Graph Theoretic Based Heuristics For the Facility Layout Design Problems

Louis Caccetta and Yaya S. Kusumah

School of Mathematics and Statistics
Curtin University of Technology
Perth 6001
Australia

Abstract

The facility layout problem is concerned with determining the location of a number of facilities which optimizes a prescribed objective such as profit, cost, or distance. This problem arises in many applications; for example, in design of buildings and in plant layout design. The facility layout problem has been modeled as: a quadratic assignment problem; a quadratic set covering problem; a linear integer programming problem; a graph theoretic problem. Since this problem is NP-complete, most approaches are heuristic in nature and based on graph theoretic concepts. Graph theoretically, when the objective is to maximize profit, the facility layout problem is to determine, in a given edge weighted graph G, a maximum weight planar subgraph. In this paper, we discuss a number of heuristics for this problem. The performance of the heuristics is established through a comparative analysis based on an extensive set of random test problems.

1. Introduction

Typically the facility layout design problems involve the selection of the most effective arrangement of physical facilities to obtain greatest efficiency with the combination of available resources. The objective is to produce a product or service that optimizes a prescribed objective such as total benefit; production cost; material-handling cost; or traffic flow. The facility layout problem arises in many industrial applications as detailed below.

In industry, facility layout design can be used for designing the layout of a system of facilities including buildings on a plant site or machines on a manufacturing floor. In the manufacturing context, the facility layout problem may be defined as the process of obtaining the optimal location of plant equipment (workstations). The solution of the facility layout problem is very important since it can increase the flexibility of a production plant and there are significant cost implications which directly impacts on the competitiveness of products produced.

In material handling, the facility layout problem deals with the physical arrangement of a given set of facilities so that the total cost to move the required material between the facilities is minimized. The material handling cost can comprise between 30 and 70% of the total manufacturing costs, depending on whether the facility is planned on a product or process basis.

In solving management problems, the facility layout problem techniques have been used for hospital organizations to reduce nursing staff effort and improve the patient perceive environment. Nowadays facility layout design is used not only in industrial plants, but also in other institutions, such as airports, office buildings, civic complexes, sport centres, police stations, and aircraft's instrumentation panel.

Sahni and Gonzalez [22], and Giffin et. al. [12] have shown that the facility layout design problem is NP-complete. Therefore, even though a number of exact algorithms have been proposed, computational difficulties arise in large applications. This problem has led researchers to consider suboptimal solutions generated by heuristic approaches. Graph theoretic based heuristics for solving the facility layout problem have been devised by Seppanen and Moore [24], Foulds and Robinson [10], Green-Al Hakim [13], Eades et al. [8], Kim and Kim [17], Leung [21], and Boswell [5]. Most heuristic approaches utilize graph theoretic concepts. The advantage of this is that there is no need to use a planarity test.

The existing graph theoretic based heuristics are constructive methods in which a solution is generated systematically by adding a vertex or a set of vertices into an existing face. The drawback of these methods, however, is that there are some edges, that contribute a poor weight and consequently a poor solution may be generated. The heuristic developed in this paper alleviates this problem by allowing bad edges to be removed. Testing on 600 randomly generated problems with 10 to 100 vertices provides good support for our methods.

2. Models for the Facility Layout Design Problem

As described by Kusiak and Heragu [19], the facility layout problem has been modeled as a quadratic assignment problem, a quadratic set covering problem, a linear integer programming problem, a mixed integer programming problem, and a graph theoretic problem. Koopmans and Beckman [18] have modeled facility layout location plants with material flow between them as a quadratic assignment problem (QAP). It is called a quadratic assignment problem because its objective function is a second degree function of the variables and the constraints are linear functions of the variables.

One of the early exact algorithms was devised by Koopmans and Beckman [18]. Their algorithm is based on the Quadratic Assignment Problem (QAP). A suggestion to linearize this model was given by Lawler [20], and Kaufman and Broeckx [16]. Foulds and Robinson [9] suggested a branch and bound method and used the Kuratowski criterion for planarity testing. Another suggestion is a cutting plane method, which was devised by Bazaraa and Sherali [3]. There are also several other integer programming formulations suggested by Lawler [20], Kaufmann and Broeckx [16], Heragu and Kusiak [22]. Bazaraa [2] modeled the problem as a quadratic set covering problem (QSP). Using this formulation, the total area occupied by all the facilities is divided into a number of blocks. Each facility is assigned to exactly one location and each block is occupied by at most one facility. It should be noted that in the total area occupied by all the facilities is divided into smaller blocks, the problem size increases.

Using graph theory the facility layout can be modeled as an edge-weight maximal planar graph, in which the vertices represent the facilities and the edges represent the "adjacencies". The edge weight represents either the cost or the benefit of having two facilities adjacent. When the edge weight represent cost (benefit) the problem is to find an arrangement which minimizes (maximizes) the total cost (benefit). The layout of a facility

can be obtained by constructing the dual of a maximal planar subgraph. In graph theoretic formulation it is assumed that the desirability of locating each pair of facilities adjacent to each other is given.

3. Graph Theoretic Based Heuristics

For our purposes, G = (V,E) is a finite simple graph with vertex set V and edge set E. K_n denotes the complete graph on n vertices. A graph is said to be planar if it can be drawn in the plane so that its edges intersect only at their vertices. A maximal planar graph is a planar graph with the property that adding an edge between any two non adjacent vertices results in a nonplanar graph. For convenience, let V = {1, 2, . . . , n} and denote an edge as an unordered pair (i,j); the weight of edge (i,j) is denoted by w_{ij}. Without loss of generality we can assume that our graph is complete and the edge weight w_{ij} represents the benefit of locating facilities i and j adjacent. For a general discussion of graph theoretic concepts we refer to the book of Bondy and Murty [4].

Given a weighted graph G, the facility layout problem is to find a maximum weighted spanning subgraph G′ of G that is planar. Mathematically the problem is:

$$\text{Maximize } B(G) = \sum_{(i,j)\in E'} w_{ij}x_{ij}$$

subject to

$x_{ij}=0,1$, for all i,j,

and $G'=(V,E')$, where $E'= \{(i,j): x_{ij} = 1\}$, is a planar graph.

As noted in the introduction, a number of constructive heuristics that generate a maximal planar graph have been proposed. Typically, these heuristics start with a K_3 or K_4 and build up the solution through vertex insertion, maintaining planarity at every stage. A major advantage of such methods is that there is no need to use planarity test at any stage of the insertion process. Once the final graph is obtained, the corresponding block plan can be easily drawn by converting a dual graph to a block layout, using procedures devised by Al-Hakim [1], Hasan and Hogg [14], or Green and Al-Hakim [13]. By using a new matrix representation of a planar graph, a floor plan as a dual graph can be easily constructed. Below is an example of a block plan together with the corresponding maximal planar graph.

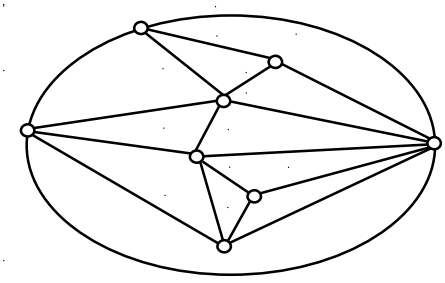

Figure 1. A graph and its corresponding block layout

Some of the heuristics incorporate an improvement procedure. The main graph theoretic based heuristics developed to date are those of: Foulds and Robinson [9] (Deltahedron Method), Green and Al-Hakim [13], Leung [21], Kim and Kim [17], Eades et al. [8] (Wheel Expansion Method), and Boswell [5] (TESSA).

In this paper, we present a new constructive heuristic. Computational results based on 600 randomly generated problems demonstrates the superiority of our heuristic over the seven literature heuristics mentioned above. A brief description of the literature heuristic is given below.

(i) The Deltahedron Method (Foulds and Robinson [9])

The application of graph theory to facility layout design was initiated by Foulds and Robinson [9] with the publication of the Deltahedron Method in 1976. The method involves simple insertion. Starting with an initial K_4, vertices are inserted one by one according to a benefit criteria. At each step, the maximum benefit of inserting each unused vertex is calculated and the vertex yielding the highest benefit is selected and inserted into the current generated subgraph. The initial K_4 can be determined (S-Construction) by selecting the four highest weighted vertices (the weight of a vertex is the sum of the weights of the edges incident to it); an alternative method (R-construction) is to generate the entire list of K_4's and select the best. Using the S-construction, a solution can be generated in $O(n^2)$. The algorithm was tested on 6 small graphs ranging from 8 to 26 vertices.

(ii) The Green-Al Hakim Algorithm [13]

This algorithm is a slight variation of the Deltahedron Method. Here we start with a maximum weight K_3 instead of a K_4 and vertices are inserted as in (i). A simple format is used to allow easy generation of the block layout plan corresponding to the solution. The complexity of this algorithm is $O(n^3)$.

(iii) The Constructive Heuristic (Leung [21])

This Heuristic can be viewed as the generalization of (i) in that at each step either one or three vertices are inserted. The single insertion option is evaluated as in (i). The triple vertex insertion involves 9 new edges, the benefit of the insertion is evaluated by dividing the sum of the weights of these 9 edges by 3. The method usually generates a better solution than (i), but the processing time is considerably longer. The complexity of the algorithm is $O(n^5)$.

This algorithm was tested using a set of 90 problems; values of n used were 20, 30, and 40. The edge weights were taken from a normal distribution with mean 100 and standard deviation 5, 10, 15, 20, 25, and 30.

(iv) The Wheel Expansion Algorithm (Eades et al. [8])

Here the initial K_4 is obtained by selecting an edge having the highest weight and then applying two successive vertex insertion according to the benefit criteria. The algorithm then proceeds with an insertion process, called the wheel expansion procedure. A wheel on n vertices is defined as a cycle on (n-1) vertices (termed the **rim**), such that each vertex is adjacent to one additional vertex (termed the **hub**).

Let W be a wheel having the hub *x*. Select two vertices *k* and *l*, which are the rims of this cycle. A vertex *y* from the set of unused vertices is then inserted to this wheel in the current partial subgraph such that *y* is a hub of the new wheel W′ containing *k*, *l* and *x* as its rims, and all rims in W are now adjacent to vertex *x* or vertex *y*. By inserting each unused vertex successively using the above fashion, the final maximal planar subgraph is obtained. The complexity of this algorithm is $O(n^4)$.

(v) The Kim-Kim Algorithm [17]

This algorithm is only a slight variation of (iii). Here instead of considering 1 vertex or 3 vertices in each iteration to obtain the highest weight insertion, only 3-vertex insertions are considered until there are two vertices or one vertex left in the set of unused vertices. This process is devoted to avoid an **umbrella effect**, a situation where a vertex is adjacent to all other vertices. The complexity of this algorithm is $O(n^4)$.

(vi) Tessa (Boswell [5])

This algorithm starts with an initial K_3 , chosen according to weight, and adds faces. It considers only triangular faces and at each step a list of available triangles is maintained. The selected face is chosen according to weight and must have at least one edge in common with the boundary of the current partial subgraph. Note that each step either a single edge or a vertex and two edges are added.

Tessa was implemented and tested using randomly generated data for each of n=10, 20, 30, and 40. The edge weights are normally distributed with a mean of 100 and standard deviations of 5, 10, 15, 20, 25, and 30. Though no computational (time) analysis is given, this algorithm requires significantly more computational time than the others. The complexity of this algorithm is $O(n^5)$.

4. A New Graph Theoretic Based Heuristic

Basically, the edge removal step as the main idea introduced in Caccetta and Kusumah [7], is used in our new algorithm. The insertion procedure in our algorithm, however, implements several new insertion types. The option of insertion can accommodate some possible high-weighted subgraphs, constructed by vertices having degree 3, 4, or 5; including octahedron or icosahedron. So, this construction technique does not restrict the type of MPG produced.

Four vertices are selected as an initial solution. This step is started by making a list of all possible subgraphs consisting of 4 vertices. Based on the sum of each weight, the best 4 vertices having the highest weight are selected as an initial subgraph K_4.

Additional vertices are inserted to the existing partial solution until all of the vertices are used. In each iteration we insert 1 or 2 vertices at a time to a face or a pair of faces in the current partial solution. We may drop a poor-weighted edge in the existing partial solution, and replace it with a better-weighted edge. An insertion of 1 vertex or 2 vertices to a pair of faces drops an edge, particularly when this edge has a poor weight and does not give a good contribution to the current partial solution.

The construction technique comprises 1-vertex and 2-vertex insertion. In 1-vertex insertions we insert a vertex to a face or to a pair of faces. When this vertex is inserted to a face, a vertex having degree 3 is added. It also constructs 3 new faces and removes 1 face. If this vertex is inserted to a pair of faces, we have a vertex having degree 4, but at the same time we also remove an existing edge from the current partial solution. There are 4 new faces constructed and 2 faces removed by this technique.

The application of 2-vertex insertion into a face gives 2 new vertices, each of which has degree 3 and 4 respectively. It also yields 5 new faces and removes 1 face. No edges are removed from the current subgraph by using this technique. If two vertices are inserted into a pair of faces, we obtain either 2 new vertices all of which have degree 4, or 2 new vertices one with degree 3 and the other with degree 5. This type of insertion gives us 6 new faces and removes 1 edge and 2 faces from the current partial subgraph.

Since a triangulation process is used at each step in this insertion technique and some new faces are constructed in each iteration, this algorithm does not need a planarity testing routine and thus is time efficient. The detailed algorithm is described below.

a. Initialization

Select 4 vertices from the list of unused vertices to form a complete graph K_4, which has the highest benefit.

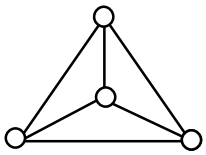

Figure 2. A complete graph K_4

b. Generation of candidates

Consider all the following possible insertions to all pair of faces having a common edge in the current partial subgraph.

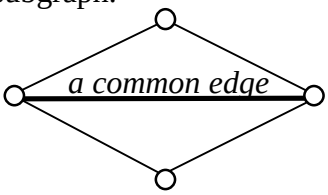

Figure 3. A pair of faces with a common edge

1. Remove the common edge from the pair of faces, to obtain a cycle of 4 vertices. Insert vertex x from the set of unused vertices into the cycle such that x is adjacent to all vertices in the cycle.
2. Remove the common edge from the pair of faces to obtain a cycle of 4 vertices. Insert vertices x and y from the set of unused vertices into the cycle such that deg(x)=deg(y)=4, x is adjacent to y, and each of x and y is adjacent to 3 vertices in the cycle.
3. Remove the common edge from the pair of faces to obtain a cycle of 4 vertices. Insert vertices x and y from the set of unused vertices into the cycle such that deg(x)=3, deg(y)=5, x and y are adjacent and each is adjacent to two vertices in the cycle.
4. Insert vertex x from the set of unused vertices into a face in the pair of faces, such that deg(x)=3, and vertex x is adjacent to all vertices in the face.
5. Insert vertices x and y from the set of unused vertices into a face in the pair of faces, such that deg(x)=4, deg(y)=3, x is adjacent to y and all vertices in the face, and y is also adjacent to 2 vertices in the face.

c. Construction

Choose the highest-weight insertion, and construct a new partial subgraph based on the corresponding type of insertion. The weight resulted from 2-vertex insertion has to be divided by 2 before compared to the weight resulted from any other insertions. Repeat until all unused vertices are inserted to the current partial subgraph.

5. Computational Results

A comparative analysis of the Deltahedron Method, The Green-Al Hakim Algorithm, The Wheel Expansion Algorithm, The Constructive Heuristic, The Kim-Kim Algorithm, Tessa, and The New Heuristic has been carried based on 600 randomly generated-uniform distribution problems. The problems consist of 30 random problems on n vertices with n ranging from 5 to 100 in increments of 5. The computational work was carried out on a Silicon Graphic Workstations (R5000) running at clock speed of 150 MHz. All algorithms were implemented in C.

Table 1 displays the performance of each algorithm compared to the upper bound. The value under the heading B (as a percentage of the upper bound) is the average total weight of the final solution. It should be noted that the upper bound is the weight of the best 3n-6 edges, and thus is not necessarily the optimal value, since the resulting graph may not be planar. We can see that the performance of the Green-Al Hakim Algorithm is better than the Deltahedron Method (S-Construction), but it is not as good as the Deltahedron Method (R-Construction). The average total benefit obtained by the Constructive Heuristic is higher than that of the Wheel Expansion Algorithm except for small values of 5, 10, and 15. The performance of the Kim-Kim Algorithm is better than that of The Deltahedron Method (S-Construction) and Tessa, but it is as good as that of The Deltahedron Method (R-Construction) and other algorithms in the table. This table also indicates that our algorithm performs well and the solutions are better than those of the other algorithms.

Table 1 shows also the average of computational CPU time of each algorithm, measured in seconds. This table indicates that Tessa is the most inefficient algorithm. Constructive heuristic, the only heuristic that implements 3-vertex insertion, has high total CPU time. The most efficient heuristic, in terms of time, is the Deltahedron Method (S-Construction). Its insertion process, based on a preselected order of vertices and 1-vertex insertions is simple and fast.

6. Conclusion and Discussion

The Deltahedron Method (Foulds and Robinson [9]) construction involves successive insertions of a vertex producing a MPG with minimum degree three. There are some MPGs with minimum degree four or five, such as octahedron and icosahedron, which also exist and may represent optimal solutions. This type of MPG, however, cannot be produced by this method.

In our heuristic vertices with degree more than 3 can be produced. For example, in a 2-vertex insertion a vertex with degree 4 or 5 is incorporated. In a 1-vertex insertion, when an edge is removed from a subgraph, the inserted vertex has degree four. This means the heuristic we develop can construct any MPG, including the octahedron and icosahedron.

In other heuristics, once a poor-weight edge is inserted to a partial subgraph, it will be always there. Thus the final solution may have a poor value. Unlike the existing heuristics, ours overcomes this problem. An edge, which has a poor weight or does not contribute a good solution to the partial subgraph, is removed and replaced by a higher-weighted edge.

In the Kim-Kim algorithm, the 3-vertex insertion is always carried whenever possible. It may be necessary to add 1-vertex insertions in the last two iterations. This procedure can save time compared to the procedure in the Constructive Heuristic. However, there is no advantage of selecting the best solution obtained from 1-vertex insertion, so the resulting benefit is not higher than that of the Constructive Heuristic.

In Tessa inserting a face to the interior is not permitted. It is only added to the boundary. However, there are some high-weight faces which cannot be inserted just because they do not fit on the boundary. This means in some instances good solutions may not be obtained. Table 1 displays the performance of Tessa which has the lowest average weights among the other heuristics.

If we refer to the computational time, Tessa is the most inefficient algorithm. This happens since in each iteration a face can be searched several times and some high-weight faces may be rejected from the insertion. In each iteration of our new algorithm, we always manage to insert one or two vertices to the existing partial solution. This process makes the algorithm time efficient.

7. References

[1] L.A. Al-Hakim, "A modified procedure for converting a dual graph to a block layout", **International Journal of Production Research**, 30(1992), 2467-2476.

[2] M.S. Bazaraa, 1975, "Computerized layout design: A branch and bound approach", **AIIE Transactions**, 7(1975), 432-437.

[3] M.S. Bazaraa, and M.D. Sherali, "Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem", **Naval Research Logistics Quarterly**, 27(1980), 29-41.

[4] J.A. Bondy and U.S.R. Murty, *Graph theory with applications*, The Mac-Millan Press, London, 1977.

[5] S.G. Boswell, "Tessa-A new greedy heuristic for facilities layout planning", **International Journal of Production Research**, 30(1992), 1957-1968.

[6] E.S. Buffa, G.C. Armour, and T.E. Vollman, "Allocating facilities with CRAFT", **Harvard Business Review**, 42(1964), 136-159.

[7] L. Caccetta and Y.S. Kusumah, "A new heuristic algorithm for the facility layout design", in **Optimization, Techniques and Applications, ICOTA '98, Proceeding (L. Caccetta et al. Editors***)*, Curtin University of Technology, (1998), 287-294.

[8] P. Eades, L.R. Foulds, L.R., and J.W. Giffin, "An efficient heuristic for identifying a maximum Weight Planar Subgraph", In **Lecture Notes in Mathematics No. 952(1982) (Combinatorial Mathematics IX)**. Springer-Verlag, Berlin.

[9] L.R. Foulds, and D.F. Robinson, "A Strategy for Solving the Plant Layout Problem", **Operational Research Quarterly**, 27(1976), 845-855.

[10] L.R. Foulds, and D.F. Robinson, "Graph theoretic heuristic for the plant layout problem", **International Journal of Production Research**, 16(1978), 27-37.

[11] L.R. Foulds, P.B. Gibbons, and J.W. Giffin, "Facilities Layout Adjacency Determination: An Experimental Comparison of Three Graph Theoretic Heuristics", **Operations Research**, 33(1985), 1091-1106.

[12] J.W. Giffin, L.R. Foulds, and D.C Cameron, "Drawing a Block Plan from a REL chart with graph theory and a microcomputer", **Computer Industrial Engineering**, 10(1984), 109-116.

[13] R.H. Green, and L. Al-Hakim, "A Heuristic for Facilities Layout Planning", **OMEGA, International Journal of Management Science**, 13(1985), 469-474.

[14] M.M.D. Hassan, and G.L. Hogg, "On constructing a block layout by graph theory", **International Journal of Production Research**, 29(1991), 1263-1278.

[15] S.S. Heragu and A. Kusiak, "Efficient models for the facility layout problem", **European Journal of Operational Research**, 5**3**(1991), 1-13.

[16] L. Kaufmann and F. Broeckx, "An Algorithm for the quadratic assignment problem using Benders' decomposition", **European Journal of Operational Research**, 2(1978), 204.

[17] J.Y. Kim and Y.D. Kim, "Graph Theoretic Heuristic for Unequal-sized Facility Layout Problems", **OMEGA, International Journal of Management Science**, 23(1985), 391-401.

[18] T.C. Koopmans and M. Beckman, "Assignment Problems and the location of economic activities", **Econometrica**, 25(1957), 53-76.

[19] A. Kusiak and S.S. Heragu, "The facility Layout Problem (Invited Review)", **European Journal of Operational Research**, 29(1987), 229-251.

[20] E.L. Lawler, "The quadratic assignment problem", **Management Science**, 9(1963), 586-599.

[21] J. Leung, "A new graph-theoretic heuristic for the facility layout**", Management Science**, 38(1992), 594-605.

[22] S. Sahni, and T. Gonzalez, "P-complete approximation problem**", Journal of the Association for Computing Machinery**, 23(1976), 55-565.

[23] J.M. Seehof and W.O. Evans, "An automated design layout program", **Journal of Industrial Engineering**, 18(1967), 690-695.

[24] J. Seppanen and J.M. Moore, "Facilities Planning with graph theory", **Management Science**, 17(1970), B242-B253.

n	Algorithms															
	Deltahedron Method (S)		Deltahedron Method (R)		Green-Al Hakim Algorithm		Constructive Heuristic		Wheel Expansion Algorithm		Tessa		Kim-Kim Algorithm		New Algorithm	
	B	T	B	T	B	T	B	T	B	T	B	T	B	T	B	T
5	99.89	0.00	99.89	0.00	99.62	0.00	99.89	0.00	100.00	0.00	97.44	0.00	99.89	0.00	100.00	0.00
10	92.41	0.00	92.62	0.00	90.22	0.00	93.72	0.00	94.29	0.01	91.46	0.01	89.53	0.00	95.15	0.01
15	89.84	0.01	90.67	0.01	86.34	0.01	91.63	0.02	91.85	0.07	89.20	0.06	90.21	0.01	92.62	0.03
20	87.93	0.03	89.88	0.05	85.49	0.04	90.98	0.11	90.69	0.40	87.40	0.51	88.28	0.05	91.69	0.14
25	87.40	0.08	89.77	0.15	85.31	0.14	90.65	0.37	90.28	1.24	87.76	2.00	87.35	0.19	91.35	0.38
30	87.32	0.16	89.72	0.36	84.88	0.35	90.66	1.02	90.43	2.72	87.64	5.61	88.60	0.46	91.45	0.77
35	87.30	0.30	89.63	0.82	84.37	0.81	90.72	2.25	90.13	5.83	87.17	14.62	88.46	1.08	91.46	1.51
40	87.40	0.63	89.60	1.97	89.13	1.90	90.71	4.92	90.28	13.73	86.79	39.18	88.15	2.31	91.28	2.98
45	87.28	0.94	90.01	3.27	89.91	3.20	90.77	8.93	90.51	22.28	86.78	75.20	88.84	4.18	91.35	4.90
50	89.09	1.70	90.05	6.54	89.99	6.41	90.69	15.89	90.35	43.97	87.56	165.68	88.87	7.62	91.66	8.36
55	87.41	2.07	90.12	8.92	89.88	8.98	90.98	25.56	90.51	58.65	86.92	238.00	88.78	11.92	91.57	10.19
60	87.42	3.48	90.49	16.04	90.28	15.95	91.24	41.03	90.69	106.29	87.05	484.11	89.34	18.97	91.75	16.47
65	87.49	4.22	90.28	21.15	90.37	21.08	91.32	59.60	90.76	136.66	86.71	684.62	89.22	28.19	91.86	20.26
70	87.63	6.69	90.56	36.38	90.53	35.90	91.50	90.81	90.83	232.31	87.42	1254.85	89.19	43.01	91.99	30.98
75	87.96	8.90	90.60	52.23	90.75	51.58	91.57	132.17	91.06	327.82	87.10	1917.15	89.99	60.55	92.07	41.15
80	87.75	10.11	90.78	64.05	90.78	63.93	91.58	178.69	91.08	388.14	87.07	2534.03	89.79	85.09	92.03	52.95
85	87.87	15.43	90.84	103.96	90.98	102.59	91.79	255.64	91.21	641.17	87.67	4295.81	89.64	120.97	92.27	75.98
90	87.96	19.46	91.03	139.93	90.93	138.56	91.86	346.16	91.37	851.72	87.85	6106.03	90.13	160.67	92.37	96.96
95	88.28	18.98	91.15	145.12	91.15	147.00	92.04	424.57	91.34	862.64	87.14	6763.24	90.15	199.78	92.54	99.66
100	88.12	29.84	91.24	242.24	91.20	240.82	92.03	581.86	91.44	1440.70	87.74	9590.57	90.10	279.34	92.43	148.14

Table 1. The average total benefit (B) and computational CPU time (T)