The Ninth International Symposium on Operations Research and Its Applications (ISORA'10)
Chengdu-Jiuzhaigou, China, August 19–23, 2010
Copyright © 2010 ORSC & APORC, pp. 89–97

# Models and Algorithms for the Constrained Orienteering Problem

Zhenping Li[1,*] Rui-Sheng Wang[2] Hong-Wei Liu[1]
Wenfeng Zhou[1]

[1]School of Information, Beijing Wuzi University, Beijing 101149,China
[2]School of Information, Renmin University of China, Beijing 100872, China

**Abstract** The constrained orienteering problem (COP) can be expressed as: Given an undirected weighted graph $G = (V,E)$ and a subset $S \subseteq V$. Each node has a score, each edge has a weight indicating distance or time between the two adjacent nodes. The starting node and ending node are specified. Given a fixed amount of weight denoting the total distance or total time, the goal is to determine a walk from the starting node to the ending node through a subset of nodes including all the nodes in $S$, in order to maximize the total score of the walk. This problem is a generalization of the orienteering problem or a special case of the Steiner Tree problem. In this paper, we first formulate the COP problem into an integer linear programming based on network flow theory and solve it to obtain the exact optimal solution for examples of small size. Then we give a heuristic algorithm for solving the large instances of the problem. Finally, we give computational results of both exact and heuristic algorithms which demonstrate the efficient of the algorithm.

**Keywords** Constrained Orienteering Problem; Integer Linear Programming; Heuristic Algorithm

## 1 Introduction

The orienteering problem (OP) is a well established problem in combinatorial optimization which is introduced in [4]. In the orienteering problem, we are given an edge-weighted graph $G(V,E)$, where $V = \{v_1, v_2, \cdots, v_n\}$, each node $v_i \in V$ has an associated non-negative score $w_i$, each edge $e_{ij} \in E$ has a weight $d_{ij}$ representing the length or time taken to travel from node $v_i$ to node $v_j$. Given a starting node $s \in V$ and an ending node $t \in V$, a length or time limit $D_{lim}$. The goal is to find a walk beginning at $s$ and ending at $t$ of total length at most $D_{lim}$ that maximizes the total score of distinct vertices visited by the walk.

Given a subset $S \in V \setminus \{s,t\}$, the constrained orienteering problem (COP) is an orienteering problem with the additional constraint that each node of $S \in V$ must be included in the walk. For the COP, the subset $S$ usually includes a few number of nodes. When $S = \emptyset$, the COP is OP.

Although the OP has many applications in the field of routing, many of these applications are actually suited for the COP. For instance, in [4], the authors describe an

[*]Corresponding author: zxs@amt.ac.cn

application of the OP to the delivery of home heating fuel. In this application, utility managers would assign each customer a score based on their urgency of need for home heating fuel and would select a subset of customers to serve based on need and geography while adhering to total time (or distance) limitations. Urgency would take into account each customers tank size as well as historical and seasonal rates of usage. Further, a company might consider a set of special customers, who must be served since their heat fuel has been used up. By combining these factors, using the COP, the heating fuel company could make a better decision about which customers to serve.

There are many other examples suit to be solved by the COP. Suppose a foreign professor come to China to take part into three meetings which will be held in Beijing, Chengdu, and Shanghai respectively. He wants to visit some other cities of China during his staying in China. But he has limited money to travel 5000 miles. He can obtain a score (satisfaction degree) in each city he visited. The problem is which cities he should choose to visit and which walk he should take along. In other words, the professor wants to find a walk visiting some cities including Beijing, Chengdu and Shanghai such that the total score he obtained is maximum. This problem is a COP problem.

In the past decades, researchers have proposed a large number of heuristics for the OP. Tsiligrides [11] proposed a stochastic algorithm for the OP. Wang et al [7] applied artificial neural networks to the OP and obtained high-quality results. Chao et al [1] applied deterministic annealing to the OP and also obtained high-quality results. Gendreau et al. [10] applied tabu search to the OP and obtained near-optimal solutions to instances with up to 300 nodes.

Although there are many algorithms for the OP, they can not be used for the COP since their constraints are different. In this paper, the COP is formulated into an integer linear programming model based on the network flow theory in section 2, then the optimal solution is obtained by solving the integer linear programming. In section 3, a heuristic algorithm for solving the COP is constructed. Section 4 gives the computational results both on the exact and the heuristic algorithms. Our algorithms are the first approaches for the COP, and high-qualified solutions can be obtained by these algorithms. Finally, the conclusions are gived in section 5.

## 2 The Constrained Orienteering Problem

The constrained orienteering problem (COP) can be formulated as follows: Given an undirected weighted graph $G = (V,E)$ with node set $V = \{v_1, v_2, \cdots, v_n\}$ , edge set $E$ and a subset $S \subseteq V$. Each node $v_i$ has a weight $w_i \geq 0$ which can be interpreted as profit obtained by visiting it. The starting node is $v_1$ and the ending node is $v_n$. Each edge $e_{ij}$ has weight $d_{ij}$ which can be interpreted as distance. The COP is to find a walk $P$ in graph $G$ which satisfies the following conditions. (1) $P$ is starting from node $v_1$ and ending at node $v_n$; (2) $S \subset V(P)$; (3) the total distance of $P$ is no more than a given limit $D_{lim}$; (4)the total score of nodes in walk $P$ is maximal.

Using the network flow theory, suppose there is a flow of size $n-1$ entering into the network from source node $v_1$. Then it flows through one edge and enters into another node. Whenever it flows into a node, a unit of flow is absorbed by the node, the other units will flow out through another edge. Finally, when the flow enters into node $v_n$, all the units will be absorbed which means that node $v_n$ is the ending node where all the flows entering into it will be absorbed. The nodes and edges along the flow consists of the walk from $v_1$ to $v_n$. The COP can be formulated into the following integer linear programming model.

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} w_i x_{ij} \qquad (1.1)$$

$$s.t. \begin{cases} \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 & (1.2) \\ \sum_{i=2}^{n} x_{i1} = \sum_{j=1}^{n-1} x_{nj} = 0 & (1.3) \\ \sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} \leq 1, k = 2, \cdots, n-1 & (1.4) \\ \sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} = 1, k \in S & (1.5) \\ \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \leq D_{lim} & (1.6) \\ x_{ij} \leq f_{ij} \leq M x_{ij}, e_{ij} \in E & (1.7) \\ \sum_{j=2}^{n} f_{kj} = \sum_{i=1}^{n-1} f_{ik} - \sum_{i=1}^{n-1} x_{ik}, k = 2, \cdots, n-1 & (1.8) \\ \sum_{j=1}^{n} f_{1j} = n-1 & (1.9) \\ x_{ij} \in \{0,1\}, f_{ij} \geq 0, i,j = 1,2,\cdots,n & (1.10) \end{cases} \qquad (1)$$

Where $x_{ij}$ is a binary variable, $x_{ij} = 1$ indicates that edge $e_{ij}$ is selected in the walk, while $x_{ij} = 0$ indicates the edge $e_{ij}$ is not selected in the walk. $f_{ij}$ is the units of flow passing through edge $e_{ij}$. $D_{lim}$ is the upper bound of the total distance in $P$. $M$ is a large positive number more than the total number of nodes in the graph.

The objective function maximizes the total score in walk $P$.

Constraint (1.2) means that there is one edge leaving node $v_1$ in walk $P$, and one edge entering into node $v_n$ in walk $P$. Constraint (1.3) indicates there is no edge entering into node $v_1$ or leaving node $v_n$, which means that node $v_1$ is the starting node and node $v_n$ is the ending node of walk $P$. Constraint (1.4) means that for every node $v_k$, if there is an edge entering into it, there must be another edge leaving it. Constraint (1.5) means that every node of subset $S$ must be in the walk $P$. Constraint (1.6) means that the total length of walk $P$ is no more than $D_{lim}$. Constraint (1.7) indicates that if there is one or more units of flow passing through edge $e_{ij}$, then $x_{ij}$ must equal 1, that is, edge $e_{ij}$ is selected in the walk. Constraint (1.8) indicates that every selected edge is in the connected walk. Constraint (1.9) indicates that there are $n-1$ units of flow entering into node $v_1$, which means that the nodes in walk $P$ is no more than $n$. Constraint (1.10) indicates that $x_{ij}$ is a binary variable while $f_{ij}$ is a nonnegative integer variable.

## 3 A Heuristic Algorithm for the COP

Since the OP is NP-hard, and the OP is a special case of the COP, so the COP is NP-hard. For the small size of the COP problem (for example the number of network nodes is no more than 30), we can directly solve the integer linear programming model to obtain global optimal solutions. But for the large instances of the COP problem with nodes more than 50, solving the integer linear problem is time consuming. So we must construct some heuristic or approximate algorithms to solve the COP within reasonable time.

In the following of this section, we will construct a heuristic algorithm for solving the COP.

Suppose $v_1$ is the starting node and $v_n$ is the ending node. $S = \{v_2, v_3, \cdots, v_{k+1}\}$ are $k$ nodes that must be visited.

**Heuristic Algorithm for the COP (HCOP)**:

- **Input:** A graph $G = (V,E)$, $V = \{v_1, v_2, \cdots, v_n\}$, score $w_i$ of each node $v_i$, weight $d_{ij}$ of each edge $e_{ij} = (v_i, v_j) \in E$. $S = \{v_2, v_3, \cdots, v_{k+1}\} \subset V$ , $P = \emptyset$, $T = V$. The maximum distance $D_{lim}$, $W = 0$.
- **Output:** The optimal walk $P$ starting from node $v_1$ and ending at node $v_n$. The total score $W$ of all nodes in the walk $P$.
- **STEP-1**: Solve the TSP problem in the induced graph $G[S \cup \{v_1, v_n\}]$. Denote the optimal walk from $v_1$ to $v_n$ including all nodes of $S$ by $P$. Let $T = T \setminus V(P)$. $W = \sum_{v_i \in P} w_i$. $L = \sum_{e_{ij} \in E(P)} d_{ij}$.
- **STEP-2:** If $L > D_{lim}$, there is no feasible solution, Stop. Otherwise, go to STEP-3.
- **STEP-3:** If $T = \emptyset$, go to STEP-6. Otherwise, let $TT = T$, $FF = \emptyset$, Go to STEP-4.
- **STEP-4:** If $TT = \emptyset$, Go to STEP-5. Otherwise, select a node $v_k \in TT$, find the proper site in $P$ such that if $v_k$ is added in the walk by this site the new walk $P(v_k) = P \cup \{v_k\}$ has the minimal length. Calculate the the length $L(P(v_k))$ of walk $P(v_k)$ and the increase length $\Delta L = L(P(v_k)) - L(P)$.
- If $L(P(v_k)) > D_{lim}$, delete node $v_k$ from both $T$ and $TT$, i.e. $T = T \setminus \{v_k\}$, $TT = TT \setminus \{v_k\}$.
- If $L(P(v_k)) \leq D_{lim}$, then calculate the ratio $R_{v_k} = \frac{w_k}{\Delta L}$ of node $v_k$. Delete node $v_k$ from $TT$, and add node $v_k$ to $FF$. i.e. $TT = TT \setminus \{v_k\}$, $FF = FF \cup \{v_k\}$. Go to STEP-4.
- **STEP-5:** If $FF = \emptyset$, Go to STEP-6. Otherwise, find the node $v_r$ in $FF$ with the maximum ratio, i.e. $R_{v_r} = \max_{v_k \in FF} R_{v_k}$.
- Add node $v_r$ to walk $P$ in the proper site to obtain the new walk $P(v_r) = P \cup \{v_r\}$. Let $P = P(v_r)$, $L = L(P(v_r))$, $W = W + w(v_r)$, $T = T \setminus \{v_r\}$. Go to STEP -3.
- **STEP-6:** Output $P$, $L$, $W$.

By this way, we can obtain a local optimal walk starting from node $v_1$ and ending at node $v_n$ including all nodes in $S$, which has sub-maximum node score sum.

**Computational Complexity of the Algorithm HCOP**

**Theorem 1** Algorithm HCOP requires $O(n^3)$ time to solve the COP problem for $G = (V,E)$, $|V| = n$, $|S| = k$.

**Proof:** The time complexity of algorithm HCOP is determined by two parts: the first is solving the TSP problem in induced graph $G[S \cup \{v_1, v_n\}]$, which take time $k!$. Since $k$ is very small, so we can find the global optimal solution in STEP-1 in no more than $k!$ time. For the large number of $k$, we can use heuristic algorithm for solving the TSP problem in graph $G[S]$ in no more than $O(k^3)$ times. The second part is the loops adding new nodes to the partial path in STEP-4 and STEP-5. For each time a new node is added, it takes time no more than $O(n^2)$, since there are no more than $n - k$ nodes needing to be added, the total times to finish STEP-4 and STEP-5 is no more than $O(n^3)$. So the total time of algorithm HCOP is no more than $O(n^3)$.

## 4 Computational results

We use the example of 27 cities in China [2] [8]. Fig 1 illustrates the sites of 27 cities in the Chinese map. Table 1 includes the longitudes, latitudes, and scores for each of these cities. Each city has a score which equals the sum of the four scores of [2]. These scores are scaled from 1 to 40. The higher the score, the more attractive the city. The distance matrix is obtained from paper [2].

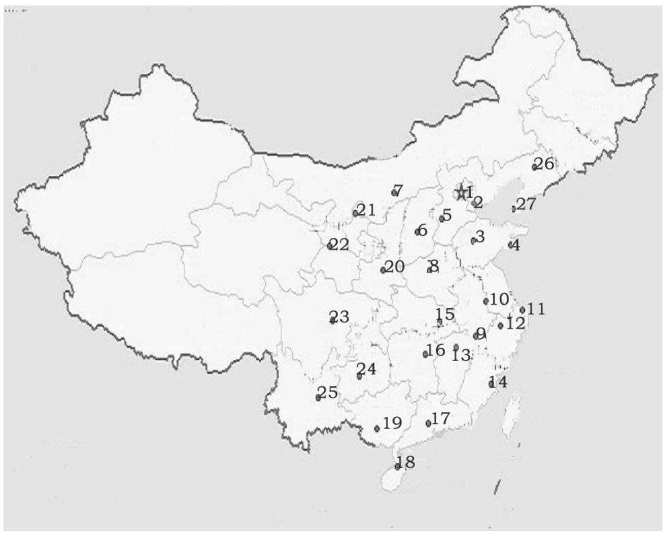

Figure 1: The graph of 27 cities' site in the Chinese map.

Table 1 Locations and Scores of 27 cities in China

| | City | Lon. | Lat. | Score | | City | Lon. | Lat. | Score |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Beijing | 116.40 | 39.91 | 35 | 15 | Wuhan | 114.30 | 30.55 | 26 |
| 2 | Tianjin | 117.18 | 39.16 | 27 | 16 | Changsha | 113.00 | 28.20 | 23 |
| 3 | Jinan | 117.00 | 36.67 | 25 | 17 | Guangzhou | 113.15 | 23.15 | 27 |
| 4 | Qingdao | 120.33 | 36.06 | 23 | 18 | Haikou | 110.35 | 20.02 | 22 |
| 5 | Shijiazhuang | 114.50 | 38.05 | 19 | 19 | Guilin | 110.29 | 25.28 | 22 |
| 6 | Taiyuan | 112.58 | 37.87 | 21 | 20 | Xi'an | 108.92 | 34.28 | 28 |
| 7 | Huhehaote | 111.70 | 40.87 | 22 | 21 | Yinchuan | 106.27 | 38.48 | 22 |
| 8 | Zhengzhou | 113.60 | 34.75 | 21 | 22 | Lanzhou | 103.80 | 36.03 | 24 |
| 9 | Huangshan | 118.29 | 29.73 | 16 | 23 | Chengdu | 104.07 | 30.66 | 24 |
| 10 | Nanjing | 118.75 | 32.04 | 29 | 24 | Guiyang | 106.70 | 26.59 | 22 |
| 11 | Shanghai | 121.45 | 31.22 | 27 | 25 | Kunming | 102.80 | 25.05 | 29 |
| 12 | Hangzhou | 120.15 | 30.25 | 30 | 26 | Shenyang | 123.40 | 41.80 | 24 |
| 13 | Nanchang | 115.88 | 28.35 | 23 | 27 | Dalian | 121.60 | 38.92 | 25 |
| 14 | Fuzhou | 119.30 | 26.10 | 23 | | | | | |

In this section, we select Beijing as both the starting and ending nodes of the walk. we choose different distance limit and different node set that must be included in the walk.

The results obtained by both the exact algorithm and the heuristic algorithm are listed in Table 2 and Table 3. The exact algorithm of solving integer linear programming model is coded by Lingo software, while the heuristic algorithm is coded by MATLAB. Both codes were run on a Windows XP, Pentium-IV with 2.0 GHz speed and 2.0 G of memory.

Table 2 Results on 27 cities of China with given node set $S = \{v_4, v_{20}, v_{23}\}$

| $D_{lim}$ | $S$ | Distance | Score | Walk |
|---|---|---|---|---|
| 4000 | {4,20,23} | 3820.8 | 223 | 1-5-6-23-20-8-3-4-2-1 |
| 5000 | {4,20,23} | 4933.5 | 353 | 1-5-6-20-23-16-15-9-12-11-10-4-3-2-1 |
| 6000 | {4,20,23} | 5996.8 | 441 | 1-3-4-10-11-12-9-15-13-16-19-24-23 -20-8-6-5-2-1 |
| 7000 | {4,20,23} | 6946.7 | 497 | 1-2-5-6-8-20-23-25-24-19-17-16-15-13- -13-9-12-11-10-4-3-1 |
| 8000 | {4,20,23} | 7956.9 | 547 | 1-2-3-27-4-10-11-12-9-13-15-16-17-19- -24-25-23-22-21-20-6-5-1 |
| 9000 | {4,20,23} | 8927.3 | 593 | 1-2-3-5-6-8-20-22-23-25-24-19-17-16- -15-13-14-9-12-11-10-4-27-26-1 |
| 10000 | {4,20,23} | 9990.8 | 637 | 1-2-5-6-7-21-22-23-25-24-19-17-14-12- -11-10-9-13-16-15-20-8-3-4-27-26-1 |
| 11000 | {4,20,23} | 10891.4 | 659 | 1-7-6-5-3-8-20-21-22-23-25-24-19-17- -18-14-13-16-15-9-11-12-10-4-27-26-2-1 |

Table 3 Results on 27 cities of China with given node set $S = \{v_{23}, v_{26}\}$

| $D_{lim}$ | $S$ | Distance | Score | Walk |
|---|---|---|---|---|
| 5000 | {23,26} | 4996.2 | 296 | 1-5-6-20-23-16-15-9-12-11-10-4-3-2-1 |
| 6000 | {23,26} | 5967.3 | 418 | 1-26-27-4-10-11-12-9-13-16-19-24- -23-20-6-5-2-1 |
| 7000 | {23,26} | 6883.7 | 497 | 1-2-26-27-4-10-11-12-9-13-15-16-19- -24-23-20-8-6-5-3-1 |
| 8000 | {23,26} | 7992.4 | 546 | 1-6-5-3-8-20-23-25-24-19-17-16-15-13- -9-12-11-10-4-27-26-2-1 |
| 9000 | {23,26} | 8927.3 | 593 | 1-2-3-5-6-8-20-22-23-25-24-19-17-16- -15-13-14-9-12-11-10-4-27-26-1 |
| 10000 | {23,26} | 9863.5 | 637 | 1-26-27-4-10-11-12-9-14-13-15-16-17-19- -24-25-23-22-21-20-8-6-7-5-3-2-1 |
| 11000 | {23,26} | 10966.5 | 659 | 1-2-3-5-6-8-20-21-22-23-25-24-19-18-17- -14-13-16-15-9-11-12-10-4-27-26-7-1 |

From Table 2 and Table 3, we can see that both algorithms can find the reasonable results under different parameters.

## 5 Conclusion

In this paper, we proposed the constrained orienteering problem (COP), and formulated it into an integer linear programming model. By solving the integer linear programming, we can obtain the global optimal solution of the COP. We also proposed a heuristic algorithm for the COP. Our heuristic is based on the idea of maximum marginal score increase. We applied our exact algorithm to 27 cities of China revised from the literature. The results showed that our algorithm is efficient and performs well on the test problems.

Although we have used our algorithms on a 27-node examples and obtained the optimal solutions. We have not test the algorithms on large size of examples with more than 60 nodes. In the future, we will explore some more algorithms for the large size of COP examples.

Our integer linear programming model for the COP can be revised to solve many other problems, such as the OP problem, the TSP problem and some VRP problems. To solve the OP problem, we only need to delete constraint (1.5) of model 1. To solve the TSP problem, we only need to change the objective function and revise some restraints. In the other words, the TSP problem can be formulated into the following integer linear programming model.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \quad (2.1)$$

$$s.t. \begin{cases} \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 & (2.2) \\ \sum_{i=2}^{n} x_{i1} = \sum_{j=1}^{n-1} x_{nj} = 0 & (2.3) \\ \sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} = 1, k = 2,\cdots,n-1 & (2.4) \\ x_{ij} \le f_{ij} \le Mx_{ij}, e_{ij} \in E & (2.5) \\ \sum_{j=2}^{n} f_{kj} = \sum_{i=1}^{n-1} f_{ik} - \sum_{i=1}^{n-1} x_{ik}, k = 2,\cdots,n-1 & (2.6) \\ \sum_{j=1}^{n} f_{1j} = n-1 & (2.7) \\ x_{ij} \in \{0,1\}, f_{ij} \ge 0, i,j = 1,2,\cdots,n & (2.8) \end{cases} \qquad (2)$$

The objective function minimizes the total length in the tour. Constraint (2.4) indicates that every node must be visited one time. The other constraints and all the variables have the same means as those in model (1). Solving this model by Lingo software, we can obtain the optimal tour of 27 cities in China, which is:1-7-6-5-3-8-20-21-22-23-25-24-19-18-17-16-13-14-9-12-11-10-4-27-26-2-1, the total length is 10428.3, the running time is 140 seconds.

To solve the VRP problem, we only need to revise constraint (2.2) in model (2). For example, if there are $R$ vehicles that can be used. We can formulate the VRP into the following integer linear programming model.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \quad (3.1)$$

$$s.t. \begin{cases} \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = R & (3.2) \\ \sum_{i=2}^{n} x_{i1} = \sum_{j=1}^{n-1} x_{nj} = 0 & (3.3) \\ \sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} = 1, k = 2,\cdots,n-1 & (3.4) \\ x_{ij} \le f_{ij} \le Mx_{ij}, e_{ij} \in E & (3.5) \\ \sum_{j=2}^{n} f_{kj} = \sum_{i=1}^{n-1} f_{ik} - \sum_{i=1}^{n-1} x_{ik}, k = 2,\cdots,n-1 & (3.6) \\ \sum_{j=1}^{n} f_{1j} = n-1 & (3.7) \\ x_{ij} \in \{0,1\}, f_{ij} \ge 0, i,j = 1,2,\cdots,n & (3.8) \end{cases} \qquad (3)$$

Let $R = 2$, we solve model (3) by Lingo software, obtain the two optimal route for 27 cities in China, which are: 1-2-1 and 1-26-27-4-10-11-12-9-14-13-15-16-17-18-19-24-25-23-22-21-20-8-3-5-6-7-1, the total length is 10557.9, the running time is 160 seconds.

Although models (1)(2)(3) are integer linear programming models. Since the number of 0-1 variables is not too many, so we can solve these models by Lingo software or other softwares. The running time is not too long for the small- or middle-size examples. But for examples of large size, efficient heuristic algorithms are still the best methods for finding the local optimal solutions. In the future, we will investigate different heuristic algorithms for solving the COP and related problems.

## Acknowledgements

This work is supported by National Natural Science Foundation of China under Grant No.10631070, No.60873205, No.10701080 and Beijing Natural Science Foundation under Grant No. 1092011. It is also partially supported by Foundation of Beijing Education Commission under Grant No. SM200910037005, the Funding Project for Academic Human Resources Development in Institutions of Higher Learning Under the Jurisdiction of Beijing Municipality (No.PHR201006217), and Funding Project for Base Construction of Scientific Research of Beijing Municipal Commission of Education (WYJD200902).

## References

[1] Chao I-M., Golden B.L.,Wasil E.A., A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88:475-489,1996.

[2] Wang X., Golden B.L.,Wasil E.A., Using a genetic algorithm to solve the generalization orienteering problem. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 263:273, Springer Science+Business Media,LLC 2008.

[3] Silberholz J., Golden B.L., The Effective Application of a New Approach to the Generalized Orienteering Problem. *MIC 2009: The VIII Metaheuristics International Conference*

[4] Golden B.L., Levy L.J., and Vohra R., The orienteering problem. *Naval Research Logistics*, 34(3): 307-318, 1987.

[5] Chekuri C. and Pal M., A Recursive Greedy Algorithm for Walks in Directed Graphs. *Proc. of IEEE FOCS*: 245-253, 2005.

[6] Frederickson G. and Wittman B., Approximation algorithms for the traveling repairman and speeding deliveryman problems. *Proc. of APPROX*, 119-133, 2007.

[7] Wang, Q., X. Sun, B. L. Golden, and Jia J., Using Artificial Neural Networks to Solve the Orienteering Problem, *Annals of Operations Research*, 61, 111- 120,1995.

[8] Wang Q., Sun X.,Golden B. L., Using Artifical Neural Networks to solve generalized orienteering problems. *Intelligent Engineering Systems Through Artificial Neural Networks*: Volume 6, ASME Press, New York,1063-1068,1996.

[9] Liang Y.C., Smith A.E., An ant colony approach to the orienteering problem. *Journal of the Chinese Institute of Industrial Engineers*, 23(5): 403-414, 2006.

[10] Gendreau M., Laporte G., and S*é*m*é*t F., A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2- 3):539-545, 1998.

[11] Tsiligirides T., Heuristics methods applied to orienteering. *Journal of the Operational Research Society*. 35(9):797-809, 1984.

[12] Chao I-M., Golden B.L.,Wasil E.A., The team orienteering problem, *European Journal of Operational Research*, 88: 464-474, 1996.

[13] Li Z., Zhang S., Zhang X.S., Chen L., Exploring the Constrained Maximum Edge-Weight Connected Graph Problem, *Acta Mathematicae Applicatae Sinica*, 25(4), 697-708, 2009.