An Enhancement of Daskin’s Algorithm for Solving $P$-centre Problem*

Al-khedhairi A.†
Department of Statistics and OR, College of Science, King Saud University,
P.O. Box 2455, Riyadh: 11451, Saudi Arabia.
February 21, 2007

Abstract

An Enhancement to the exact algorithm of Daskin (1995) to solve the vertex $P$-center problem is proposed. A simple enhancement which uses tighter initial lower and upper bounds, and a more appropriate binary search method are introduced to reduce the number of subproblems to be solved. These ideas are tested on a well known set of problems from the literature (i.e., TSP-Lib problems) with encouraging results.

Key words: $P$-center, location, 0-1 Programming.

1 Introduction

The $P$-center problem which was also introduced first by Hakimi (1964, 1965), is to find the facility locations such that the maximum distance between any demand point (customer) and its respective nearest facility is minimised.

It has been used to model locations of emergency facilities such as ambulance stations and firehouses, the location of a helicopter to minimise the maximum time to respond

---

*This project is supported by College of Science - Research centre project No (Stat/2007/21).
†Email: akhediri@ksu.edu.sa
to an emergency, and the location of a transmitter to maximise the lowest signal level received in a communication network (Caruso et al., 2003).

There are several possible variations of the basic model. If facility locations are restricted to the nodes of the network, the problem is referred to as a *vertex center problem*. Center problems which allow facilities to be located anywhere on the network are known as *absolute center problems*. Both versions can be either *weighted* or *unweighted*. In the weighted problem, the distances between demand nodes and facilities are multiplied by a weight usually associated with the demand node. In the unweighted problem, all demand nodes are treated equally. For more information on the $P$-center problem and the associated solution techniques the reader is referred to Handler (1990), and Daskin (1995).

To formulate the vertex $P$-center problem, we define:

$I = \text{set of demand nodes, } I = \{1, \ldots, N\}$.

$J = \text{set of candidate facility sites, } J = \{1, \ldots, M\}$.

$d_{ij} = \text{distance between demand node } i \in I \text{ and candidate site } j \in J$.

$P = \text{number of facilities to be located,}$

$w_j = \begin{cases} 
1 & \text{if a facility is located at candidate site } j \in J. \\
0 & \text{otherwise.}
\end{cases}$

$Y_{ij} = \begin{cases} 
1 & \text{if demand node } i \in I \text{ is assigned to an open facility at candidate site } j \in J. \\
0 & \text{otherwise.}
\end{cases}$

$D = \text{maximum distance (or time) between a demand node and the nearest facility (D is also referred to as the covering distance or time)}.$

The binary linear programming formulation of the vertex $P$-center problem is as follows:

\[
\text{Minimise} \quad D \quad \quad (1)
\]

subject to:

\[
\text{2}
\]
\[ \sum_{j \in J} Y_{ij} = 1 \quad \forall i \in I \]  
(2)

\[ Y_{ij} \leq w_j \quad \forall i \in I, \ j \in J \]  
(3)

\[ \sum_{j \in J} w_j = P \]  
(4)

\[ D \geq \sum_{j \in J} d_{ij} Y_{ij} \quad \forall i \in I \]  
(5)

\[ w_j, Y_{ij} \in \{0, 1\} \quad \forall i \in I, \ j \in J \]  
(6)

The objective function (1) minimises the maximum distance between each demand node and its closest open facility. Constraint (2) ensures that each demand node is assigned to exactly one facility, while constraints (3) restrict demand nodes to be assigned to open facilities. Constraint (4) stipulates that \( P \) facilities are to be located. Constraints (5) define the maximum distance between any demand node \( i \) and the nearest facility at node \( j \). Finally, constraints (6) refer to integrality constraints.

For fixed values of \( P \), the vertex \( P \)-center problem can be solved in polynomial time. This can be done by evaluating each of the \( O(N^P) \) possible combinations of \( P \) facility sites. Evaluating each of these can be done in polynomial time (Daskin, 1995) though it may take a considerable amount of CPU time. For variable values of \( P \), the \( P \)-center problem is NP-hard (Kariv and Hakimi, 1979).

Different authors have used an auxiliary problem (e.g., the Set Covering Problem; SCP) to solve the \( P \)-center problem optimally. The objective of the SCP is to find the minimum number of facilities and their locations so that each demand point has to be served by a facility within a specified maximum response time (or distance) which can be referred to as radius. The solution to this problem can be easily found by solving its linear programming relaxation with occasional branch and bound applications. However, for large problems the size of the relaxed version of the SCP can be reduced using successive row and column reductions (see Daskin, 1995 for more discussion of such reduction rules). This idea is to find the smallest radius such that the optimal solution of the considered
auxiliary problem yields a feasible solution to the $P$-center problem. Initially, Minieka (1970) suggested a rudimentary algorithm that relies on solving a finite sequence of Set Covering Problems. The idea is to choose a threshold distance (covering distance) as radius and to check whether all demand points (customers) are covered within this radius using no more than $P$ facilities. Based on Minieka’s idea, Daskin (1995) developed an algorithm to solve this problem optimally using the bisection method that systematically reduces the gap between the upper and lower bounds of the optimal solution and hence finds the optimal covering distance. Also, Daskin (2000) and Elloumi, Labbe and Pochet (2001) proposed two efficient and exact algorithms for the vertex $P$-center problem. Both algorithms solve successive subproblems and rely on carrying out an iterative search over coverage distances rather than sets of facility locations. Daskin has formulated a maximum set covering subproblem and solved it by Lagrangian relaxation, whereas Elloumi et al. used Minieka’s subproblems and solved it by a greedy heuristic and the IP formulation of the subproblem. Ilhan and Pinar (2001) proposed an interesting exact solution method for the vertex $P$-center problem. Their method is composed of two phases. The first phase, called the LP-Phase, computes a lower bound to the optimal solution of the problem by solving a series of feasibility problems based on a LP formulation. The second phase, referred to as the IP-Phase, uses also feasibility problems to check whether or not it is possible to serve all customers with no more than $P$ facilities within a given radius.

In this study we introduce some modifications to the algorithms Daskin (1995). As this algorithm is used as a basis in this work, its respective formulation will be given in the next section.

The remainder of this paper is organized as follows. In section 2 the algorithm of Daskin (1995) is given followed by our enhancement scheme. Our computational results for the modification is provided in section 3 and section 4 summarises our findings and points out some research issues.

2 Daskin (1995) Algorithm

Based on Minieka’s (1970) algorithm, Daskin proposed an improved algorithm to find the optimal solution of the vertex $P$-center problem. This algorithm can be described as
follows:
- Select initial lower and upper bounds on the value of the \( P \)-center objective function.
- Solve the Set Covering Problem (SCP) using the average (\( D \)) of lower and upper bounds as the coverage distance (rounded down) and let \( k \) be the number of facilities found to cover all nodes (or customers). If \( k \leq P \), reset the value of the upper bound to \( D \), else (\( k > P \)) reset the lower bound to \( D + 1 \).
- If the lower and upper bounds are equal then the lower bound (or the upper bound) is the optimal solution to the \( P \)-center problem and stop; otherwise solve the SCP with the new coverage distance (\( D \)) set to the average of the lower and upper bounds (rounded down), and continue the process.

The main steps of this algorithm are given in Figure 1.

<table>
<thead>
<tr>
<th>Step 0:</th>
<th>Set ( L = 0 ) (zero) and ( U = \max_{i,j} (d_{ij}) ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1:</td>
<td>Calculate ( D = \left\lfloor \frac{L + U}{2} \right\rfloor )</td>
</tr>
<tr>
<td>Step 2:</td>
<td>Solve SCP for the coverage distance ( D ), and let ( k ) be the number of facilities found.</td>
</tr>
<tr>
<td>(i)</td>
<td>If SCP is feasible ( (i.e. \ k \leq P) ) then set ( U = D ).</td>
</tr>
<tr>
<td>(ii)</td>
<td>Else ( (i.e. \ SCP ) is infeasible, ( k &gt; P )) set ( L = D + 1 ).</td>
</tr>
<tr>
<td>Step 3:</td>
<td>If ( L = U ), then the optimal solution is ( L ), and stop.</td>
</tr>
<tr>
<td></td>
<td>Else go to Step 1.</td>
</tr>
</tbody>
</table>

Figure 1: The original algorithm of Daskin (1995).

This approach is based on searching over the range of coverage distances for the smallest coverage distance that allows all demand nodes to be covered. This search procedure is usually referred to as the \textit{binary search}.

One way to reduce the number of iterations needed to find the optimal solution of the \( P \)-center problem using this algorithm could be done by designing a more powerful binary search. Our modification is based on this observation.

\textbf{Modified Scheme of Daskin’s (1995) Algorithm}

To reduce the number of iterations needed to find the optimal solution, a scheme for speeding up the shrinkage of the gap between the lower and the upper bounds are proposed using scheme based on computing the coverage distance (\( D \)) more efficiently.
At this research we will use a different concept in reducing the gap between the lower and the upper bounds to find the optimal solution. In the original algorithm, if the difference between the lower \((L)\) and the upper \((U)\) bounds is large, the use of the average of \(L\) and \(U\) as coverage distance for a SCP may lead to extra iterations which may not be needed. This happens especially at the first iterations and if the optimal solution is closer to one end of the range. In this variant, we direct the search using the proposed scheme. This can be done by splitting the distance values (or customers) into three parts.

(i) First, we find two points, say \(X\) and \(Y\), in the distances range \([\text{min}, \text{max}]\) (see Figure 2). The values of \(X\) and \(Y\) are as follows: \(X=\text{min} + \alpha(\text{max} - \text{min})\) and \(Y=\text{min} + (1 - \alpha)(\text{max} - \text{min})\) where \(\text{min}\) and \(\text{max}\) are the minimum and maximum distance values in the distance matrix. In our experiments we used \(\alpha = \frac{1}{4}\) (see \(X\) and \(Y\) in Figure 2).

(ii) Second, we set the coverage distance \((D)\) to the value of \(X\) and solve a SCP at this coverage distance \((D)\). If the solution is feasible then the optimal solution occurs between \(\text{min}\) and \(X\) (i.e. \(L = \text{min}\) and \(U = X\)); otherwise, we reset the coverage distance \((D)\) to the value of \(Y\) and solve a SCP. If the solution is feasible then the optimal solution occurs between \(X\) and \(Y\) (i.e. \(L = X\) and \(U = Y\)); otherwise the optimal solution occurs between \(Y\) and \(\text{max}\) (i.e. \(L = Y\) and \(U = \text{max}\)). If the optimal solution occurs between \(\text{min}\) and \(X\) then we find again the upper limit of the \(\frac{1}{4}\) of the distances range at this interval \(([\text{min}, X])\) and reset it to the coverage distance \((D)\) and solve a SCP. If it is feasible then we record the lower and upper limits of the interval and repeat this step until we get an
infeasible solution to the SCP.

(iii) Third, during the testing in both of the data sets we observed that to reach the optimal solution fast we may need to use a correction factor to increase the coverage distance \((D)\) rapidly if the SCP using \(D\) is infeasible and to decrease it sharply if the SCP using \(D\) is feasible. Then at a certain stage, when the solution contains \(p + 1\) facilities, we use a bisection method and SCP to find this optimal solution. This process usually needs between one and three iterations to reach the optimal solution. By conducting some experiments using both data sets we found that this correction factor can be expressed as follows:

\[
\delta = \frac{\alpha}{\beta} = \frac{P_1 - p}{P_1 - P_2}
\]

where \(P_1\) is the objective function value of SCP (i.e. the number of facilities needed to cover all demand nodes) at the lower bound of the interval where the optimal solution occurred, \(P_2\) is the objective function value of SCP at the upper bound of the interval, and \(p\) is the number of facilities (or centres) required \((0 < \delta < 1)\).

Observations:

(1) It can be noticed that the correction factor will be equal to 1 when \(P_2 = p\) and hence the coverage distance will be unchanged and hence the scheme will cycle.

To avoid this weakness we reset \(\alpha = \begin{cases} P_1 - (p + 1) & \text{if } P_2 = p \\ P_1 - p & \text{i.e. as before} \end{cases}\) otherwise

(2) At the beginning we set the coverage distance, in step 3 of our algorithm, (rounded down) to \(D_3 = \lfloor L + (U - L) \times \delta \rfloor\). If the problem using the coverage distance \((D_3)\) produces \(p + 1\) facilities, we use the bisection method and SCP to find this optimal solution quickly. After performing some experiments we noticed that if the correction factor, \(\delta = \frac{\alpha}{\beta}\), happens to be close to 1, then the coverage distance \((D_3)\) will decrease by a relatively negligible amount. Similarly if \(\delta\) is near zero, the value of \(D_3\) will increase by a very small amount until it reaches the target value of \(D_3\) when \(P_1 = p + 1\). So too many redundant iterations are needed before a change happens. To update the coverage distance \((D_3)\) in a more efficient manner (i.e. it will decrease by a reasonable amount in case \(\delta\) is near 1 and increase by a reasonable amount in case \(\delta\) is near zero), we introduced
two correction factors ($\theta_1, \theta_2$, where $\theta_1 + \theta_2 = 1$). Though we added two parameters to the search it is only one value as $\theta_2 = 1 - \theta_1$. Inspired by the results based on the values of $\delta$ during the testing of both data sets, we experimented initially with three values for $\theta_1$ ($\theta_1 = 0.05, 0.10$ and $0.25$). The new formula of the coverage distance $D_3$ (rounded down) is then written as follows:

$$
D_3 = \begin{cases} 
|L + (U - L) \times \theta_2| & \text{if } \delta > \theta_2 \\
|L + (U - L) \times \delta| & \text{if } \theta_1 \leq \delta \leq \theta_2 \\
|L + (U - L) \times \theta_1| & \text{if } \delta < \theta_1
\end{cases}
$$

This new formula will guarantee that $D_3$ will increase (or decrease) by a reasonable value larger than in the previous variant if $\delta$ close to zero (or one) and will stay as before if $\delta$ lies in the range $(\theta_1, \theta_2)$. Note that we kept both $\theta_1$ and $\theta_2$ in the formulation for completeness in case there is no relationship between $\theta_1$ and $\theta_2$.

(3)- At the original algorithm we set $L = D + 1$ if the SCP is infeasible, here in our proposed algorithm we always reset $L$ to the minimum distance greater than $D$. This jump to the next customer is used as there may not be a customer at distance $D + 1$. Also, this update can minimise the gap between lower and upper bound faster especially if this gap happens to be large due to the sparsity of the values in the distance matrix. Such a modification will also guarantee that we reset the lower bound to an existing customer site.

In addition, once $P_1 = p + 1$ we resort back to the use of the bisection method and SCP where the new coverage distance (rounded down) is as before, $D = \left\lfloor \frac{U + L}{2} \right\rfloor$.

Initially we set the values of $P_1$ and $P_2$ to $N$ (number of demand nodes) and 1, respectively.

The steps of our proposed algorithm are given in Figure 3.

3 Computational Results

The original and the modified algorithm of Daskin (1995) are coded in C, and tested on a Sun Enterprise Workstation 450 running Solaris 2.6. We use Xpress-MP Optimisation
Step 0: (i) Set initial value of $P_1 = N$ & $P_2 = 1$.
   
   Set $L =$ Minimum distance $(d_{ij})$ in distance matrix,
   
   $U =$ Maximum distance $(d_{ij})$ in distance matrix.
   
   (ii) Set $D = \left\lceil L + \frac{U-L}{4} \right\rceil$.

Step 1: Solve a SCP for the coverage distance $D$.

   (A) If feasible then set $P_2 =$ the objective of the SCP, and $U = D$.
   
   (i) Set $D_1 = \left\lceil L + \frac{U-L}{4} \right\rceil$ and solve a SCP for the coverage distance $D_1$.
      
      - If feasible then set $P_2 =$ the objective of the SCP, and $U = D_1$.
         
         Go to Step 1(A) (i).
      
      - Else set $P_1 =$ the objective of the SCP and $L = D_1$.
   
   (B) Else set $P_1 =$ the objective of the SCP and $L = D$.
   
   (i) Set $D_2 = \left\lceil L + \frac{3(U-L)}{4} \right\rceil$ and solve a SCP for the coverage distance $D_2$.
      
      - If feasible then set $P_2 =$ the objective of the SCP, and $U = D_2$.
      
      - Else set $P_1 =$ the objective of the SCP and $L = D_2$.

Step 2: (i) If $P_2 = p$ then set $\alpha = P_1 - (p+1)$, and $\beta = P_1 - P_2$.

   - Else set $P_1 = P_1 - p$ and $\beta = P_1 - P_2$.

   (ii) If $P_1 = p + 1$ then go to Step 4.

Step 3: (i) Set $\theta_1 = 0.25$, $\theta_2 = 1 - \theta_1$.

   - Calculate $\delta = \frac{\alpha}{\beta}$, $0 < \delta < 1$.

   (ii) Set $D_3 = \left\lbrack \begin{array}{ll}
   L + (U - L) \cdot \theta_2 & \text{if } \delta > \theta_2 \\
   L + (U - L) \cdot \delta & \text{if } \theta_1 \leq \delta \leq \theta_2 \\
   L + (U - L) \cdot \theta_1 & \text{if } \delta < \theta_1
   \end{array} \right\rbrack$

   and solve a SCP for the coverage distance $D_3$.

   - If the solution is feasible then set $P_2 =$ the objective of the SCP and $U = D_3$.

   - Else set $P_1 =$ the objective of the SCP and,

   
   $L = \min \{ d_{ij} : d_{ij} > D_3, \forall i \in I, j \in J \}$.

   (iii) If $L = U$ then stop. The optimal solution is $L$. Else go to Step 2(i).

Step 4: (i) Set $D_4 = \left\lceil \frac{U + L}{2} \right\rceil$ and solve a SCP for the coverage distance $D_4$.

   - If the solution is feasible then set $U = D_4$.

   - Else set $L = \min \{ d_{ij} : d_{ij} > D_4, \forall i \in I, j \in J \}$.

   (ii) If $L = U$ then stop. The optimal solution is $L$. Else go to Step 4(i).

Figure 3: The proposed algorithm for solving $P$-center problem.
software (Modeller Release 12.06, Optimiser Release 12.50) to solve the corresponding LP and IP problems. The data sets were taken from the TSP-LIB (Reinelt, 1991) for the Travelling Salesman Problems which we refer to as TSP instances.

In the TSP-Lib test problems, the set of facilities is equal to the set of demand nodes or customers \( (M = N) \). The coordinates of the nodes are given, and hence the shortest path distances are computed as the Euclidean distances between every pair of nodes. The entries of the distance matrix \( (d_{ij}) \) are nonintegral, but as the original algorithms and their modified versions apply only to integral data, we rounded down every distance \( (d_{ij}) \) as used in the literature. The size of the instances range from 226 to 657 nodes. For each instance, four \( P \) values are used namely \( P=5, 10, 20, \) and 40.

Table 1 reports the number of iterations required to find the optimal solution of the original and the modified algorithm of Daskin (1995) for the TSP data sets. Column 1 gives the file name, columns 2 and 3 refer to the number of demand nodes (or customers) and the number of facilities (centers) required to locate respectively. Column 4 represents the number of iterations performed by the original algorithm to find the optimal solution. Columns 5, 6 and 7 refer to the number of iterations needed by the modified algorithm of Daskin (1995) to get the optimal solution for \( \theta_1 = 0.05, 0.10 \) and 0.25, respectively. The bold numbers in the table mean that the modified algorithm of Daskin (1995) is performed better than the original algorithm. Note that “?” in data file \( pr264 \) \( (p=40) \), in Table 1, means that the original algorithm, and its modified algorithm could not find the optimal solution after 48 hours of running time. The number with “?” is the last iteration performed before stopping.

According to the results in Table 1, we can conclude that over the three values of the correction factor \( \theta_1 \) (resp. \( \theta_2 \)), \( \theta_1 = 0.25 \) (and \( \theta_2 = 0.75 \)) yielded better results than the other two values. In addition, the modified algorithm, when \( \theta_1 = 0.25 \), solved 32 TSP instances in fewer iterations than the original algorithm, whereas at \( \theta_1 = 0.05 \) and \( \theta_1 = 0.10 \), only 21 and 28 instances are solved in fewer iterations, respectively. So, when \( \theta_1 = 0.25 \) the average number of iterations needed to solve the problem optimally is reduced from 12.12 iterations in the original algorithm to 10.40 iterations in the modified algorithm, whereas when \( \theta_1 = 0.05 \) and 0.10 the average number of iterations decreased from 12.12 iterations in the original algorithm to 11.58 and 10.93 iterations respectively. Moreover,
just 4 of the TSP instances are solved in one iteration more than the original algorithm while 13 and 11 of the TSP instances required more iterations when using $\theta_1 = 0.05$ and $\theta_1 = 0.10$ (5 and 3 iterations more than the original algorithm), respectively. Also, a reduction of 5 iterations is obtained in 3 instances.

4 Conclusion

In this paper we show how simple and easy to implement enhancements can improve results of some existing algorithms. In this work, a modification put forward to speed up the process of shrinking the gap between the lower ($L$) and upper ($U$) bounds in order to solve the vertex $p$-centre problem. So, we introduced a correction factor to the coverage distance ($D$) to accelerate the procedure of finding the optimal solution adaptively. This is done either by enlarging the coverage distance ($D$) to a reasonable value if the SCP at $D$ is infeasible or by decreasing it to a realistic value if the SCP at $D$ is feasible. In addition, when the coverage distance becomes close to the optimal solution based on the criterion that $p + 1$ facilities are found, we refer back to the bisection method. A feature which assigns relatively smaller weights to the extreme points of the range is also added ($\theta_1$ and $\theta_2$) and this proved promising.

As for future work the author is investigating the use of a ‘good’ heuristic to produce tighter upper bound, which can then be used as a measure to define a corresponding tighter lower bound. Such upper and lower bounds will obviously help in speeding up the process of finding the optimal solution even further. Also, clearly from the results achieved so far, our attempt in introducing the correction factor ($\delta$) to speed up the process of minimising the gap between lower and upper bounds showed encouraging results. A dynamic adjustment of $\delta$ that uses the quality of the solution during the search may be worth exploring.

Acknowledgement- The author is grateful to both referees for their constructive comments. I would also like to thank the college science research centre for supporting this project.
<table>
<thead>
<tr>
<th>File Name</th>
<th>N</th>
<th>P</th>
<th>Daskin Algorithm (# Iters.)</th>
<th>Modified Algorithm (# Iters.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\theta_1 = 0.05$</td>
</tr>
<tr>
<td>pr226</td>
<td>226</td>
<td>40</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>pr226</td>
<td>226</td>
<td>20</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>pr226</td>
<td>226</td>
<td>10</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>pr226</td>
<td>226</td>
<td>5</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>pr264</td>
<td>264</td>
<td>40</td>
<td>37</td>
<td>5?</td>
</tr>
<tr>
<td>pr264</td>
<td>264</td>
<td>20</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>pr264</td>
<td>264</td>
<td>10</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>pr264</td>
<td>264</td>
<td>5</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>pr299</td>
<td>299</td>
<td>40</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>pr299</td>
<td>299</td>
<td>20</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>pr299</td>
<td>299</td>
<td>10</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>pr299</td>
<td>299</td>
<td>5</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>pr439</td>
<td>439</td>
<td>40</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>pr439</td>
<td>439</td>
<td>20</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>pr439</td>
<td>439</td>
<td>10</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>pr439</td>
<td>439</td>
<td>5</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>40</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>20</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>10</td>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>5</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>kroA200</td>
<td>200</td>
<td>40</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>kroA200</td>
<td>200</td>
<td>20</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>kroA200</td>
<td>200</td>
<td>10</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>kroA200</td>
<td>200</td>
<td>5</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>kroB200</td>
<td>200</td>
<td>40</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>kroB200</td>
<td>200</td>
<td>20</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>kroB200</td>
<td>200</td>
<td>10</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>kroB200</td>
<td>200</td>
<td>5</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>lin318</td>
<td>318</td>
<td>40</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>lin318</td>
<td>318</td>
<td>20</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>lin318</td>
<td>318</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>lin318</td>
<td>318</td>
<td>5</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>gr202</td>
<td>202</td>
<td>40</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>gr202</td>
<td>202</td>
<td>20</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>gr202</td>
<td>202</td>
<td>10</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>gr202</td>
<td>202</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>d493</td>
<td>493</td>
<td>40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>d493</td>
<td>493</td>
<td>20</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>d493</td>
<td>493</td>
<td>10</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>d493</td>
<td>493</td>
<td>5</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>d657</td>
<td>657</td>
<td>40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>d657</td>
<td>657</td>
<td>20</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>d657</td>
<td>657</td>
<td>10</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>d657</td>
<td>657</td>
<td>5</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>521</td>
<td>498</td>
</tr>
</tbody>
</table>

Table 1: Comparison results of the original algorithm of Daskin and its modified algorithm on TSP instances for $\theta_1=0.05$, 0.10 and 0.25.
References


