A Data Structure to Significantly Speed up A Simulated Annealing Heuristic for Solving the Dynamic Facility Layout Problem

I-Ming Chao*, Nai-Chien Wei, Tai-Lioan Chen, and Pei-Tsang Wu
Department of Industrial Management, I-Shou University
No.1, Sec. 1, Syuecheng Rd., Dashu District, Kaohsiung City, Taiwan
*iming@csu.edu.tw

Abstract

In this paper a data structure is designed to significantly speed up a simulated annealing heuristic for solving the dynamic facility layout problem (DFLP) existed in literature. The DFLP augments the static facility layout problem (SFLP) to take the flexibility of layout planning into account by considering that the material handling flows between each pair of departments may vary over time periods. Unlike that in the SFLP, the DFLP might require rearranging facility layout over periods to trade off costs between material handling flows and rearranging departments. By companioned the existing simulated annealing heuristic for the DFLP with our designed data structure, the revised version of the algorithm can significantly speeded up the computing time for solving problems, and the data structure can be applied as a useful tool to develop some more efficient solution methods for solving the DFLP related optimization problems in future.

Keywords: Data structure, Dynamic facility layout problem, simulated annealing

1. Introduction

In today’s volatile competing environment, facility layout plays an important role for enterprises to create and maintain their substantial competitive advantage, since it accounts for one of the most important portions of an enterprise’s total indirect costs that often share the most of the enterprise’s total operating costs. The facility layout problem has classically been treated statically, in the classical static facility layout problem (SFLP) facilities are divided into a group of divisions called departments, a set of candidate locations are preserved for locating these departments, and quantities of material flows between pairs of departments, distances of pairs of candidate locations, and costs of per unit of flow-distance are calculated by assuming that material flows between pairs of departments to be constant over planning time period. The SFLP is to assign a location for each department to form a layout such that the sum of the costs of flow between the departments in the layout is minimized. The SFLP can be formulated as follows:

\[
\text{Minimize } Z_i = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} f_{ij} d_{ik} x_{ik} x_{jk} 
\]

subject to

\[
\sum_{i=1}^{n} x_{ik} = 1, \quad \forall k = 1,2,...,n
\]

\[
\sum_{k=1}^{n} x_{ik} = 1, \quad \forall i = 1,2,...,n
\]

\[
x_{ik} \in \{0,1\}
\]
where \(x_{ik} \) equals to 1 if department \(i\) is assigned to location \(k\) in period \(t\), to 0 otherwise; \(f_{ij} \) is the quantity of material flowing between departments \(i\) and \(j\); \(d_{kl} \) is the distance between locations \(k\) and \(l\). The quadratic objective function (1) is to minimize the total flow-distance cost between each pair of departments. Constraints (2) with integral variables (4) insure each location is assigned an exact one department, and constraints (3) with integer (4) insure each department is located in an exact one location.

The dynamic facility layout problem (DFLP) augments the SFLP to take into account today’s volatile competing environment by planning the facility layouts based on multiple period horizons assuming the material flows between pairs of departments vary over time periods. In the DFLP, the material flows between the different departments in the layout planning period may change over time and a more sophisticated model might be required to solve a layout rearrangement problem to concerning costs of rearrangement of departments. The tradeoffs between the costs of excess material handling if a layout is not rearranged when required and the costs of such rearrangements needs to be considered. The DFLP can be formulated by augmenting the SFLP problem as follows:

\[
\text{Minimize } Z = \sum_{n=k}^{n=t} \sum_{j=t}^{j=1} \sum_{k=t}^{k=1} \sum_{l=t}^{l=1} f_{ij} d_{kl} x_{ik} x_{jl} + \sum_{n=k}^{n=t} \sum_{j=t}^{j=1} s_{ij}
\]

subject to

\[
\sum_{j=t}^{j=1} x_{ik} = 1, \quad \forall k = 1,2,\ldots,n, t = 1,2,\ldots,T
\]

\[
\sum_{k=t}^{k=1} x_{ik} = 1, \quad \forall i = 1,2,\ldots,n, t = 1,2,\ldots,T
\]

\[
y_{it} = 1 - \sum_{k=t}^{k=1} x_{ik} y_{it-1}, \quad \forall i = 1,2,\ldots,n, t = 2,3,\ldots,T
\]

\[
x_{ik} \in \{0,1\}, \quad \forall i = 1,2,\ldots,n, k = 1,2,\ldots,n, t = 1,2,\ldots,T
\]

\[
y_{it} \in \{0,1\}, \quad \forall i = 1,2,\ldots,n, t = 2,3,\ldots,T
\]

where \(x_{ikt} \) equals to 1 if department \(i\) is assigned to location \(k\) in period \(t\), to 0 otherwise; \(y_{it} \) equals to 1 if department \(i\) is reallocated from time period \(t-1\) to period \(t\), to 0 otherwise; \(f_{ij} \) is the quantity of material flowing between departments \(i\) and \(j\) in period \(t\); \(d_{kl} \) is the distance between locations \(k\) and \(l\); \(s_{i} \) is the cost of relocating department \(i\); \(T\) is the number of considered periods. The quadratic objective function (5) is to minimize the total layout cost being the summation of the total products of material flow between two departments and distance between their located locations and the total department relocating costs over all periods. Constraints (6) with integral variables (9) insure each location is assigned an exact one department in each period, and constraints (7) with integral variables (9) insure each department is located an exact one location in each period. Constraints (7) with integral variables (9) and (10) indicate department \(i\) is relocated or not. If there exists a location \(k\) where both variables \(x_{ikt} \) and \(x_{ik(t-1)} \) equal one, department \(i\) is located in location \(k\) in period \(t-1\) and is not relocated in period \(t\), it incurs no relocation cost, and \(y_{it} \) equals zero. If no such a location exists, department \(i\) is relocated in period \(t\), and it incurs a relocation cost \(s_{i}\). There are \(n!\) possible solutions in the SFLP and \([n!]\) possible solutions in the DFLP, so both are complicated optimization problems. For their high combinatorial complexities, we need to apply the modern heuristic solution methods to solve the DFLP problems with
reasonable large departments. In this paper we add our developed data structure into a simulated annealing (SA) heuristic existed in literature [10] to solve the DFLP efficiently. In the next section, we review the DFLP solution method in the literature. In the third section, the simulated annealing method [10] will be briefly reviewed. In the fourth section, we present the data structure for solving the DFLP problem. In the fifth section, the computational comparisons are presented and discussed, and finally we conclude this paper in the last section.

2. The DFLP related literature review

Rosenblatt [1] first defined and addressed the DFLP in 1986, and developed an exact and two heuristic procedures for solving the problem. The exact method is a dynamic programming approach that can solve only small scaled DFLP problems in a reasonable time. Both heuristics are also based on a dynamic programming in which the first one involves solving optimally the SFLP for all periods, and then the set of layouts to be considered in each period includes only the best layouts in all period, and the second one starts with a set of randomly generated arrangements to be used in each period of the DFLP. A DFLP example with six departments and 5 periods is solved with his approaches.

Balakrishnan et al. [2] added a rearrangement cost constraint to the DFLP in 1992, and solved the problem by using a constraint shortest algorithm. Urban [3] solved the DFLP with a heuristic based on applying a CRAFT (the steepest-descent pairwise-interchange) procedure once per period by developing layouts utilizing material handling cost data from varying lengths of forecast windows as well as the explicit consideration of the corresponding rearrangement costs in 1993. Total 52 test problems were generated and solved in his work.

Lackso nen and Enscore [4] studied the DFLP by assuming unit department sizes and formulated the DFLP problem as a modified quadratic assignment problem in 1993. Five algorithms (cutting plane, exchange, branch and bound, dynamic programming, and cutting tree) were modified to include the dynamic aspects. The approaches were tested with 32 DFLP test problems, and the cutting plane method found the best solutions to most of test problems, outperforming the other four algorithms. Conway and Venkataramanan [5] solved the DFLP by using a genetic algorithm (GA) to find the best layouts by using the principles of genetics to evolve an initial population of solutions into a population of superior solution in 1994. Their algorithm was tested with two DFLP samples. The first one is a 5-period and 6 department DFLP problem given in Rosenblatt [1], and the second one is a 5-period and 9 department DFLP problems given by the authors.

Lackso nen [6] presented a two-step algorithm for solving the DFLP problem while assuming the departments can have varying area in 1994. A quadratic assignment problem formulation of the DFLP is solved first by using a heuristic cutting plane procedure, and a mixed-integer linear programming problem is solved to find the desired block diagram layout. The algorithm was tested by some problems with 12 to 20 departments. Lackso nen [7] developed another solution approach for solving the DFLP with varying size in 1997. The method uses estimated location, department sized, and flow costs to determine the probable variable values.

Balakrishnan and Cheng [8] presented a good survey about the DFLP and surveyed approaches for the DFLP in 2000. In their later paper [9], they also applied genetic algorithm to solve the DFLP. Recently, Baykasoglu and Gindy [10] developed a heuristic based on the simulated annealing to solve the DFLP in 2001. They applied the heuristic to solve equal area DFLP, and the codes for the heuristic are also given in their paper, and the heuristic was tested with 48 DFLP benchmark test problems taking from literature. The computing results are compared with those published in the literature. The comparison indicates that their heuristic is among the best heuristic solution methods at that time. However, the computing time is too large to solve some reasonable large real life application. In our research, we design a type of computing data structure to speed up their heuristic to solve the DFLP problem efficiently. In the next section, we briefly review the simulated annealing heuristic for solving the DFLP given by Baykasoglu and Gindy [10], and present the data structure in the four section.

After Baykasoglu and Gindy [10] proposed their heuristics, several heuristics were also proposed to
solve the DFLP in literature. Balakrishnan et al. [13] extended and improved the use of genetic algorithms by creating a hybrid genetic algorithm to solve the DFLP in 2003. McKendall et al. [14] proposed two simulated annealing (SA) heuristics for solving the DFLP in which the first SA heuristic is a direct adaptation of SA to the DFLP and the second SA heuristic the same as SA I with a look-ahead/look-back strategy added in 2006. Ulutas and McKendall [15] proposed a clonal selection algorithm for solving the DFLP that can be extended to more general cases although equal area machines and standardized handling equipment with identical unit cost. Sahin and Turkbey [16] proposed a hybrid heuristic based on the simulated annealing (SA) approach supplemented with a tabu list for solving the DFLP in 2009. McKendall and Liu [17] proposed three tabu search (TS) heuristics for solving the DFLP in which the first one is a simple TS heuristic, the second one adds diversification and intensification strategies to the first, and the third heuristic is a probabilistic TS heuristic in 2012. Hosseini-Nasab and Emami [18] presented a hybrid particle swarm optimisation (HPSO) algorithm finding near-optimal solutions of DFLP in 2013 in which a coding and decoding technique that permits a one to one mapping of a solution in discrete space of DFLP to a PSO particle position in continuous space. Compared with all proposed heuristics, the SA proposed in Baykasoglu and Gindy [10] is still among the better solution methods for solving the DFLP in literature but it is time consuming, so we want to speed up the solution method to make the heuristic solve the DFLP problems efficiently and provide a data structure tool to help researchers design some more efficient heuristics to solve the DFRP related problem.

3. The Simulated Annealing Heuristic for the DFLP [10]

The simulated annealing (SA) was first proposed by Kirkpatrick et al. [11] based on the analogy between the process of annealing of solids and the solution methodology of combinatorial optimization problems. It has been successfully solving hundreds of combinatorial optimization problems. For more detailed SA survey, readers can refer to the book edited by Aarts and Korst [12]. By accepting with probability neighboring solutions worse than the current solution, the SA is able to jump out of a local optima for global optimization. Its acceptance probability is determined by a control parameter, called temperature, decreasing with a calling ratio during the procedure.

The elements in the SA heuristic developed in Baykasoglu and Gindy [10] consists of the followings: A configuration is a solution to the DFLP represented as a 2 dimensional matrix, each row represents a period and each column represents a location. Facilities are placed at the intersections; neighborhood moving is a transition from one configuration to another one that results in neighbor solution. The neighbor solutions are generated by a swapping two randomly selected departments in a randomly selected period; An accepting criterion is used to accept a neighboring solution if its objective function improves, or to accept it with a probability depending on the temperature; and termination rules are used to terminate the procedure. The heuristic can be briefly outlined as figure 1. For more detail of the heuristic, one can refers to Baykasoglu and Gindy [10].

Step 1. Parameter initialization; and initial configuration generating.
Step 2. Annealing schedule;
   2.a generate an neighboring solution (nei) by randomly 2-swap;
   2.b compute the objective of nei;
   2.c if the objective improves, accepting nei;
      else; accepting nei with probably exp(-d/t); where d is the deviation of the
      objective incurred for nei, and t is the current temperature;
   2.d redo step 2 until the local terminative rule is met.
Step 3. update the temperature with the cooling ratio;
Step 4. redo steps 2 and 3 until the global terminative rule is met.

Figure 1. The brief steps of the SA heuristic for solving the DFLP provided by Baykasoglu and Gindy [10]
In the heuristic, a neighbor of a current solution is generated by swapping two randomly selected departments within a randomly selected period. The objective value is recomputed once while a swapping of two departments is considered. Obviously, it takes dramatic large computing time to re-compute the new objective value of the swapping solution. This is why the heuristic is so time consuming. In this paper, we design a specific computing data structure to speed up the heuristic.

4. A Computing Data Structure for speeding up the SA heuristic

In this section we present the computing data structure for computing the cost incurred in the two-department swapping, that is the way to generate a neighboring solution of the current solution. This is main method we use to speed up the SA heuristic provided by Balakrishnan and Cheng [10]. Let \( p(i,t) \) indicate the location that the department \( i \) is located in period \( t \). For a current solution, we define a set of cost matrices associated it. Let

\[
C_f = [cf_{it}]_{n \times T} \text{ be an } n \times T \text{ matrix, where } \quad cf_{it} = \sum_{k=1, k \neq i}^{n} (f_{ik} + f_{kt})d_{p(i,t)p(k,t)} \quad \text{is the flow cost incurred by locating department } i \text{ in location } p(i,t) \text{ in period } t; \\
C_d = [cd_{it}]_{n \times (T-1)} \text{ be an } n \times (T-1) \text{ matrix, where } t=1,2,...,T-1 \text{ and } \quad cd_{it} = \begin{cases} s; & p(i,t) \neq p(i,t+1) \\ 0; & p(i,t) = p(i,t+1) \end{cases} \quad \text{is the forward relocated cost incurred by locating department } i \text{ in periods } t-1 \text{ and } t; \\
C_u = [cu_{it}]_{n \times (T-1)} \text{ be an } n \times (T-1) \text{ matrix, where } t=2,3,...,T-1 \text{ and } \quad cu_{it} = \begin{cases} s; & p(i,t) \neq p(i,t-1) \\ 0; & p(i,t) = p(i,t-1) \end{cases} \quad \text{is the backward relocated cost incurred by locating department } i \text{ in periods } t-1 \text{ and } t, \text{ then the total costs of the current solution can be computed as follows,}

\[
TC_{current} = \frac{1}{2} \left( \sum_{i=1}^{n} \sum_{t=1}^{T} cf_{it} + \sum_{i=1}^{n} \sum_{t=1}^{T-1} cd_{it} + \sum_{i=1}^{n} \sum_{t=2}^{T} cu_{it} \right).
\]  

(11)

When departments \( i \) and \( j \) are considered to swap in period \( t \) to generate a neighboring solution, that is, department \( i \) is moved from \( p(i,t) \) to \( p(j,t) \) and department \( j \) is moved from \( p(j,t) \) to \( p(i,t) \) simultaneously. The swapping cost can be computed as follows. Let

\[
\text{Save}_{ij} = cf_{it} + cf_{jt} + cu_{it} + cu_{jt} + cd_{it} + cd_{jt} + d_{p(i,t)p(j,t)}(f_{ij} + f_{ji}).
\]  

(12)

Equation (12) is the cost saved by removing departments \( i \) and \( j \) out of the current solution (11). Let

\[
\text{Cost}_{ij} = cf_{it} + cf_{jt} + cu_{it} + cu_{jt} + cd_{it} + cd_{jt} + d_{p(i,t)p(j,t)}(f_{ij} + f_{ji}), \quad \text{where}
\]
\[
cfs_{il} = \sum_{k=1}^{n} d_{p(i,t) p(k,t)} (f_{ik} + f_{kl}), \quad \text{cfs}_{lj} = \sum_{k=1}^{n} d_{p(j,t) p(k,t)} (f_{jk} + f_{jl}),
\]

\[
cds_{il} = \begin{cases} 
  s_i; & \text{if } p(j,t) \neq p(i,t+1), \text{if } t = 1, 2, \ldots, T-1; = 0 \text{ if } t = T, \\
  0; & \text{if } p(j,t) = p(i,t+1)
\end{cases}
\]

\[
cds_{lj} = \begin{cases} 
  s_j; & \text{if } p(i,t) \neq p(j,t+1), \text{if } t = 1, 2, \ldots, T-1; = 0 \text{ if } t = T, \\
  0; & \text{if } p(i,t) = p(j,t+1)
\end{cases}
\]

\[
cus_{il} = \begin{cases} 
  s_i; & \text{if } p(j,t) \neq p(i,t-1), \text{if } t = 2, 3, \ldots, T; = 0 \text{ if } t = 1, \\
  0; & \text{if } p(j,t) = p(i,t-1)
\end{cases}
\]

\[
cus_{lj} = \begin{cases} 
  s_j; & \text{if } p(i,t) \neq p(j,t-1), \text{if } t = 2, 3, \ldots, T; = 0 \text{ if } t = 1.
\end{cases}
\]

\[\text{Cost}_{ij} \text{ in (13) is the cost incurred by locating departments } i \text{ and } j \text{ in their new location. The objective value of the neighboring solution (nei) can be computed as }
\]

\[TC_{\text{nei}} = TC_{\text{current}} - \text{Save}_{ij} + \text{Cost}_{ij}.
\]

\[\text{(14)}
\]

With equation (14), the SA heuristic method for solving the DFLP can be speeded up significantly. When the neighboring solution nei is accepted, the procedure updates the matrices \(C_f\), \(C_d\), and \(C_u\) by substituting the elements \((c_{fi}, c_{fj}, cu_{ij}, cu_{ji}, cd_{ij}, cd_{ji})\) with \((cfs_{il}, cfs_{lj}, cus_{il}, cus_{lj}, cds_{il}, cds_{lj})\) respectively. By the computing data structure, we can speed up the SA heuristic for solving the DFLP significantly, and the computational results presented in the next section can evidence the superior performance of the new computing data structure.

5. Computational results

There are 48 DFLP benchmark test problems given in Balakrishnan and Cheng [8], consisting problems of 6 (2×3), 15 (3×5), 30 (5×6) departments with 5 and 10 periods. The problems can be grouped into 6 sets as shown in Table 1.

<table>
<thead>
<tr>
<th>Problem set</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of departments</td>
<td>6</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Number of periods</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Number of problems</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

The data set of the testing problems and the original SA Fortran programs are available in Baykasoglu and Gindy [10]. In comparisons analysis, the results generated by the SA heuristic are still among the better methods in literature. We companion their Fortran programs with our data structure to form a revised version of the heuristic. We test both of their original and our revised programs with all 48 problems in a 3.2 GHz i5 CPU with 64-bit system of personal computer. The solution results generated by two versions of heuristics are all the
same, but the computing time used by the revised version is much less than the original one provided in Baykasoglu and Gindy [10]. The comparisons of computing time used by both versions of the heuristic for each problem set are given in Tables 2 to 7.

Table 2. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 1 (6 (2×3) departments, 5 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>0.312</td>
<td>0.2652</td>
<td>0.2808</td>
<td>0.312</td>
<td>0.2964</td>
<td>0.2964</td>
<td>0.3276</td>
<td>0.2808</td>
</tr>
<tr>
<td>b.</td>
<td>0.0936</td>
<td>0.0468</td>
<td>0.0936</td>
<td>0.0936</td>
<td>0.078</td>
<td>0.1092</td>
<td>0.078</td>
<td></td>
</tr>
</tbody>
</table>

| (a/b)%  | 30.00%| 17.65%| 33.33%| 30.00%| 31.58%| 26.32%| 33.33%| 27.78%|

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes

Table 3. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 2 (6 (2×3) departments, 10 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>0.9672</td>
<td>1.014</td>
<td>1.014</td>
<td>0.9984</td>
<td>1.0452</td>
<td>0.9516</td>
<td>0.9984</td>
<td>0.9672</td>
</tr>
<tr>
<td>b.</td>
<td>0.1092</td>
<td>0.156</td>
<td>0.156</td>
<td>0.156</td>
<td>0.1872</td>
<td>0.1092</td>
<td>0.156</td>
<td>0.1248</td>
</tr>
</tbody>
</table>

| (a/b)%  | 11.29%| 15.38%| 15.38%| 15.63%| 17.91%| 11.48%| 15.63%| 12.90%|

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes

Table 4. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 3 (15 (3×5) departments, 5 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>b.</td>
<td>0.3432</td>
<td>0.4056</td>
<td>0.39</td>
<td>0.39</td>
<td>0.3276</td>
<td>0.3276</td>
<td>0.3588</td>
<td>0.312</td>
</tr>
</tbody>
</table>

| (a/b)%  | 4.95% | 5.80% | 5.58% | 5.61% | 4.74% | 4.75% | 5.17% | 4.50% |

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes

Table 5. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 4 (15 (3×5) departments, 10 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>27.284</td>
<td>27.237</td>
<td>27.315</td>
<td>27.237</td>
<td>27.113</td>
<td>27.191</td>
<td>27.237</td>
<td>27.222</td>
</tr>
<tr>
<td>b.</td>
<td>0.702</td>
<td>0.6864</td>
<td>0.7488</td>
<td>0.6864</td>
<td>0.624</td>
<td>0.6552</td>
<td>0.702</td>
<td>0.6396</td>
</tr>
</tbody>
</table>

| (a/b)%  | 2.57% | 2.52% | 2.74% | 2.52% | 2.30% | 2.41% | 2.58% | 2.35% |

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes
Table 6. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 5 (30(6×5) departments, 5 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>94.4118</td>
<td>94.3182</td>
<td>94.3806</td>
<td>94.5992</td>
<td>95.1762</td>
<td>95.0826</td>
<td>95.2234</td>
<td>95.1454</td>
</tr>
<tr>
<td>b.</td>
<td>0.7644</td>
<td>0.78</td>
<td>0.5616</td>
<td>0.4524</td>
<td>0.4836</td>
<td>0.4368</td>
<td>0.4836</td>
<td>0.4056</td>
</tr>
<tr>
<td>(ab)%</td>
<td>0.81%</td>
<td>0.83%</td>
<td>0.60%</td>
<td>0.48%</td>
<td>0.51%</td>
<td>0.46%</td>
<td>0.51%</td>
<td>0.43%</td>
</tr>
</tbody>
</table>

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes

Table 7. Comparison in computational CPU time by two versions of heuristics for solving DFLP test problem set 5 (30(6×5) departments, 10 periods)

<table>
<thead>
<tr>
<th>Problem</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>379.36</td>
<td>378.16</td>
<td>379.67</td>
<td>382.32</td>
<td>383.02</td>
<td>383.18</td>
<td>382.70</td>
<td>383.15</td>
</tr>
<tr>
<td>b.</td>
<td>0.9048</td>
<td>0.8736</td>
<td>0.936</td>
<td>0.9048</td>
<td>0.936</td>
<td>0.9204</td>
<td>0.9048</td>
<td>0.8892</td>
</tr>
<tr>
<td>(ab)%</td>
<td>0.24%</td>
<td>0.23%</td>
<td>0.25%</td>
<td>0.24%</td>
<td>0.24%</td>
<td>0.24%</td>
<td>0.24%</td>
<td>0.23%</td>
</tr>
</tbody>
</table>

a: CPU seconds by the codes provided in [10], b: CPU seconds by the revised version of the codes

As pointed out in the above tables for the result comparisons in 6 test problem sets, we can cleanly find that the revised version significantly outperforms the SA heuristic given in [10] in term of computational CPU time. Moreover; the larger the problem size, the faster the revised version as shown in Table 8 by comparing average CPU times of all 6 test problem sets.

Table 8. Comparison in average computing CPU time of 6 test problems by two versions

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of departments (n)</td>
<td>6</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Number of periods (T)</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Average computing time in seconds by the codes given in [10] (Time₁)</td>
<td>0.2964</td>
<td>0.9945</td>
<td>6.942</td>
<td>27.23</td>
<td>94.79205</td>
<td>381.4497</td>
</tr>
<tr>
<td>Average computing time in seconds by the new version (Time₂)</td>
<td>0.0858</td>
<td>0.1443</td>
<td>0.35685</td>
<td>0.68055</td>
<td>0.546</td>
<td>0.9087</td>
</tr>
<tr>
<td>Percentage of Time₂/Time₁</td>
<td>28.95%</td>
<td>14.51%</td>
<td>5.14%</td>
<td>2.50%</td>
<td>0.58%</td>
<td>0.24%</td>
</tr>
</tbody>
</table>

We also conduct a multiple regression analysis to estimate the computing time with respect to the number of departments and the number of periods by these two versions with 48 test problems. The estimation equations are as follows where n is the number of departments, T is the number of planning time periods, Time₁ is computing CPU time by the original codes provided by Baykasoglu and Gindy [10], and Time₂ is computing CPU time by the revised version of the heuristic.
\[ \text{Time}_1 \approx -245 + 10.4\ n + 20.5\ T \]  

and

\[ \text{Time}_2 \approx -0.332 + 0.0243\ n + 0.0497\ T \]

Regression Analysis: Time\(_1\) versus n, T

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Coef</th>
<th>SE Coef</th>
<th>T</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-245.15</td>
<td>39.42</td>
<td>-6.22</td>
<td>0.000</td>
</tr>
<tr>
<td>n</td>
<td>10.389</td>
<td>1.107</td>
<td>9.39</td>
<td>0.000</td>
</tr>
<tr>
<td>T</td>
<td>20.510</td>
<td>4.382</td>
<td>4.68</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[ S = 75.9049 \quad \text{R-Sq} = 71.0\% \quad \text{R-Sq(adj)} = 69.7\% \]

Analysis of Variance

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>2</td>
<td>633880</td>
<td>316940</td>
<td>55.01</td>
<td>0.000</td>
</tr>
<tr>
<td>Residual Error</td>
<td>45</td>
<td>259270</td>
<td>5762</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>47</td>
<td>893150</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source \ DF \ Seq SS

| n | 1 | 507687 |
| T | 1 | 126193 |

Figure 2. Output of regression analysis of computing time in seconds by the original codes [10]

Regression Analysis: Time\(_2\) versus n, T

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Coef</th>
<th>SE Coef</th>
<th>T</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-0.33234</td>
<td>0.06557</td>
<td>-5.07</td>
<td>0.000</td>
</tr>
<tr>
<td>n</td>
<td>0.024329</td>
<td>0.001841</td>
<td>13.22</td>
<td>0.000</td>
</tr>
<tr>
<td>T</td>
<td>0.049660</td>
<td>0.007289</td>
<td>6.81</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[ S = 0.126242 \quad \text{R-Sq} = 83.1\% \quad \text{R-Sq(adj)} = 82.3\% \]

Analysis of Variance

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>2</td>
<td>3.5240</td>
<td>1.7620</td>
<td>110.56</td>
<td>0.000</td>
</tr>
<tr>
<td>Residual Error</td>
<td>45</td>
<td>0.7172</td>
<td>0.0159</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>47</td>
<td>4.2412</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source \ DF \ Seq SS

| n | 1 | 2.7842 |
| T | 1 | 0.7398 |

Figure 3. Output of regression analysis of computing time in seconds by the revised version of the heuristic
where estimation (15) with $R^2=0.71$ and estimation (16) with $R^2=0.83$, the output results of regression analysis are shown in Figure 2 and Figure 3 in which all p-values of all predictors being near zero show statistical significant of the models. When the problem size is getting larger, the revised version speeds up the original one provided in [10] significantly. For instance, as $n=50$ and $T=10$, $\text{Time}_1$ will be around 480 seconds, and $\text{Time}_2$ will be around only 1.38 seconds, only 0.29% of $\text{Time}_1$; and as $n=200$ and $T=20$, $\text{Time}_1$ will be around 2,245 seconds, and $\text{Time}_2$ will be around only 5.522 seconds, only 0.25% of $\text{Time}_1$. The results comparisons are able to show the new version speed up the existing one dramatically, and we plot the average computation time by these two versions on each of 6 sets of test problems in Figure 4. In the figure, we can point out that average time used in the new version ranges from 0.0858 to 0.9087 seconds, but average time used in the existing version ranges from 0.2964 to 381.4497 seconds. When the sizes of test problem are small, the differences aren’t so obviously, but the differences drops down dramatically when problem sizes getting larger. The computing time ratio between $\text{time}_1$ and $\text{time}_2$ is 28.95% for 6 departments and 5 periods in test problems set 1, but it drops down to only 0.24% when problem sizes become 30 departments and 10 periods in test problems set 6.

![Figure 4. Comparison in computing CPU time in second for 6 sets of test problems by two versions of the heuristics](image)

6. Conclusions

In this paper we have designed a data structure that can significantly speed up the existing SA heuristics for solving the DFLP. The average computing time used by the revised version is only less than 1 percent of that used by the best existing heuristic in literature when problem sizes become reasonable large. With this specific data preparation and structure, we can develop some more efficient solution methods for solving the DFLP and other related hard optimization problems in future.
Reference