| | |
|---|---|
| Publié par :<br>Published by :<br>Publicación de la : | Faculté des sciences de l'administration<br>Université Laval<br>Québec (Québec) Canada G1K 7P4<br>Tél. Ph. Tel. : (418) 656-3644<br>Fax : (418) 656-7047 |
| Édition électronique :<br>Electronic publishing :<br>Edición electrónica : | Aline Guimont<br>Vice-décanat - Recherche et partenariats<br>Faculté des sciences de l'administration |
| Disponible sur Internet :<br>Available on Internet<br>Disponible por Internet : | http ://www.fsa.ulaval.ca/rd<br>rd@fsa.ulaval.ca |

# DOCUMENT DE TRAVAIL 2003-020

FAST AND EFFICIENT HEURISTICS TO SOLVE TWO VERSION OF THE MEDIAN CYCLE PROBLEM

Jacques Renaud
Fayez F. Boctor
Gilbert Laporte

| | |
|---|---|
| Version originale :<br>Original manuscript :<br>Version original : | ISBN – 2-89524-170-8 |
| Série électronique mise à jour :<br>On-line publication updated :<br>Seria electrónica, puesta al dia | 05-2003 |

# Fast and Efficient Heuristics to Solve Two Version of the Median Cycle Problem

**Jacques Renaud[1,2], Fayez F. Boctor[1,2] and Gilbert Laporte[3]**
[1] Centre de Recherche sur les Technologies de l'Organisation Réseau, Université Laval, Québec, Canada G1K 7P4
[2]Faculté des sciences de l'administration, Université Laval, Québec, Canada G1K 7P4
[3] Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

**Abstract**

This article proposes a number of efficient heuristics for two versions of the *Median Cycle Problem* (MCP). In both versions the aim is to construct a simple cycle containing a subset of the vertices of a mixed graph. In the first version the objective is to minimize the cost of cycle and the cost of assigning vertices not on the cycle to the nearest vertex on the cycle. In the second version the objective is to minimize the cost of the cycle subject to an upper bound on the total assignment cost. Two heuristics are developed. The first, called the multistart greedy add heuristic, is composed of two main phases. In the first phase, a cycle composed of a limited number of randomly chosen vertices is constructed and augmented by iteratively adding the vertex yielding the largest cost reduction until either no further reduction is possible (for the first version) or the assignment cost is below the upper bound (for the second version). The second phase applies a number of improvement routines. The second heuristic is a random keys evolutionary algorithm. Computational results on a number of benchmark test instances show that the proposed heuristics are highly efficient for both versions of the problem, and superior to the only other available heuristic for these two versions of the problem.

Key words: Median cycle problem, greedy add heuristic, evolutionary algorithm

## Introduction

The purpose of this article is to describe heuristics for two versions of the *Median Cycle Problem* (MCP) defined as follows. Let $G = (V, E \cup A)$ be a complete mixed graph where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set, $E = \{[v_i,v_j]: v_i, v_j \in V, i<j\}$ is the edge set and $A = \{(v_i, v_j): v_i, v_j \in V\}$ is the arc set. The vertex $v_1$ is referred to as the *depot*, each edge is associated with a non-negative *routing cost* $c_{ij}$ and each arc is associated with a non-negative *assignment cost* $d_{ij}$. The first version of the median cycle problem, called MCP1, is to determine a Hamiltonian cycle through a subset

$V'$ of $V$ including $v_1$ that minimizes the sum of the routing cost of the cycle and the assignment cost of the vertices not on the cycle to their nearest vertex on the cycle. The routing cost $r(V')$ is the sum of all edge costs on the cycle, and the assignment cost $a(V')$ is computed as $\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} \{ d_{ij} \}$. The second version, called MCP2, is to determine a Hamiltonian cycle over $V'$ that minimizes $r(V')$, subject to $a(V') \leq d_0$.

Both versions of the MCP are solved by Moreno Pérez, Moreno-Vega and Rodríguez Martín[1] by means of a variable neighbourhood tabu search heuristic. Labbé, Laporte, Rodríguez Martín, and Salazar González[2,3] also provide integer linear programming formulations and develop a branch-an-cut algorithm for the MCP1 and MCP2. They use the appellation "*Ring Star Problem*" for MCP1.

Applications of MCP1 arise in the design of telecommunications networks in which user nodes are connected to concentrators lying on a backbone network linked to a root (depot)[4,5]. Both versions of the MCP have applications in the design of circular metro lines or motorways where the cost of a circular structure has to be weighted against its access costs. Another application is the location of post-boxes where both collection cost and user access time has to be considered[6].

The MCP is related to a number of *Cycle Problems* in which it is required to construct a cycle through a subset of vertices of a graph[7]. It can also be viewed as a *Location Routing Problem*[8]. In these problems, there may be constraints on the cycle length, or on the distance between the cycle and vertices not on it, or penalties for not visiting vertices, or profits for visiting them. Well known examples include the *Traveling Salesman Problem with Non-Visiting Penalties*[9], the *Selective Traveling Salesman Problem*[10,11], the *Prize Collecting Traveling Salesman Problem*[12] and the *Covering Tour Problem*[13]. Similar problems exist in context where a structure such as a path or a tree must be located through a subset of the vertices of a graph[14,15].

Our aim is to present two new heuristics for MCP1 and MCP2, both of which outperform tabu search. The first heuristic, called *Multistart Greedy Add* (MGA) is a multistart greedy construction heuristic followed by an improvement phase. The second, called *Random Keys Evolutionary Algorithm* (RKEA), is a mechanism combining and improving solutions produced by MGA. These heuristics will be presented in the next sections. This will be followed by computational results and by our conclusions.

## Basic procedures for the Multistart Greedy Add Heuristic

As is the case in a number of location-routing problems, simple construction and improvement heuristics tend to produce rather poor results on the MCP. This is due to the fact that removing a vertex from the cycle or introducing a new vertex has repercussions not only on the cycle itself, but also on the assignment cost. To obtain good results, it is necessary to produce a right blend of cycle reduction, cycle augmentation and cycle improvement operations. In what follows, we present five basic procedures used as building blocks for MGA, followed, in the next section, by the description by the MGA itself. The five basic procedures are called *cycle reduction*, *cycle augmentation*, *vertex exchange*, *cycle improvement* and *cycle perturbation*.

### *Cycle reduction*

Consider in turn each vertex $v_j \in V'$, where $V'$ is the set of vertices on the cycle, its predecessor $v_i$ and its successor $v_k$. Compute: 1) the saving $s_j = c_{ij} + c_{jk} - c_{ik}$, obtained by removing $v_j$ from the cycle and reconnecting $v_i$ with $v_k$, and, 2) a penalty $p_j$ obtained by reassigning $v_j$ and all vertices previously assigned to $v_j$ to their closest vertex on the cycle.

In MCP1, remove the vertex $v_j$ yielding $\max_{v_j \in V'} \{ s_j - p_j \}$ as long as this value is positive. In MCP2, remove the vertex $v_j$ yielding $\max_{v_j \in V'} \{ s_j \}$ as long as this value is positive and the updated assignment cost does not exceed $d_0$.

### *Cycle augmentation*

Consider in turn each vertex $v_j$ not on the cycle and compute: 1) the loss $s_j = c_{ij} + c_{jk} - c_{ik}$, and 2) the gain $p_j$ obtained by reassigning vertices not on the cycle to $v_j$ if this yields a smaller assignment cost (the assignment cost of $v_j$ is now zero).

In MCP1, insert the vertex $v_j$ yielding $\max_{v_j \in V \setminus V'} \{ p_j - s_j \}$ as long as this value is positive. In a feasible MCP2 solution, insert the vertex $v_j$ yielding $\min_{v_j \in V \setminus V'} \{ s_j \}$ as long as this value is negative (this can only happen if the routing cost matrix does not satisfy the triangle inequality). In an infeasible MCP2 solution, insert the vertex $v_j$ yielding $\min_{v_j \in V \setminus V'} \{ s_j \}$. In the latter case, this means that the routing cost increases can be accepted in order to reach feasibility.

### *Vertex exchange*

Consider each vertex pair $\{v_j, v_l\}$ where $v_j \in V'$ and $v_l \in V \setminus V'$. Remove $v_j$ from the cycle and insert $v_l$ in the cycle as described in the cycle reduction and augmentation procedures. In MCP1 implement the vertex exchange yielding the maximal total cost reduction as long as it is positive. In a feasible MCP2 solution implement the vertex exchange yielding the largest routing cost reduction as long as it is positive and the solution remains feasible. This procedure is never applied to an infeasible MCP2 solution.

### *Cycle improvement*

Attempt to improve the routing cost by means of a *Traveling Salesman Problem* (TSP) heuristic We have used both the $I^3$ heuristic[16] and the Lin-Kerninghan[17] implementation of Helsgaum[18].

### *Cycle perturbation*

The perturbation procedure repeats the following three steps for $\alpha = 0.10, 0.09, \ldots, 0$. The role of $\alpha$ is to generate several different perturbated solutions. For each value of $\alpha$, the procedure is applied to a feasible initial solution in which the cycle cost is equal to $z$.

1) Select the unrouted vertex $v_j \in V \setminus V'$ yielding the largest gain $p_j$. Insert this vertex in the cycle in the position yielding the minimal insertion cost as long as the routing cost does not exceed $(1+\alpha)z$. Repeat this step as long as this condition holds and the cycle contains fewer than $n$ vertices.

2) Apply the Cycle reduction procedure described above.

3) Apply the Cycle improvement procedure using $I^3$.

## Description of the Multistart Greedy Add Heuristic

For simplicity we first define two mega-procedures involving the above basic procedures. These are then used in the MGA heuristic.

### *Mega-procedure MP1*

Starting from a feasible solution, repeatedly apply Cycle reduction, Cycle augmentation, Vertex exchange and $I^3$ as long as the solution improves.

### *Mega-procedure MP2*

Starting from a feasible solution, repeatedly apply MP1 and Cycle perturbation as long as the solution improves.

### *Detailed Description of MGA*

The *Multistart Greedy Add* heuristic works with four parameters: $\beta_1 \in \{1, 3, 5\}$, $\beta_2 \in \{2, 3, 4\}$, $\beta_3 \in \{0.5, 1\}$ and $\beta_4 \in \{5, 10, 15, 20\}$. It applies the following five steps.

1) Generate $\beta_1 n$ solutions as follows. Create a first cycle containing $\beta_2$ randomly selected vertices (including the depot). Apply the Cycle augmentation procedure.

2) Retain the best $\beta_3 n$ solutions (with $\beta_3 \leq \beta_1$).

3) Apply MP1 to the best $\beta_4$% of these solutions where $\beta_4 \leq 20\%$.

4) Apply MP1 to another (25-$\beta_4$)% of the remaining solutions, randomly selected.

5) Apply MP2 and Lin-Kerninghan heuristic to the overall best solution.

## A Random Keys Evolutionary Algorithm

Since MGA produces a family of good solutions at the end of Step 4, it is natural to use these as the basis of an evolutionary algorithm in the hope of generating even better solutions. Our random keys evolutionary algorithm was constructed in this spirit. It works on the $\beta_3 n$ solutions generated at the end of Step 4 of the MGA heuristic. It uses the random keys encoding mechanism developed by Bean[19]. We first describe this scheme followed by the evolutionary algorithm itself.

## *The random keys encoding mechanism*

In our implementation of the random keys encoding mechanism, a random key $x_i \in [0,1]$ is assigned to each vertex $v_i$ on the cycle ($v_1$, …, $v_i$, …, $v_j$, …) where $x_1$=0. These keys are such that if $v_j$ comes after $v_i$ on a cycle, then $x_j > x_i$. For example, the cycle ($v_1$,$v_3$,$v_7$,$v_4$) could be assigned the keys (0, 0.12, 0.39, 0.46). For $n = 7$, this can also be represented as shown in Figure 1. In this figure, the sign "-"means that the corresponding vertex does not appear on the cycle.

| Vertex | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Key | 0 | - | 0.12 | 0.46 | - | - | 0.39 |

**Figure 1**: Random keys representation of a solution

Using these keys, applying a crossover operator to two parent solutions becomes easy. A crossover position is selected to combine the first part of the first parent with the second part of the second parent. Each vertex retains its key, which easily enables the reconstruction of the offspring (if two keys have the same value, the corresponding vertices are ordered arbitrarily). Figure 2 depicts a crossover of the two cycles ($v_1$,$v_3$,$v_7$,$v_4$) and ($v_1$,$v_2$,$v_5$,$v_4$,$v_7$,$v_3$) yielding a new cycle ($v_1$,$v_3$,$v_4$,$v_5$,$v_7$). The crossover position in this example is after vertex 4. The interest of this key encoding mechanism is that it easily allows for recombination of cycles of different lengths and also for large solution diversification. Thus a vertex near the end of one of the two parent solutions could move to an earlier position in the offspring solution. This process does not always yield a feasible offspring for MCP2. This may be the case if the offspring corresponds to a cycle containing very few vertices thus causing the total assignment cost of vertices not on the cycle to exceed the upper bound $d_0$. However, feasibility can be regained by introducing additional vertices on the cycle through the application of the Cycle augmentation procedure. We have also tested a variant of the crossover operator where two offspring are generated instead of only one but it turned out that generating only one offspring was better.

Parent 1

| Vertex | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Key | 0 | - | 0.12 | 0.46 | - | - | 0.39 |

Parent 2

| Vertex | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Key | 0 | 0.30 | 0.95 | 0.76 | 0.70 | - | 0.81 |

Offspring

| Vertex | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Key | 0 | - | 0.12 | 0.46 | 0.70 | - | 0.81 |

**Figure 2**: Crossover operation and the resulting offspring

## *Detailed description of the Random Keys Evolutionary Algorithm*

The proposed *Random Keys Evolutionary Algorithm* uses two parameters $\gamma_1$ and $\gamma_2$. The first parameter controls the number of successive iterations without improvement in the best known solution. The second parameter controls the proportion of solutions transferred from one population to another in Step 2 below. The algorithm works on a population *P* composed of the $\beta_3 n$ solutions generated at the end of Step 4 of MGA and transforms it into another population *P*' of the same size. The population *P*' is derived from *P* by following these five steps.

1) Set $P' = \varnothing$.

2) Move to *P*' the subset *P** of the $\gamma_2\beta_3 n$ best solutions of *P*, where $0 \leq \gamma_2 \leq 1$.

3) Generate $(1-\gamma_2)\ \beta_3 n$ new solutions to be included in *P*'. Each new solution is generated by randomly selecting two solutions from *P*, combining them using the crossover operator and applying MP1 to the offspring.

4) Starting with the worst offspring just created, replace it by a solution $P \backslash P^*$ if its cost is less than that of the offspring.

5) Set $P = P'$.

Repeat steps 1 to 5 as long as no improvement has been observed for $\gamma_1$ consecutive iterations. Then apply MP2 and Lin-Kernighan heuristic to the best solution of $P$.

## Computational Results

The *Multistart Greedy Add* heuristic and the *Random Keys Evolutionary Algorithm* were coded in Delphi 3.0 and run under Windows 2000 on an IBM Pentium III Desktop, 900 MHz. We first describe our test instances, followed by results obtained with the *Multistart Greedy Add* heuristic and the *Random Keys Evolutionary Algorithm*.

### *Test instances*

Both heuristics were tested on a subset of the instances used by Labbé *et al*[2,3] which are derived from the TSPLIB instances[20] with $51 \leq n \leq 200$ and distance matrices $l_{ij}$. To define the routing and assignment costs, we have proceeded as in Labbé *et al*[2,3]. For MCP1 instances, we have set $c_{ij} = \alpha_1 l_{ij}$ and $d_{ij} = (10\text{-}\alpha_1) l_{ij}$, where $\alpha_1 \in \{5,7,9\}$. For MCP2 instances, we have set $c_{ij} = d_{ij} = l_{ij}$. To determine $d_0$, we have first computed $l_0$, the cost of the shortest cycle with respect to the $l_{ij}$ distances including the depot and three other vertices. We have then set $d_0 = \alpha_2 l_0$, where $\alpha_2 \in \{0.08, 0.22, 0.42\}$. We have restricted our tests to the 52 instances of MCP1 and the 41 instances of MCP2 for which Labbé *et al*[2,3] have produced optimal solutions.

### *Multistart Greedy Add results and best parameters*

We have first conducted tests for the MGA heuristic over the 52 MCP1 and the 41 MCP2 instances, with several values of the parameters $\beta_1$, $\beta_2$, $\beta_3$ and $\beta_4$. We present in Tables 1 and 2 average results over all instances considered for each problem. The table headings are as follows:

| | |
|---|---|
| $\beta_1$: | multiplier used to generate a number of initial solutions ($\beta_1 n$ solutions are generated); |
| $\beta_2$: | number of vertices in the initial solution; |
| $\beta_3$: | multiplier used to select the solutions to be retained ($\beta_3 n$ solutions are selected); |
| $\beta_4$: | percentage of the best solution to which the mega-procedure MP1 is applied; |
| %: | percentage deviation of the heuristic solution value with respect to the optimum; |
| Seconds: | computation time in seconds. |

We first observe that irrespective of the parameter values, MGA always yields very accurate solutions within short computing times. For MCP1, irrespective of parameter values, MGA always yields average deviations not exceeding 0.45% of the optimum within 8 to 15 seconds of computing time. This can be compared with an average deviation of 0.69% and an average computing time of 75 seconds (for a tabu search heuristic coded in C++ and run on a Sun Ultra 60 computer running at 300 MHz) obtained by Moreno Pérez, Moreno-Vega and Rodríguez Martín[1]. For MCP2, our average deviation is always below 1.46% within 2 to 8 seconds, compared to an average deviation of 1.47% and an average running time of 32 seconds for Moreno Pérez, Moreno-Vega and Rodríguez Martín[1]. It should be stressed that our results are for a greedy procedure whereas Moreno Pérez, Moreno-Vega and Rodríguez Martín[1] use tabu search.

Our results also show that using $\beta_1$= 3 or 5 is much better than $\beta_1 = 1$. In other words, it pays to start with a larger pool of initial solutions. The algorithm seems much less sensitive to the other three parameters $\beta_2$, $\beta_3$ and $\beta_4$ although the best solutions were produced by using $\beta_1 = 5$, $\beta_3 = 1$ and $\beta_2 = 3$ or 4.

## *Random Keys Evolutionary Algorithm results and best parameters*

We first summarize, in Table 3, the average results obtained with RKEA for MCP1 and MCP2 on the modified TSPLIB instances. Two new headings are used in this table:

| | |
|---|---|
| $\gamma_1$: | number of successive iterations without improvement; |
| $\gamma_2$: | proportion of population best solutions directly moved to the new population. |

With respect to MGA, the computation times of RKEA are much higher (averages range from 53 to 161 seconds for MCP1 and from 85 to 437 seconds for MCP2) but accuracy is much improved (average deviation range from 0.16% to 0.27% for MCP1 and from 0.31% to 0.67% for MCP2). In contrast, minimum deviation over five runs obtained by Moreno Pérez, Moreno-Vega and Rodríguez Martín[1] are 0.25% and 0.56% for MCP1 and MCP2, respectively.

As expected a larger value of $\gamma_1$ yields better accuracy and larger computation times, but it is more difficult to assess the behaviour of the algorithm with respect to $\gamma_2$.

Finally, we present in Tables 4 and 5 individual results for each of MCP1 and MCP2 instances. We present average and minimum deviations as well as average computation times for the Moreno

Pérez, Moreno-Vega and Rodríguez Martín[1] *Variable Neighbourhood Tabu Search* (VNTS) heuristic and for RKEA ($\gamma_1 = 5$). It should be noted that the VNTS minimum and average results were obtained by running five times the same version of the heuristic (which contains random components), whereas the RKEA minimum and average results were obtained by successively running it with five different values of $\gamma_2$ (20, 25, 30, 35 and 40). A summary of the statistics of Tables 4 and 5 is presented in Table 6. It can be seen that RKEA is superior to VNTS both in terms of the average deviation from the optimum and in terms of the proportion of optimal solutions.

## Conclusions

We have presented two heuristics for two versions of the Median Cycle Problem, a combinatorial optimization problem arising, for example, in the design of telecommunications networks. The first of these heuristics, the Multistart Greedy Add (MGA) heuristic, quickly constructs a solution by means of a greedy mechanism. Despite its relative simplicity, MGA produces within very short computing times highly accurate solutions and compares well with a more elaborate Variable Neighbourhood Tabu Search heuristic developed by Moreno Pérez, Moreno-Vega and Rodríguez Martín[1]. The second heuristic (RKEA) applies evolutionary search to improve upon the solutions produced by MGA. Its accuracy level is much higher than that of the MGA but computation times increase as a result. However, these times remain reasonable given the fact that the MCP usually involves medium or long term decisions which do not call for instantaneous solutions.

***Acknowledgements***

This work was partially supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants OPG0039682, OPG0036509 and OPG0172633. This support is gratefully acknowledged. Thanks are also due to Keld Helsgaum who provided us with his Lin-Kernighan code and to Juan José Salazar González and Inmaculada Rodríguez Martín who provided their solutions of the TSPLIB instances. Finally, we thank two referees for their valuable comments.

**Table 1:** Multistart Greedy Add results for the 52 MCP1 test problems

| $\beta_1$ | $\beta_2$ | $\beta_4$ | $\beta_3 = 0.5$ | | $\beta_3 = 1$ | |
|---|---|---|---|---|---|---|
| | | | % | Seconds | % | Seconds |
| 1 | 2 | 5 | 0.41 | 5.4 | 0.39 | 9.3 |
| | | 10 | 0.43 | 5.4 | 0.38 | 9.1 |
| | | 15 | 0.43 | 5.3 | 0.40 | 8.9 |
| | | 20 | 0.41 | 5.1 | 0.45 | 8.5 |
| | 3 | 5 | 0.38 | 5.2 | 0.34 | 9.3 |
| | | 10 | 0.44 | 5.3 | 0.34 | 9.1 |
| | | 15 | 0.37 | 5.1 | 0.31 | 9.0 |
| | | 20 | 0.40 | 4.8 | 0.37 | 8.5 |
| | 4 | 5 | 0.42 | 5.4 | 0.37 | 9.6 |
| | | 10 | 0.39 | 5.1 | 0.37 | 9.4 |
| | | 15 | 0.37 | 5.1 | 0.39 | 9.1 |
| | | 20 | 0.38 | 4.9 | 0.38 | 8.5 |
| 3 | 2 | 5 | 0.37 | 8.0 | 0.38 | 11.8 |
| | | 10 | 0.39 | 7.8 | 0.39 | 11.6 |
| | | 15 | 0.38 | 7.7 | 0.35 | 10.9 |
| | | 20 | 0.41 | 7.4 | 0.39 | 10.6 |
| | 3 | 5 | 0.35 | 7.3 | 0.35 | 11.2 |
| | | 10 | 0.32 | 7.1 | 0.34 | 10.9 |
| | | 15 | 0.37 | 7.0 | 0.32 | 10.6 |
| | | 20 | 0.30 | 6.8 | 0.31 | 10.1 |
| | 4 | 5 | 0.35 | 7.0 | 0.30 | 11.1 |
| | | 10 | 0.34 | 6.9 | 0.30 | 10.7 |
| | | 15 | 0.36 | 6.7 | 0.29 | 10.5 |
| | | 20 | 0.32 | 6.6 | 0.30 | 10.2 |
| 5 | 2 | 5 | 0.39 | 10.2 | 0.35 | 14.0 |
| | | 10 | 0.36 | 10.2 | 0.35 | 13.7 |
| | | 15 | 0.36 | 10.0 | 0.34 | 13.5 |
| | | 20 | 0.37 | 9.9 | 0.33 | 12.9 |
| | 3 | 5 | 0.35 | 9.1 | 0.35 | 13.3 |
| | | 10 | 0.28 | 9.0 | 0.35 | 12.6 |
| | | 15 | 0.28 | 8.9 | 0.32 | 12.3 |
| | | 20 | 0.33 | 8.7 | 0.30 | 12.3 |
| | 4 | 5 | 0.31 | 8.6 | 0.24 | 12.5 |
| | | 10 | 0.30 | 8.4 | 0.29 | 12.3 |
| | | 15 | 0.34 | 8.4 | 0.29 | 11.9 |
| | | 20 | 0.30 | 8.2 | 0.30 | 11.8 |

**Table 2:** Multistart Greedy Add results for the 41 MCP2 test problems

| $\beta_1$ | $\beta_2$ | $\beta_4 = 0.5$ | $\beta_3 = 0.5$ |  | $\beta_3 = 1$ |  |
|---|---|---|---|---|---|---|
|  |  |  | % | Seconds | % | Seconds |
| 1 | 2 | 5 | 1.46 | 3.1 | 1.15 | 5.3 |
|  |  | 10 | 1.43 | 3.1 | 1.36 | 5.1 |
|  |  | 15 | 1.31 | 3.1 | 1.16 | 5.1 |
|  |  | 20 | 1.38 | 2.9 | 1.20 | 4.8 |
|  | 3 | 5 | 1.30 | 3.2 | 1.35 | 5.3 |
|  |  | 10 | 1.21 | 3.0 | 1.17 | 5.1 |
|  |  | 15 | 1.19 | 3.0 | 1.11 | 5.1 |
|  |  | 20 | 1.24 | 2.9 | 1.19 | 4.8 |
|  | 4 | 5 | 1.17 | 3.0 | 1.42 | 5.4 |
|  |  | 10 | 1.41 | 3.0 | 1.28 | 5.4 |
|  |  | 15 | 1.34 | 2.8 | 1.17 | 5.1 |
|  |  | 20 | 1.35 | 2.8 | 1.04 | 5.0 |
| 3 | 2 | 5 | 1.10 | 4.4 | 1.05 | 6.3 |
|  |  | 10 | 1.08 | 4.3 | 1.25 | 6.2 |
|  |  | 15 | 1.01 | 4.3 | 1.23 | 6.2 |
|  |  | 20 | 1.12 | 4.3 | 1.05 | 6.1 |
|  | 3 | 5 | 1.23 | 4.0 | 0.98 | 6.3 |
|  |  | 10 | 1.06 | 4.0 | 0.93 | 6.1 |
|  |  | 15 | 1.16 | 3.9 | 1.03 | 5.9 |
|  |  | 20 | 1.00 | 3.9 | 1.12 | 5.7 |
|  | 4 | 5 | 1.13 | 4.1 | 1.10 | 6.1 |
|  |  | 10 | 1.03 | 3.8 | 1.18 | 6.1 |
|  |  | 15 | 0.92 | 3.8 | 1.22 | 6.0 |
|  |  | 20 | 1.20 | 3.8 | 1.06 | 5.8 |
| 5 | 2 | 5 | 1.12 | 5.7 | 1.13 | 7.6 |
|  |  | 10 | 1.11 | 5.7 | 0.91 | 7.5 |
|  |  | 15 | 1.05 | 5.6 | 1.03 | 7.4 |
|  |  | 20 | 0.93 | 5.5 | 1.05 | 7.2 |
|  | 3 | 5 | 0.98 | 5.2 | 1.02 | 7.1 |
|  |  | 10 | 0.93 | 4.9 | 0.88 | 7.0 |
|  |  | 15 | 1.00 | 5.0 | 0.94 | 6.8 |
|  |  | 20 | 1.01 | 5.0 | 0.95 | 6.8 |
|  | 4 | 5 | 1.17 | 4.9 | 1.03 | 7.0 |
|  |  | 10 | 1.07 | 4.8 | 0.99 | 6.9 |
|  |  | 15 | 1.06 | 4.7 | 0.99 | 6.6 |
|  |  | 20 | 1.05 | 4.7 | 1.10 | 6.7 |

**Table 3:** RKEA results for MCP1 and MCP2 test problems

| $\gamma_1$ | $\gamma_2$ | MCP1 | | MCP2 | |
|---|---|---|---|---|---|
| | | % | Seconds | % | Seconds |
| 2 | 20 | 0.22 | 70 | 0.58 | 108 |
| | 25 | 0.27 | 65 | 0.49 | 157 |
| | 30 | 0.27 | 69 | 0.53 | 108 |
| | 35 | 0.25 | 60 | 0.58 | 105 |
| | 40 | 0.27 | 53 | 0.67 | 85 |
| 3 | 20 | 0.26 | 99 | 0.46 | 258 |
| | 25 | 0.20 | 101 | 0.51 | 214 |
| | 30 | 0.21 | 87 | 0.39 | 254 |
| | 35 | 0.24 | 82 | 0.51 | 152 |
| | 40 | 0.24 | 90 | 0.49 | 153 |
| 4 | 20 | 0.24 | 114 | 0.40 | 367 |
| | 25 | 0.21 | 120 | 0.46 | 268 |
| | 30 | 0.20 | 126 | 0.35 | 312 |
| | 35 | 0.19 | 104 | 0.46 | 174 |
| | 40 | 0.23 | 118 | 0.42 | 281 |
| 5 | 20 | 0.18 | 161 | 0.33 | 437 |
| | 25 | 0.22 | 123 | 0.36 | 333 |
| | 30 | 0.21 | 148 | 0.31 | 432 |
| | 35 | 0.16 | 155 | 0.40 | 245 |
| | 40 | 0.18 | 144 | 0.39 | 291 |

**Table 4:** Results for MCP1 instances

| | **VNTS** | | | **RKEA** | | | | **VNTS** | | | **RKEA** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Avg | Min | Seconds | Avg | Min | Seconds | Name | Avg | Min | Seconds | Avg | Min | Seconds |
| Eil51A-5 | 0.92 | 0.40 | 4.1 | 1.15 | 0.00 | 2.2 | KroD100-9 | 0.00 | 0.00 | 68.7 | 0.00 | 0.00 | 43.0 |
| Eil51A-7 | 0.00 | 0.00 | 5.9 | 0.00 | 0.00 | 2.2 | KroE100-5 | 0.59 | 0.41 | 37.4 | 0.31 | 0.23 | 33.8 |
| Eil51A-9 | 0.00 | 0.00 | 2.6 | 0.00 | 0.00 | 1.4 | KroE100-7 | 0.16 | 0.00 | 54.3 | 0.00 | 0.00 | 32.6 |
| ST70-5 | 0.12 | 0.00 | 11.2 | 0.00 | 0.00 | 10.6 | KroE100-9 | 0.00 | 0.00 | 26.9 | 0.00 | 0.00 | 37.8 |
| ST70-7 | 0.00 | 0.00 | 20.7 | 0.00 | 0.00 | 6.8 | Eil101-5 | 1.49 | 0.34 | 35.6 | 0.85 | 0.34 | 27.2 |
| ST70-9 | 0.00 | 0.00 | 23.6 | 0.00 | 0.00 | 7.6 | Eil101-7 | 0.64 | 0.00 | 33.5 | 0.00 | 0.00 | 44.6 |
| Eil76-5 | 0.99 | 0.20 | 14.5 | 0.08 | 0.00 | 12.4 | Eil101-9 | 0.00 | 0.00 | 49.5 | 0.00 | 0.00 | 34.4 |
| Eil76-7 | 0.00 | 0.00 | 21.5 | 0.00 | 0.00 | 9.4 | Pr107-5 | 0.13 | 0.00 | 45.3 | 0.09 | 0.00 | 41.8 |
| Eil76-9 | 0.00 | 0.00 | 20.8 | 0.00 | 0.00 | 9.2 | Pr107-7 | 0.00 | 0.00 | 43.6 | 0.01 | 0.00 | 98.4 |
| PR76-5 | 3.53 | 1.93 | 15.8 | 0.09 | 0.00 | 11.8 | Pr107-9 | 0.00 | 0.00 | 100.3 | 0.00 | 0.00 | 64.0 |
| PR76-7 | 0.29 | 0.00 | 13.5 | 0.00 | 0.00 | 19.0 | Pr136-5 | 1.51 | 0.04 | 108.6 | 0.37 | 0.19 | 110.0 |
| PR76-9 | 0.05 | 0.00 | 10.7 | 0.00 | 0.00 | 11.6 | Pr136-7 | 1.31 | 0.05 | 86.9 | 0.06 | 0.00 | 429.0 |
| Rat99-5 | 1.64 | 1.10 | 36.7 | 1.10 | 0.76 | 54.2 | Pr136-9 | 0.00 | 0.00 | 120.7 | 0.00 | 0.00 | 159.8 |
| Rat99-7 | 0.62 | 0.26 | 29.3 | 0.00 | 0.00 | 37.0 | Pr144-5 | 1.93 | 0.69 | 126.9 | 0.71 | 0.52 | 122.8 |
| Rat99-9 | 0.00 | 0.00 | 27.3 | 0.00 | 0.00 | 39.0 | Pr144-7 | 0.72 | 0.00 | 140.2 | 0.00 | 0.00 | 160.2 |
| KroA100-5 | 0.12 | 0.00 | 38.5 | 0.02 | 0.00 | 32.2 | Pr144-9 | 0.00 | 0.00 | 174.3 | 0.00 | 0.00 | 331.0 |
| KroA100-7 | 1.30 | 0.00 | 44.5 | 0.00 | 0.00 | 33.8 | KroA150-5 | 2.40 | 0.43 | 142.1 | 0.54 | 0.15 | 230.8 |
| KroA100-9 | 0.14 | 0.00 | 27.1 | 0.00 | 0.00 | 39.4 | KroA150-7 | 0.37 | 0.00 | 122.2 | 0.01 | 0.00 | 297.2 |
| KroB100-5 | 1.04 | 0.26 | 39.2 | 0.06 | 0.02 | 30.8 | KroA150-9 | 0.00 | 0.00 | 142.2 | 0.00 | 0.00 | 210.8 |
| KroB100-7 | 0.23 | 0.19 | 31.3 | 0.14 | 0.00 | 53.4 | KroB150-7 | 0.90 | 0.00 | 131.9 | 0.00 | 0.00 | 225.2 |
| KroB100-9 | 0.03 | 0.00 | 32.0 | 0.00 | 0.00 | 31.8 | KroB150-9 | 0.05 | 0.00 | 91.0 | 0.00 | 0.00 | 235.2 |
| KroC100-5 | 0.53 | 0.34 | 36.6 | 0.12 | 0.00 | 33.4 | PR152-9 | 0.49 | 0.00 | 116.9 | 0.00 | 0.00 | 436.8 |
| KroC100-7 | 0.00 | 0.00 | 43.3 | 0.00 | 0.00 | 28.1 | RAT195-5 | 3.29 | 2.39 | 301.6 | 2.60 | 2.25 | 549.4 |
| KroC100-9 | 0.00 | 0.00 | 85.3 | 0.00 | 0.00 | 36.2 | RAT195-7 | 0.63 | 0.11 | 237.7 | 0.39 | 0.32 | 1309.4 |
| KroD100-5 | 2.08 | 0.08 | 37.3 | 0.45 | 0.29 | 37.2 | KroB200-5 | 2.73 | 1.78 | 347.4 | 0.67 | 0.11 | 660.6 |
| KroD100-7 | 0.54 | 0.06 | 36.4 | 0.04 | 0.00 | 43.8 | KroB200-7 | 2.47 | 1.79 | 323.5 | 0.05 | 0.00 | 1044.2 |

**Table 5:** Results for MCP2 instances

| | **VNTS** | | | **RKEA** | | | | **VNTS** | | | **RKEA** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Avg | Min | Seconds | Avg | Min | Seconds | Name | Avg | Min | Seconds | Avg | Min | Seconds |
| Eil51A-0.08 | 2.25 | 0.87 | 4.7 | 1.79 | 0.87 | 6 | KroB100-0.42 | 0.24 | 0.00 | 24.8 | 0.12 | 0.00 | 272 |
| Eil51A-0.22 | 2.44 | 0.81 | 2.3 | 0.81 | 0.81 | 4 | KroC100-0.08 | 1.12 | 0.34 | 38.3 | 0.32 | 0.15 | 191 |
| Eil51A-0.42 | 0.12 | 0.00 | 2.8 | 0.23 | 0.00 | 5 | KroC100-0.22 | 0.25 | 0.00 | 29.4 | 0.00 | 0.00 | 199 |
| ST70-0.08 | 1.74 | 0.61 | 12.8 | 0.32 | 0.20 | 13 | KroC100-0.42 | 0.11 | 0.11 | 24.1 | 0.02 | 0.00 | 119 |
| ST70-0.22 | 0.59 | 0.00 | 8.6 | 0.17 | 0.00 | 20 | KroD100-0.08 | 1.74 | 1.45 | 41.1 | 0.19 | 0.07 | 216 |
| ST70-0.42 | 0.17 | 0.00 | 8.2 | 0.00 | 0.00 | 27 | KroD100-0.22 | 1.36 | 0.40 | 26.6 | 0.13 | 0.00 | 256 |
| Eil76-0.08 | 5.70 | 4.91 | 17.0 | 0.83 | 0.74 | 16 | KroD100-0.42 | 0.38 | 0.00 | 23.0 | 0.22 | 0.04 | 224 |
| Eil76-0.22 | 1.87 | 0.75 | 9.1 | 0.67 | 0.00 | 40 | KroE100-0.08 | 1.70 | 1.06 | 39.8 | 0.24 | 0.17 | 96 |
| Eil76-0.42 | 0.00 | 0.00 | 9.4 | 0.00 | 0.00 | 26 | KroE100-0.22 | 0.22 | 0.16 | 27.0 | 0.01 | 0.00 | 278 |
| PR76-0.08 | 5.39 | 1.06 | 15.2 | 0.24 | 0.02 | 19 | KroE100-0.42 | 0.10 | 0.00 | 22.6 | 0.00 | 0.00 | 170 |
| PR76-0.22 | 1.04 | 0.53 | 10.9 | 0.17 | 0.00 | 48 | Eil101-0.08 | 4.58 | 3.02 | 41.5 | 2.93 | 2.59 | 82 |
| PR76-0.42 | 0.29 | 0.01 | 9.4 | 0.01 | 0.00 | 35 | Eil101-0.22 | 1.53 | 1.05 | 22.3 | 0.91 | 0.70 | 158 |
| Rat99-0.08 | 1.71 | 1.21 | 36.9 | 0.92 | 0.66 | 72 | Eil101-0.42 | 0.60 | 0.60 | 20.4 | 0.24 | 0.00 | 178 |
| Rat99-0.22 | 0.27 | 0.00 | 28.5 | 0.16 | 0.00 | 113 | Pr107-0.22 | 0.13 | 0.00 | 29.5 | 0.00 | 0.00 | 161 |
| Rat99-0.42 | 0.29 | 0.00 | 21.0 | 0.24 | 0.00 | 114 | Pr107-0.42 | 0.15 | 0.00 | 16.9 | 0.00 | 0.00 | 130 |
| KroA100-0.08 | 2.39 | 0.87 | 39.9 | 0.02 | 0.00 | 157 | Pr136-0.22 | 3.86 | 0.46 | 72.8 | 0.33 | 0.22 | 652 |
| KroA100-0.22 | 4.30 | 0.03 | 23.8 | 0.09 | 0.02 | 243 | Pr136-0.42 | 0.29 | 0.25 | 49.6 | 0.19 | 0.00 | 456 |
| KroA100-0.42 | 0.47 | 0.00 | 20.9 | 0.00 | 0.00 | 207 | Pr144-0.22 | 0.63 | 0.19 | 78.4 | 0.45 | 0.33 | 1880 |
| KroB100-0.08 | 1.94 | 0.94 | 40.6 | 0.51 | 0.27 | 164 | KroA150-0.22 | 3.82 | 0.36 | 87.8 | 0.11 | 0.04 | 3429 |
| KroB100-0.22 | 2.50 | 0.14 | 25.4 | 0.11 | 0.02 | 317 | KroB150-0.08 | 1.70 | 0.73 | 150.3 | 0.73 | 0.37 | 1717 |
| | | | | | | | KroB150-0.22 | 0.36 | 0.12 | 91.3 | 0.15 | 0.02 | 1728 |

**Table 6:** Average computational results for 5 runs of VNTS and RKEA ($\gamma_1 = 5$)

| | MCP1 | | MCP2 | |
|---|---|---|---|---|
| | VNTS | RKEA | VNTS | RKEA |
| Minimum deviation from the optimum (%) | 0.25 | 0.09 | 0.56 | 0.19 |
| Average deviation (%) | 0.69 | 0.19 | 1.47 | 0.36 |
| Average (per run) computational time in seconds | 75.4 | 146 | 31.8 | 347 |
| Proportion of optimal solutions | 32/52 | 41/52 | 13/41 | 21/41 |

## References

1. Moreno Pérez J, Moreno-Vega M and Rodríguez Martín I (2002). Variable neighbourhood tabu search and its application to the median cycle problem, To appear in *European Journal of Operational Research,* 2004.
2. Labbé M, Laporte G, Rodríguez Martín I and Salazar González JJ (2001). The Ring star problem: Polyhedral analysis and exact algorithm, unpublished manuscript.
3. Labbé M, Laporte G, Rodríguez Martín I and Salazar González JJ (2001). The median cycle problem, unpublished manuscript.
4. Gourdin E, Labbé M and Yaman H (2002). Telecommunication and location, in *Facility location: Applications and theory*, Drezner, Z. and Hamacher H. (eds), Springer, Berlin, 275-305.
5. Klincewicz J (1998). Hub location in backbone/tributary network design: A review, *Location Science*, **6**, 307-335.
6. Labbé M and Laporte G (1986). Maximizing user convenience and postal service efficiency in post box location, *Belgian Journal of Operations Research, Statistics and Computer Science*, **26**, 21-35.
7. Bauer P (1997). The circuit polytope: Facets, *Mathematics of Operations Research*, **22**, 110-145.
8. Laporte G (1988). Location-routing problems. In *Vehicle Routing: Methods and Studies,* B. L. Golden and A. A. Assad (Editors). North-Holland, Amsterdam, 1988, 163-197.
9. Volgenant T and Jonker R (1987). On some generalizations of the traveling salesman problem, *Journal of the Operational Research Society,* **38**, 1073-1079.

10. Gendreau M, Laporte G and Semet F (1998). The selective traveling salesman problem, *Networks*, **32**, 263-273.
11. Fischetti M, Salazar González JJ and Toth P (1998). Solving the orienteering problem through branch-and-cut, *INFORMS Journal on Computing*, **10**, 133-148.
12. Balas E (1989). The prize collecting traveling salesman problem, *Networks*, **19**, 621-636.
13. Gendreau M, Laporte G and Semet F (1997). The covering tour problem, *Operations Research*, **45**, 568-576.
14. Labbé M, Laporte G and Rodríguez Martín I (1998). Path, tree and cycle location, in *Fleet Management and Logistics*, T. Crainic and G. Laporte (edts), Kluwer, Boston, 187-204.
15. Feremans C, Labbé M and Laporte G (2003). Generalized network design problems, To appear in *European Journal of Operational Research*.
16. Renaud J, Boctor F and Laporte G (1996). Fast composite heuristic for the symmetric traveling salesman problem, *ORSA Journal on Computing*, **3**, 134-143.
17. Lin S and Kernighan BW (1973). An effective heuristic algorithm for the traveling salesman problem, *Operations Research*, **20**, 498-516.
18. Helsgaum K (2000). Effective implementation of the Lin-Kerninghan traveling salesman heuristic, *European Journal of Operational Research*, **126**, 106-130.
19. Bean J (1994). Genetic algorithms and random keys for sequencing and optimization, *ORSA Journal on Computing*, **6**, 154-160.
20. Reinelt G (1991). TSPLIB – A Traveling salesman problem library, *ORSA Journal on Computing*, **3**, 376-384.