11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,20 ,19 ,32
70	34.04	260	255	2.58	69.13	1 ,28 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 ,8 ,10 , 11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,20 ,19 ,32
73	35.47	265	265	0.00	72.23	1 ,19 ,20 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 ,10 , 11 ,9 ,8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,16 ,17 ,29 ,28 ,32
75	35.75	270	260	4.22	74.73	1 ,19 ,27 ,31 ,30 ,26 ,25 ,24 ,23 ,22 ,21 ,12 ,11 , 10 ,9 ,8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,16 ,17 ,29 ,28 ,32
80	35.92	280	275	2.42	79.92	1 ,28 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,3 ,7 ,6 ,13 ,2 ,8 ,9 , 10 ,11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,30 ,31 ,27 ,20 ,19 ,32
85	38.35	285	280	2.42	84.42	1 ,19 ,20 ,27 ,31 ,30 ,26 ,22 ,23 ,25 ,24 ,21 ,12 , 11 ,10 ,8 ,9 ,13 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,16 ,17 ,29 ,28 ,18 ,32

Table B2. Sequence Found by the GA for Problem Set 2

DMAX	CPU Time(s)	Max Score	Min Score	Std	TD	Tour
15	3.78	120	120	0.00	14.25	1 ,12 ,8 ,9 ,10 ,11 ,13 ,14 ,21
20	6.12	200	200	0.00	19.88	1 ,12 ,7 ,6 ,5 ,3 ,2 ,8 ,9 ,10 ,11 ,13 ,14 ,21
23	7.63	210	210	0.00	22.65	1 ,7 ,6 ,5 ,4 ,3 ,2 ,8 ,9 ,10 ,11 ,14 ,21
25	8.26	230	230	0.00	24.13	1 ,12 ,7 ,6 ,5 ,4 ,3 ,2 ,8 ,9 ,10 ,11 ,13 ,14 ,21
27	9.54	230	230	0.00	24.79	1 ,12 ,7 ,6 ,5 ,4 ,3 ,2 ,8 ,9 ,10 ,11 ,14 ,13 ,21
30*	10.78	275	265	4.22	30.01	1 ,7 ,6 ,5 ,3 ,2 ,8 ,17 ,16 ,15 ,9 ,10 ,11 ,13 ,21
30	11.12	265	250	3.72	29.85	1 ,7 ,6 ,2 ,8 ,17 ,16 ,15 ,9 ,10 , 11 ,13 ,14 ,21
32	13.25	300	300	0.00	31.63	1 ,7 ,6 ,5 ,3 ,2 ,8 ,17 ,16 ,15 ,9 , 10 ,11 ,13 ,14 ,21
35	14.25	320	310	3.16	34.51	1 ,7 ,6 ,5 ,3 ,4 ,20 ,19 ,18 ,17 ,9 , 10 ,11 ,13 ,14 ,21
38	15.24	360	355	2.11	37.84	1 ,7 ,6 ,5 ,2 ,3 ,4 ,20 ,19 ,18 ,17 , 8 ,9 ,10 ,11 ,13 ,14 ,21
40*	17.14	400	390	4.38	40.05	1 ,7 ,6 ,5 ,3 ,4 ,20 ,19 ,18 ,16 ,15 , 17 ,9 ,10 ,11 ,13 ,14 ,21
40	3.78	395	385	3.22	39.78	1 ,7 ,6 ,5 ,3 ,4 ,20 ,19 ,18 ,16 ,15 , 17 ,8 ,9 ,10 ,11 ,13 ,21
45	6.12	450	450	0.00	44.44	1 ,12 ,7 ,6 ,5 ,2 ,3 ,4 ,20 ,19 ,18 ,16 , 15 ,17 ,8 ,9 ,10 ,11 ,13 ,14 ,21

* Rounded to one decimal place.

Table B3. Sequence Found by the GA for Problem Set 3

DMAX	CPU(s)	Max Score	Min Score	Std	TD	Tour
15	5.98	170	170	0.00	14.47	1 ,24 ,22 ,7 ,5 ,14 ,4 ,27 ,23, 33
20	9.45	200	190	3.16	19.79	1 ,24 ,22 ,7 ,5 ,28 ,14 ,4 ,3 ,27 ,23 ,33
25	11.56	260	250	3.16	24.46	1 ,24 ,22 ,7 ,5 ,14 ,4 ,20 ,13 ,3 ,23 ,33
30	13.45	320	320	0.00	29.19	1 ,24 ,22 ,7 ,5 ,14 ,28 ,20 ,17 ,13 ,3 ,4 ,27 ,23 ,33
35	16.15	390	380	3.16	34.79	1 ,24 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,16 ,15 ,13 ,3 ,23 ,33
40	18.56	430	420	3.16	38.88	1 ,24 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,16 ,15 ,13 ,3 , 6 ,2 ,32 ,33
45	20.89	470	470	0.00	44.26	1 ,24 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,16 ,15 ,13 ,3 , 6 ,2 ,8 ,29 ,26 ,33
50	22.98	520	510	4.22	49.99	1 ,24 ,22 ,7 ,5 ,4 ,14 ,28 ,20 ,17 ,16 ,15 ,13 ,3 ,6 , 2 ,8 ,31 ,12 ,29 ,26 ,33
55	24.89	550	540	3.16	54.79	1 ,24 ,23 ,27 ,22 ,7 ,5 ,28 ,14 ,4 ,3 ,20 ,17 ,16 ,15 , 13 ,6 ,2 ,8 ,31 ,12 ,29 ,30 ,33
60	27.56	580	570	4.22	59.87	1 ,24 ,22 ,7 ,5 ,27 ,14 ,28 ,4 ,3 ,13 ,20 ,17 ,21 ,16 , 15 ,6 ,2 ,8 ,31 ,12 ,29 ,26 ,33
65	33.82	610	600	4.83	63.75	1 ,24 ,23 ,27 ,22 ,7 ,5 ,28 ,14 ,4 ,3 ,20 ,17 ,21 , 16 ,15 ,13 ,6 ,2 ,8 ,31 ,12 ,29 ,30 ,26 ,33
70	34.15	640	630	5.16	69.57	1 ,24 ,23 ,27 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,21 ,16 , 15 ,13 ,3 ,6 ,2 ,8 ,31 ,12 ,11 ,30 ,29 ,26 ,33
75	35.96	670	660	5.27	74.62	1 ,24 ,25 ,9 ,10 ,18 ,11 ,29 ,12 ,31 ,8 ,2 ,6 ,3 ,13 , 15 ,16 ,17 ,20 ,4 ,14 ,28 ,5 ,7 ,22 ,27 ,23 ,33
80	36.05	710	710	0.00	79.71	1, 24, 22, 7, 5, 28, 14, 4, 20, 17, 16, 15, 13, 3, 6, 2, 8, 31, 12, 29, 30, 11, 19, 18, 10, 9, 25, 33
85	38.05	740	730	4.22	84.98	1 ,23 ,27 ,24 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,16 ,15 ,13 ,3 ,6 , 2 ,8 ,31 ,12 ,11 ,19 ,18 ,10 ,9 ,30 ,29 ,26 ,32 ,33
90	40.52	770	760	5.16	89.31	1, 24, 22, 7, 5, 28, 14, 4, 20, 17, 21, 16, 15, 13, 3, 6, 2, 8, 31, 12, 11, 19, 18, 10, 9, 30, 29, 26, 32,33
95	42.15	790	780	4.83	93.96	1 ,24 ,27 ,23 ,22 ,7 ,5 ,28 ,14 ,4 ,20 ,17 ,21 ,16 ,15 ,13 ,3 , 6 ,2 ,8 ,31 ,12 ,29 ,26 ,30 ,11 ,19 ,18 ,10 ,9 ,25 ,33
100	43.86	800	790	4.22	99.83	1 ,23 ,27 ,14 ,4 ,3 ,13 ,15 ,16 ,21 ,17 ,20 ,28 ,5 ,7 ,22 ,24 , 25 ,9 ,10 ,18 ,19 ,11 ,30 ,26 ,29 ,12 ,31 ,8 ,2 ,6 ,32 ,33
105	45.38	800	800	0.00	104.61	1 ,24 ,7 ,5 ,28 ,22 ,25 ,9 ,10 ,18 ,19 ,11 ,30 ,26 ,29 ,12 , 31 ,8 ,32 ,2 ,6 ,3 ,13 ,15 ,16 ,21 ,17 ,20 ,14 ,4 ,23 ,27 ,33
110	46.98	800	800	0.00	105.85	1 ,24 ,7 ,5 ,22 ,27 ,14 ,28 ,20 ,17 ,21 ,16 ,15 ,13 ,4 ,3 ,6 ,2 , 8 ,31 ,12 ,30 ,25 ,9 ,10 ,18 ,19 ,11 ,29 ,26 ,32 ,23 ,33

Table B4. Sequence Found by the GA for Problem Set 4

DMAX	CPU(s)	Max Score	Min Score	Std	TD	Tour
5	2.2	10	10	0.00	4.14	1, 28, 32
10	4.32	15	15	0.00	6.87	1, 28, 18, 32
15	6.55	45	45	0.00	14.26	1, 27, 31, 26, 20, 19, 32
20	8.71	65	65	0.00	19.60	1 ,27 ,31 ,26 ,22 ,21 ,20 ,19 , 32
25	10.86	90	90	0.00	24.82	1 ,27 ,31 ,26 ,22 ,21 ,12 ,11 ,10 ,9 ,32
30	13.72	110	110	0.00	29.71	1 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 ,19 ,32
35	15.82	135	130	1.58	34.68	1 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 ,11 , 10 ,9 ,13 ,32
40	18.35	155	150	1.58	38.97	1 ,28 ,27 ,31 ,26 ,25 ,23 ,22 ,21 ,12 ,11 , 10 ,8 ,2 ,3 ,7 ,6 ,32
46	21.17	175	175	0.00	45.39	1 ,28 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 , 11 ,10 ,8 ,2 ,3 ,7 ,5 ,32
50	23.36	190	185	1.58	49.19	1 ,28 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 , 11 ,10 ,9 ,8 ,2 ,3 ,7 ,6 ,5 ,18 ,32
55	25.32	205	200	2.58	54.80	1 ,28 ,27 ,31 ,26 ,25 ,23 ,22 ,21 ,12 ,11 , 10 ,8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,18 ,32
60	27.82	225	220	2.11	59.89	1 ,27 ,31 ,26 ,25 ,23 ,22 ,21 ,12 ,11 ,10 , 8 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,16 ,17 ,28 ,32
65	30.39	240	230	3.16	64.24	1 ,27 ,31 ,26 ,25 ,24 ,23 ,22 ,21 ,12 ,11 , 10 ,9 ,8 ,2 ,3 ,7 ,5 ,4 ,14 ,15 ,16 ,17 ,28 ,32
70	32.96	260	250	3.16	69.13	1 ,28 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 , 8 ,10 ,11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,20 ,19 ,32
73	34.69	265	265	2.42	72.16	1 ,28 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 , 8 ,9 ,10 ,11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,19 ,20 ,32
75	34.92	275	265	3.16	74.66	1 ,28 ,30 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 ,8 , 9 ,10 ,11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,20 ,19 ,32
80	41.54	280	275	1.58	79.27	1 ,28 ,30 ,29 ,17 ,16 ,15 ,14 ,4 ,5 ,6 ,7 ,3 ,2 ,9 , 8 ,10 ,11 ,12 ,21 ,22 ,23 ,24 ,25 ,26 ,31 ,27 ,20 ,19 ,13 ,32
85	41.95	285	280	1.58	83.48	1 ,19 ,20 ,27 ,31 ,26 ,25 ,23 ,24 ,22 ,21 ,12 ,11 ,10 ,9 , 8 ,13 ,2 ,3 ,7 ,6 ,5 ,4 ,14 ,15 ,16 ,17 ,29 ,30 ,28 ,18 ,32