Arab Journal of Administrative Sciences

2004, Vol. 11, No. 2, 227 - 243

Merza H. Hasan

Kuwait University, Kuwait

FACILITY DESIGN PROBLEM: A GRAPH THEORETICAL APPROACH

Key Words

Integer Programming, Facility Design, Maximal Planar Graph, Exact Algorithm, Heuristic.

Abstract

The facility layout problem is formulated in terms of a graph theoretical approach. The solution attempts to find a sub-graph from a given weighted complete graph such that the sub-graph is planar and can be embedded on the plane without any arc intersecting. It is weighted maximal, no additional arc can be added to the sub-graph without destroying its planarity and it has the highest sum of arc weights. In this paper a new 0-1 integer programming formulation will be introduced and its variants discussed for the facility design problem using structural graph-theoretic properties. Bounding procedures, based on Linear Programming (LP) relaxation, will be presented. Moreover, a branch-and-bound heuristic algorithm will be implemented using the LP bound and a tree search method based on a set of three branching rules. The effects of the different branching rules on the quality of the LP bounds will be investigated after which the best tree-search implementation will be identified. Computational results show that optimal or near optimal solutions can be obtained for practical size problems.

Introduction

Given a complete undirected weighted graph, G = (N, A, W, F), where N is the set of nodes (facilities), A is the set of arcs, W is the set of positive weights associated with arcs, and F is the set of faces. Each arc $k \in A$ is associated with two end-nodes, i and j, and has a positive weight w_k representing a desirability value for the two facilities (nodes) to be adjacent. If G is not a complete graph, the missing arcs can be added arbitrarily at zero weights. A graph is planar if it can be embedded (drawn)

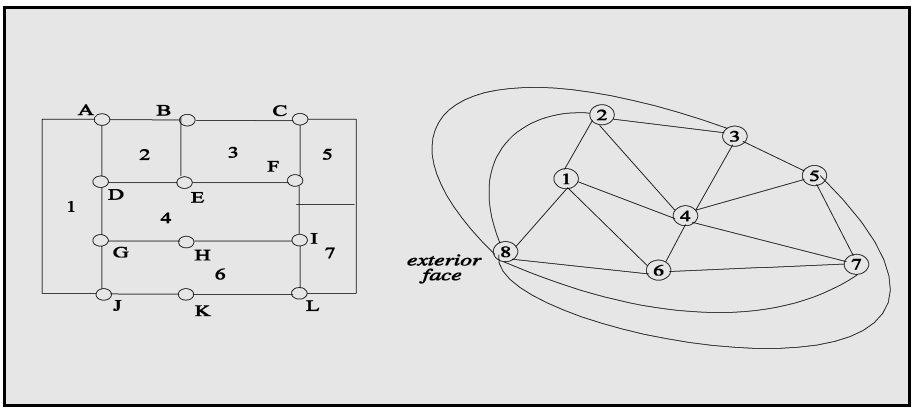
Submitted August 2003, accepted October 2003.

on the plane without any intersection of its arcs. A maximal planar graph is a planar graph, to which no additional arc can be added without destroying its planarity. The objective of the weighted maximal planar graph (WMPG) problem is to find a maximal planar sub-graph, $= (N, A_p, W_p, F_p)$ with the highest sum of arc weights, $W(G_P) = \sum_{k \in A_p} w_k$, where $A_p \subseteq A$.

In Figure 1, the weighted maximal planar graph $G_p = (N, A_p, W_p)$ can be represented by a set of nodes $N = \{1,..., 8\}$ and the set of arcs $A_p = \{(1,4), (2,4), (3,4), (5,4), (6,4), (7,4),..., (1,8)\}$. Each node of G_p is associated with one facility, while node 8 is only a special case defining the external area. Each arc of A_p is translated into a common wall (boundary) between two adjacent-end facilities in the block layout. It can be seen that all the faces of are triangular, i.e. each is bounded by three arcs and determined by three adjacent nodes. It should be noted that the planarity is required for the feasibility and optimality assured by the maximally planar graph.

Applications of the WMPG problem are in the design of skeletons for a family of facility design and location problems, Domschke and Krispin (1997). These include the layout of facilities in modern manufacturing systems, the layout and location of electrical circuits in VLSI design, (Hassan & Hogg, (1987), graph planarization, Resende and Riberio, (1997), and automatic graph drawing, Montreuil and Ratiff, (1989), Al-Hakim, (1991) and Junger and Mutzel, (1993). The WMPG problem is known to be NP-hard problem, Giffin (1984). The proposed research works

Figure 1
A WMPG solution (b) and its block layout (a)



are divided into heuristics and exact methods with a few procedures for the later one.

The heuristic algorithms for the WMPG problem can be grouped into classical heuristics and recent meta-heuristics, Osman and Laporte (1996). The classical heuristics include: Construction heuristics with a planarity testing, Moore and Carrie (1976); Carries *et al*., (1978); Foulds *et al*,. (1985) and Osman and Hasan (1997), Contruction Heuristics Without a Planarity Testing, Foulds and Robinson (1976); Eades *et al*., (1982); Green and Al-Hakim (1985); Leung (1992); Boswell 1992 and Wascher and Merker (1997) and Local Search Improvement Heuristics, Foulds *et al*., (1985); Al-Hakim (1991) and Glover, Pesch and Osman (1995). Whereas, Metaheuristics for the WMPG contains: Simulated annealing and tabu search in Hasan and Osman (1995); Greedy random adaptive search procedures and periodic improvement procedures, Barakeh (1997).

Exact algorithms were proposed by Foulds and Robinson (1976). They proposed a branch-and-bound algorithm with planarity testing for obtaining an optimal solution to the WMPG problem. The algorithm begins by arranging the set of arcs in descending order of their weights for branching. For each arc, a penalty value for not being included it in the set of feasible solutions is computed. The tree-search branches are created using a dichotomy of including or excluding an arc. The arc with the least penalty is selected and then tested for planarity before adding it to the incumbent graph. If it is non-planar, the branching arc is fathomed. The algorithm continues until all arcs with a penalty value less than that of the latest incumbent have been considered. The last incumbent is then reported as the optimal solution. An optimal solution was reported for a 10-node graph instance solved by hand. Recently, Junger and Mutzel (1996) designed an exact branch-and-cut algorithm with planarity testing and using facet-defining inequalities for a closely related problem, namely the unweighted maximal planar graph. The algorithm searches for forbidden structures in a graph that contains any sub-graph of the form K_5 (a complete graph on five nodes) or $K_{3,3}$ (a complete bipartite graph on 6 nodes and each node has three incident arcs). These structures are used to generate the facet-defining inequalities (the cutting planes). The branch-and-cut algorithm has been used to find good approximate feasible solutions (in many cases provably optimal) for sparse and dense un-weighted instances having up to 100 nodes. However, when it was tested on WMPG instances, only up to 10-node

complete graphs and up to 20-node with 20-percent density graphs were solved to optimality.

Cimikowski (1994) introduced the first exact algorithm using the tree search procedure. He utilized heuristics to obtain an initial lower bound for the size of a maximum planar sub-graph, and then he applied a sequence of planarity tests to eliminate infeasible solutions. Let G = (N, A), where N is set of nodes (facilities) and A is set of arcs, which measures the desirability of each facility to be adjacent to another. Since the planar graph can have no more than 3n-6 arcs, a graph with fewer than 9 arcs are trivially planar. Larger sized instances are more difficult and complex. Three sequential stages of branch and bound search are introduced: breadth first, backtracking, and best first solution, to find set of sub-graphs for planarity testing. At each stage the number of sub-graphs processed are reduced during the search.

The above procedures either exact or heuristic rely on the graph planarity testing to confirm its feasibility. The literature indicates a few planarity testing algorithms. Jayakumar *et al.* (1989) described two $O\ (n^2)$ planarization algorithms, based on the planarity testing algorithm of Lemple and Cedarbaun (1966). Chiba *et al.* (1983) used a path-embedding heuristic based on the planarity algorithm of Hopcroft and Tarjan (1974), where a path (P) is a sequence of connected nodes. Note that P_n denotes a graph consisting only of a path with n-1 arcs. A cycle (C) is a path with n arcs (closed path). Using a depth first search, an initial cycle is obtained in a graph G, with some arcs deleted and then embedded in the plane. The remainder of G is then decomposed into disjoint paths, and an attempt is made to embed each path inside or outside the cycle. If all paths can be embedded, the graph is planar; otherwise it is non-planar, the complexity is O (| n || m |), where m is the number of arcs and *n* is the number nodes. Generally, the depth first phase constructs a spanning planar sub-graph of a given non-planar graph by embedding one node at a time, and, at each step, adds the largest set of edges that does not lead to a non-planar graph. The second phase starts from a biconnected spanning planar sub-graph and constructs a maximal planar sub-graph containing it.

Takefuji and Lee (1989) proposed a heuristic using the separation of computation into two phases. The first phase involves devising a linear permutation of the nodes of the input graph, followed by placing them along a line. The second phase determines two sets of arcs that may be represented without crossing above and

below that line respectively. They use an arbitrary sequence of nodes in the first phase and apply a parallel heuristic using a neural network for the second phase.

Cimikowski (1992) proposed a heuristic based on finding, for each bi-connected component of a non-planar graph, a pair of arc-disjoint spanning trees whose union is planar. Although the author gives no computational results, the main interest of this approach is that, under certain conditions, the number of arcs of the generated planar subgraph is at least 2/3 of the optimum.

Cai *et al*., (1993) proposed another heuristic based on arc embedding. The initial path is obtained in the same way as the initial phase of Chiba *et al*., (1983). While the second phase, used an arc embedding rather than a path embedding. The complexity of their algorithm is $O\ (m \log n)$, where m is the number of arcs and n is the number nodes.

Goldschmidt and Takvorian (1994) introduced an improvement approach for Takefuji and Lee (1989). In the first phase, they attempt to use a linear permutation of the nodes associated with a Hamiltonian cycle of G. Two strategies are used: (i) a randomized algorithm that almost certainly finds a Hamiltonian cycle if one exists, and (ii) a greedy deterministic algorithm that seeks a Hamiltonian cycle. In the latter, the first node in the linear permutation is a minimum degree node in G. After the first k nodes of the permutation have been determined, say, $n_1, n_2, \ldots, n_k$ the next node n_{k+1} is selected from the nodes adjacent to n_k in G, having the least adjacencies in the sub-graph G_k of G induced by $N \setminus \{n_1, n_2, \ldots, n_k\}$. If there is no node of G_k adjacent to n_k in G, then n_{k+1} is selected as a minimum degree node. The linear permutation obtained in the first phase leads to the size of the planar sub-graph found in the second phase of the above heuristic. Moreover, it is not clear that the permutation produced by the greedy algorithm is the best. The total complexity of this heuristic is O (max $\{n \log n^2, m\}$).

Poutre (1994) introduced an incremental heuristic for graph planarization. The algorithm starts with an empty graph, and then adds arcs arbitrarily and one at a time, disregarding if it causes non-planarity. The remaining arcs form a maximal planar sub-grpah. After each arc addition, a planarity test is performed using an incremental planarity testing method.

Cimikowski (1995) preformed an empirical evaluation of heuristics for the graph planarization problem. Several heuristics were tested on a large and comprehensive set of problems.

These include: random graphs with unknown maximum planar sub-graph size; non-planar graphs containing a maximum planar sub-graph of size, 3n-6; and random graphs. Cimikowskis experiments show that the two-phase heuristic of Goldschmidt and Takvorian (1994) markedly outperforms the previous classical heuristics in terms of solution quality, although its running time makes it impossible for very large graphs where a minimum arc arc deletion for G is proposed to turn the graph into planar.

Last, Resende and Reiro (1997) proposed a new method for graph planarization based on GRASP; an approach that relies on graph reduction.

It can be seen from the literature survey that there are few exact solution methods for solving the GP problem. They can only solve small sized instances due to the lack of good bounding procedure. A B&B tree search procedure with upper bound obtained from the relaxation of new Integer Linear Programming (ILP) formulation for the GP problem will be presented next. The motive is to develop good exact methods for this challenging practical problem.

Integer Linear Programming Formulation (ILP)

In this section, a new integer linear programming formulation for the GP problem is introduced. It is based on the graph theoretic properties of the weighted maximal planar sub-graph solution of a complete graph. Hence, such properties will be presented first, followed by an appropriate modification to derive a GP formulation.

Graph Theoretic properties

Given a complete graph, G = (N, A, W) where N = {1,.... , n} is the set of n nodes, A = {1,...,α} is the set of undirected arcs, W = $\{w_k \mid w_k \geq 0, \forall k \in a\}$ is the set of arc weights. Let G_p = (N, A_p, W) be the weighted maximal planar sub-graph solution of G defined by its set of arcs $A_p = \{\alpha_1,...,\alpha_p\}$, and its set of triangular faces $F_p = \{f_1,...,f_t\}$; then, we have the Eulers formula (n-p + t = 2) and the following implied properties are valid for any $n \geq 3$:

a) For any given planar graph, $p \leq 3n - 6$ and $p = 3n - 6$ when it is maximal.

b) If G_p is a bipartite (connected) graph then the number of triangular faces $t = 2n - 4$.

c) G_p is a 3-node connected graph, i.e., the number of incident arcs (degree of node i) of any node i must be greater or equal to three.

From the above properties, the combinatorial nature of the weighted maximal planar graph can be illustrated from the need to find a set of A_p of (3n-6) arcs out of a total of number of arcs $\{\alpha = \frac{n(n-1)}{2}\}$ and

a set F_p of (2n-4) of triangular faces out a total number of faces $\{f=\frac{n(n-1)(n-2)}{6}\}$ that satisfy planarity and maximally requirements. For more details on graph theory terminology and details, refer to Chvatal (1969) and Harary (1971).

Graph Planarization Formulation

Let GP = (N, AP), where N is the set of n nodes, AP is the set of m arcs and WMPG is a complete graph where N is the set n nodes and A is the set $\alpha = \frac{n(n-1)}{\text{n}}$ arcs. Therefore, the main difference between GP and G is that AP $\subset A$ i.e. m $< \alpha$. In order to convert GP into WMPG instances, the following modifications are introduced. First, the set AP is augmented by a set of artificial missing arcs AA such that A = AP $\cup$ AA.

Second, the weights for the two subsets are set as follows. Let

$$w_k = \begin{cases} M & If\ an\ arc\ k\ \in\ AP \\ 1 & If\ an\ arc\ k\ \in\ AA \end{cases}$$

where M is a large number, say 100.

Now, we can proceed to an ILP formulation for the GP problem, assuming the modified weighted graph of GP is denoted by G = (N,A,W).

For simplicity, let us denote a triangular face f_j by j, and then using the above theoretic properties, the ILP formulation of the GT problem is as follows:

Variables definition:

$$X_k = \begin{cases} 1\ If\ an\ arc\ k\ is\ in\ the\ GP\ solution,\ k = 1, K, \alpha \\ 0\ Otherwise \end{cases}$$

$$Y_j = \begin{cases} 1\ If\ a\ triangular\ face\ j\ is\ selected,\quad j = 1, K, f \\ 0\ Otherwise \end{cases}$$

Constants definition:

$$\alpha_{kj} = \begin{cases} 1\ If\ an\ arc\ k\ belongs\ to\ triangular\ face\ j, \\ 0\ Otherwise \end{cases}$$

Then, the Integer Linear programming formulation is:

$$Maximize\ Z = \sum_{k=1}^{\alpha} w_k X_k \qquad (1)$$

Subject to:

$$\sum_{j=1}^{f} \alpha_{kj} Y_j = 2X_k \qquad k = 1, ..., \alpha \qquad (2)$$

$$\sum_{k=1}^{\alpha} X_k = 3n - 6 \qquad (3)$$

$$\sum_{k\in 1} X_k \geq 3 \begin{cases} \forall i \in,\ N,\ and\ \bar{i} = N \backslash \{i\} \\ I = \{(i,l), l \in \bar{i}\}. \end{cases} \qquad (4)$$

$$X_k \in (0,1), Y_j \in (0,1), \forall k \in A, \forall j \in F \qquad (5)$$

The objective function (1) is to maximize the total sum of weights for the selected arcs in the associated GP solution. Constraints (2) state that every arc belongs to two triangular faces, property (b). Constraint (3) indicates that the total number of arcs is equal to (3n-6), property (a). Constraint (4) ensures that the GP solution is 3-node connected graph, i.e. each node must be connected to at least three other nodes (degree of three)

property (c). Constraint (5) are the integral restriction.

The Branch and Bound Tree Search Method

The branch and bound (B&B) tree search procedure developed for the WMPG problem is used to find the optimal solution for the GP problem using the modified weighted maximal planar graph associated with GP.

In general, the B&B method for the GP problem starts with a modified weighted maximal planar graph instance, and the corresponding integer linear programming relaxation is used to obtain good upper bounds. Since the planarity property is embedded in the formulation, no planarity testing procedure is applied within the B&B method. The B&B procedure continues searching until an optimal solution is found. The B&B tree search can be used to derive the following approximate solutions (heuristics) for the GP problem: the first feasible heuristic and the divide and merge heuristic.

First Feasible heuristic

The First Feasible (FF) heuristic is based on a branch and bound tree search without further backtracking to prove optimality. The first feasible solution found by the branch and bound method is the heuristic solution. It satisfies all ILP formulation model constraints, but it is possibly not optimal. The aim of this approach is to find a good feasible solution with minimum time computation effort. Most of the effort of B&B tree search is to prove whether a given solution is optimal by backtracking.

Divide and Merge Heuristic

The divide and merge (D&M) heuristic is considered a new strategy for solving GP problem. It is based on the branch and bound tree search; the main thrust of the approach is to reduce the problem into sub-problems of manageable size. The sub-problems are then combined and merged to find a solution to the original problem. The D&M heuristic is based on three stages, namely graph division (decomposition), graph optimization and finally graph merger by inserting the missing arcs to obtain a feasible solution to the original problem. The steps of D&M procedure are described below:

(a) Graph Division

Previous studies introduced different graph division strategies where the weights are assigned to each arc to solve the WMPG problem. But the weights are neglected to solve the GP problem. In this case, we divide the GP graph randomly into two and three sub-graphs G_i of equal sizes.

(b) Solving Stage

A complete graph is required to solve each sub-graph G_i by the proposed ILP formulation model. G_i is sparse; therefore, additional arcs are introduced to generate a complete graph on each G_i.The sub-graph G_i will then be solved optimally (G_{Pi}) by using the B&B tree search procedure.

(c) Merger Stage

The optimal solution generated by solving B&B tree search on each sparse sub-graph G_i is to obtain G_{Pi}. Each G_{Pi} solution is a maximal planar graph. It should be noted that G_{Pi}'s are disconnected sub-graphs; therefore, we have to connect them by inserting the missing arcs and keeping the graph planner.

The optimal sub-graphs $G_{p1},....,G_{pd}$ are initially disconnected. Any two G_{pi} and G_{pj} can be connected by considering any face f_i from G_{Pi} and any $f_j \in G_{pj}$, and adding the six missing arcs if possible. The addition of the missing arcs will turn the graph into a maximum planar. If there is no such pair fi and f_j with linking arcs, another pair is considered until the sub-graphs are connected.

In order to reduce the computational effort of the B&B method to reach to a good solution, next an LP based heuristic to solve large instant problems is presented.

Linear Programming Based Meta-Heuristics

In this section, a meta-heuristic technique for solving the GP problem is presented. It is based on solving iteratively Multiple Linear Programming relaxations of Integer values (M-LPI) to obtain a reduced sub-graph from which a GP is obtained. Let G = (N, A) be a complete graph with N as a set of nodes and A a set of arcs with *m* real arcs and (*n-m*) artificial arcs. The M-LPI heuristic works as follows.

Let A_R be the set of arcs to be found. Initially A_R is an empty set. At iteration 1, the M-LPI starts by solving the LP-relaxation model to obtain a set of arcs A_I with their variable values in the upper bound solution as one. Then, the set of arc A_I is added to A_R. The M-LPI heuristic repeats the above iteration for a number of times to obtain a reduced sub-graph $G_R = (N, A_R)$. Finally, GRASP is applied to A_R to obtain a GP solution.

The following steps explain the M-LPI heuristic.

Step 0: {*Initialization*}:

1) Set $A_R = 0$ set of reduced arcs
2) $i = 0$
3) Let A_i be the set of arcs with value of one in the LP relaxation

Step 1: {*LP relaxation*}:

1) $i = i + 1$

2) Solve the LP relaxation of G
3) Identify the set of arcs with value of 1 in A_i

Step 2: {*Updating*}:

Update the reduced sub-graph, $A_R = A_R \cup A_i$.

Step 3: {*Stopping criteria*}:

If {the size of arcs in A is equal to *m*} Stop Else

Set $W_k = 1$ for all $k \in A_R$

Go to step 1

Endif

Step 4: {*Planarity testing*}:

Apply GRASP to $G_R = (N, A_R)$

Note that imax value depends on the size of the instance being solved. Normally a value of imax = 4 was used.

Computational Experience

The results of the proposed algorithms are coded in FORTRAN, run on a laptop Pentium II, 400 MHZ with 128 MB RAM and executed using FTN77 compiler. CPLEX version 6.0 is used for solving the linear integer-programming (LIP) model. A set of instances used by Resende and Ribeiro (1997) and described in Goldschmit and Takvorian (1994), is considered for testing the proposed algorithms. Table1 shows the problem name, number of nodes, number of arcs, branch-and-bound (B&B) solution, first feasible solution (FF), divide and merge (D&M) solutions and the multiple linear programming relaxations (M-LPI) solutions. The M-LPI column is divided into two sub-columns after instances G_{12}. The first part of M-LPI column contains the solution for G_1 - G_{12} instances of sizes up to 45 nodes. Since the M-LPI heuristic uses GRASP for planarity testing, a maximum number of 50 GRASP iterations is performed. While the two sub-columns contain two sets of solutions for the G_{13} - G_{19} instances of sizes varying from 50 to 150 nodes, GRASP is run for a maximum number of 50 iterations and 10000 iterations in the first and second column, respectively.

The results represent the maximum number of planar arcs obtained by different algorithms. A number of observations can be made from the results in Table1. First, we could solve up to 50 nodes by the branch-and-bound algorithm. It shows that, even disregarding the weights in a sparse graph, the WMPG and PG problems remain combinatorially very hard to solve optimally by the B&B algorithm for large-sized instances. Second, we could not solve more than 50 nodes by the FF heuristic due to the large computational effort needed for larger sizes. Third, the other two D&M and M-LPI heuristics could solve all the

instances. These heuristics obtain good solutions. The D&M heuristic performs very well for instances up to 100 nodes, where the sparse graph is divided into two independents sub-graphs. It did not perform well in the instance of 150 node size, because the graph is divided into three independents sub-graphs. Fourth, it can be seen that M-LPI heuristic is the best performing algorithm. However, its quality of solution depends on the number of iterations used by GRASP. The larger the GRASP iteration number, the better the solution.

Table 1
Comparisons of B&B, F.F, D&C and M-LPI Solutions with Others

Problem	Nodes	Arcs	3n-6	B&B	F.F*	D&M	M-LPI	
G1	10	22	24	20	20	-----	20	
G2	10	24	24	24	21	-----	24	
G3	10	25	24	24	22	-----	24	
G4	10	26	24	24	21	-----	24	
G5	10	27	24	24	21	-----	24	
G6	10	34	24	24	22	-----	24	
G7	25	69	69	69	60	60	69	
G8	25	70	69	69	60	62	69	
G9	25	71	69	69	61	62	69	
G10	25	72	69	69	62	68	69	
G11	25	90	69	68	59	66	68	
G12	45	85	129	82	77	80	82	
Average	19.17	51.25	51.50	47.17	41.83	66.33	47.17	
							Iteration	
							50	10000
G13	50	367	144	138	104	135	127	133
G14	50	491	144	144	110	140	131	141
G15	50	582	144	144	112	142	134	142
G16	100	451	296	----	----	184	188	194
G17	100	742	296	----	----	228	208	236
G18	100	922	296	----	----	224	231	244
G19	150	1064	444	----	----	288	287	306
Average	85.71	659.86	252.00	142.00	108.67	191.57	186.57	199.43

Table 2 shows a comparison of solutions between B&B, FF and M-LPI with GT of Goldschmidt and Takvorian (1994), RR of Resende and Ribeiro (1997) and JM of Junger and Mutzel (1996) heuristics. The column under M-LPI heuristic is divided into two sub-columns for G_{13}-G_{19} as explained in Table 1. From Table 2 it can be seen that the M-PLI heuristic is able to produce the same optimal solutions generated by the B&B algorithm and the JM branch and cut approach. This result ranks the M-LPI heuristic first among all other heuristics on this set of instances of sizes up to 45 nodes. On

Table 2
Comparisons of B&B, F.F, D&M and M-LPI Solutions with Others

Problem	GT	RR	JM	B&B	F.F*	D&M	M-LPI	
G1	20	20	20	20	20	-----	20	
G2	21	24	24	24	21	-----	24	
G3	21	24	24	24	22	-----	24	
G4	21	24	24	24	21	-----	24	
G5	21	24	24	24	21	-----	24	
G6	22	24	24	24	22	-----	24	
G7	60	69	69	69	60	60	69	
G8	60	69	69	69	60	62	69	
G9	59	69	69	69	61	62	69	
G10	59	69	69	69	58	68	69	
G11	62	67	68	68	59	66	68	
G12	80	82	82	82	77	80	82	
Average	42.17	47.08	47.17	47.17	42.17	47.08	47.17	
							Iteration	
							50	10000
G13	131	135	125	138	104	135	127	133
G14	136	143	133	144	110	140	131	141
G15	142	144	138	144	112	142	134	142
G16	180	196	187	----	----	184	188	194
G17	219	236	213	----	----	228	208	236
G18	237	246	223	----	----	224	231	244
G19	197	311	290	----	----	288	287	306
Average	177.43	201.57	187.00	142.00	108.67	191.57	186.57	199.43

large-sized instance G_{13} - G_{19}, the following can be observed. The results of M-LPI and D&M heuristics are better than that of GT and JM heuristics. When the number of GRASP iterations is increased from 50 to 10000, on average the solution quality of M-LPI increased from 186.57 to 199.43. For this set of instances it seems that GRASP performed slightly better than the M-LPI heuristic in terms of solution quality.

However, the CPU times of the compared heuristics are reported in Table 3. The CPU times for GT and RR heuristics are reported until the best solution is found and not until they stop. They are executed on a Silicon Graphics Challenge machine, with 250 MHz and M4400 processors. The JM heuristic is run on SUN SPARC 10/41 machine. Using the CPU benchmark reports of Kennedy and Patrick (1998) and Dongarra and Jack (2002), the following speed is in terms of Mflop/s, the rate of execution for millions of floating-point operations completed per second, as follows. The SGI Challenge 250 MHz, Intel Pentium 400 MHz and SUN SPARC 10/41 have 32, 86 and 23 Mflop/s respectively, i.e. the Intel Pentium 400 is 2.68 and 3.73 times faster than SGI Challenge and SUN SPARC 10/41 respectively. Looking at the adjusted average CPU time, it can be seen that RR is taking more time on the average than any other heuristic. Our M-LPI uses almost half the RR CPU time and produces solutions that are less than 1% on the average. In addition, we are reporting the total time rather than the time to the best solution, which is reported by RR. The huge saving is due to good graph reduction in our approach. However, the CPU reduction has resulted in a slight loss in solution quality, but would allow solving large instances better.

Conclusion

The WMPG problem known as a graph planarization problem (GP) has been considered to solve the facility design problem. The drawbacks of ILP models proposed by different authors for WMPG were modified to obtain a new formulation for the GP problem. A new mapping between a GP instance and a corresponding WMPG instance is proposed. This mapping has enabled us to use the B&B tree search procedure of the WMPG problem. Optimal solutions for instances up to 50 nodes have been obtained for the GP instances in the literature. Moreover, three different heuristics and meta-heuristics have been introduced to solve large GP instances. The first B&B tree search with stopping at first feasible (FF) solution, divides and merge (D&M) and multiple LP relaxations (M-LPI) heuristics has been implemented on large GP instances. The computational

results show that the M-LPI heuristic was ranked the best heuristic among all heuristics in the literature in terms of solution quality and computational efforts on a set of benchmark instances. A hybrid procedure between simulated annealing and tabu search can be considered for the future study.

Table 3
Comparisons of B&B, F.F, D&C and M-LPI CPU times with Others

Problem	GTb	RRb	JMc	B&Ba	F.Fa	D&Ma	M-LPIa	
G1	0.01	0.0	0.3	1.2	1.2	-----	0.0	
G2	0.00	0.0	0.1	1.8	0.8	-----	0.0	
G3	0.01	0.0	0.0	1.1	0.9	-----	0.01	
G4	0.00	0.0	0.5	1	0.8	-----	0.0	
G5	0.01	0.1	0.1	1.1	0.9	-----	0.01	
G6	0.02	0.1	0.2	1.09	0.8	-----	0.01	
G7	0.08	10.3	0.1	1258	685	2.8	2.21	
G8	0.00	10.5	0.1	1136	692	3.3	1.22	
G9	0.01	18.5	0.6	1214	771	2.9	3.35	
G10	0.0	1.2	0.6	1416	583	3.6	2.63	
G11	13.3	1.0	0.6	1894	674	3.2	4.54	
G12	0.6	1.0	10.2	5802	3435	8.6	2.24	
Average	1.17	3.56	1.12	1060.61	570.45	4.07	1.35	
							Iteration	
							50	10000
G13	8830	6830	2523	11.6	6154	508.2	1146.7	1000.0
G14	9994	7085	2618	9.93	4864	84.2	136.7	1000.0
G15	9163	7172	2557	8.14	3655	51.2	225.4	1000.0
G16	-----	-----	16812	17.4	8014	2481.9	28848.6	1000.0
G17	------	------	17989	19.3	9715	47725.2	10805.6	1000.0
G18	------	------	17182	18.6	8672	226.6	83508.1	1000.0
G19	-----	-----	28629	23.4	9318	224113.8	97939.	1000.0
Average	9329.00	7029.00	12615.71	15.48	7198.86	39313.01	31801.44	1000.0
Avergaed	9329.00	7029.00	12615.71	15.48	7198.86	14669.0	31801.4	268.0

a: CPU time on Pentium, Intel 400 MHz
b: CPU time on SGI Challenge 250 MHz
c: CPU time on SUN SPARC 10/41
d: CPU itme converted into an equivalent 400 Pentium Intel 400 MHz.

References

Al-Hakim, L. 1991. Two Graph Theoretical Procedures for an Improved Solution to the Facility Layout Problem. *Int. J. Prod. Res*, 29: 1701-1718.

Barake, M.A. 1997. *Approximate Algorithms for the Weighted Maximal Planar Graph Problem*. M.Sc. thesis, Institute of Mathematics and Statistics, University of Kent at Canterbury, UK.

Boswell, S. 1992. TESSA-A New Greedy Heuristic for Facilities Layout Planning, *Int J Prod Res*, 30: 1957-1968.

Cai, Han and Tarajan. 1992. An O (m log n)-time Algorithm of Maximal Planar Subgraph Problem, *SIAM J. Comput*., 22: 1142-1162, 1994.

Carrie, A., J. Moore, M., M. Roczniak. and J. Seppanen 1978. Graph Theory and Computer Aided Facilities Design, *Omega*, 6 (4): 353-361.

Chiba, N., J. Nishizeki, and N. Saito 1983. An Algorithm for Finding a Large Independent Set in Planar Graph, *Networks*, 13: 247-252.

Chvatal, V. 1969. Planarity of Graphs with Given Degree of Vertices, *Nieuw Arch. Wisk*., 17.

Cimikowski R. 1992. *Graph Planarization and Skewenss*. In Proc. 23rd Southeastern Conference on Combinatorics, Tampa, Florida.

Cimikowski, R., 1994. Branch and Bound Techniques for the Maximum Planar Subgraph Problem. *Int. J. Computer Math*. 53: 135-147.

Cimikowski, R. 1995. An Analysis of Some Heuristics for Maximum Planar Sub-graph Problems. *Discrete Algorithms*, 322-331.

Domschke, W. and Krispin G. 1997. Location and Layout Planning: A survey. *OR-Spektrum*, 19: 181-194.

Dongarra, Jack. 2002. *Performance of Various Computers Using Standard Linear Equations Software*; http://www.netlib.org/benchmark/performance.ps.

Eades, L.,L. Foulds, and J. Giffin, 1982. An Efficient Heuristics for Identifying a Maximum Wright Planar Subgraph, *Lecture Notes in Mathematics* No. 952, Combinatorial Mathematics IX, Spring-Verlag, Berlin.

Foulds, L.R. and Robinson D.F. 1976. A Strategy for Solving the Plant Layout Problem. *Operations Research Quarterly* 27: 845-855.

Foulds, LR, P. Gibbons and Giffin J. 1985. Facilities Layout Adjacency Determination: an Experimental Comparison of Three Graph Theoretic Heuristics. *Oper Res*, 33: 1091-1106.

Giffin, J.W. 1984. *Graph Theory Techniques for Facilities Layout*. Ph.D. thesis, University of Canterbury, New Zealand.

Goldschmidt, and Takvorian. 1994. An Efficient Graph Planarization Two-face Heuristic. *Networks*, 24: 69-73.

Green R. and L.A. Al-Hakim 1985. An Heuristic for Facilities Layout Planning. *Omega,* 13: 469-474.

Harary, F. 1971. *Graph Theory*, Addison Wasley, Reading.

Hasan, M. and Osman I.H. 1995. Local Search Algorithms for the Maximal Planar Layout Problem. *International Transactions in Operational Research*, 2: 89-106.

Hassan M.M.D. and Hogg G.L. 1991. On Cnstructing a Bock Lyout by Gaph Theory. *Iternational Journal of Production Research*, 32: 231-234.

Hasan, M. and Hogg, G. 1987. A Review of graph Theory Application to the Facility Layout Problem, *Omega*, 15, (4): 291-300.

Hopcroft, J. and Tarjan, R. 1974. Efficient Planarity Testing. *Journal of the Association for Computer Machinery*, 21, (1): 549-468.

Jayakumar *et al*. 1989. O(n^2) Algorithmes for Graph Planarization, *IEEE Trans., on Comput. Aises Design*, vol 8, pp 257-267.

Junger, M. and Mutzel P 1996. Maximum Weighted Planar Sub-graphs and Nice Embeddings: Practical Layout Tools. Algorithmica 16: 33-59.

Junger, M. and Mutzel P. 1993, Solving the Maximum Weighted Planar Subgraph Problem by Branchand Cut, in G. Rinaldi and L. Wolsey L., (eds), *Proceedings of the Third Conference on Integer Programming and Combinatorial Optimization* (IPCO), 479-492.

Kennedy, Patrick. 1998. *RISC Processor Benchmarks Scores*. http:// Kennedey p.iccim.com/risscore.htm.

Lempel, A. and I. Cedarbaum, 1966. An Algorithm for Planarity Testing of Graphs. *Grapph Theory Symposium, Rome*, Italy.

Leung, J. 1992. A New Graph-theoretic Heuristic for Facility Layout. *Management Science*, 38: 595-605.

Moore, J. and A. Carrie 1976. Facilities Design with Graph Theory and String, *Omega*, 4, (2): 193-203.

Montreuil, M., and Ratliff, H. 1989. Utilization Cut Trees and Design Skeletons for Facility Layout. *IIE Transactions*, 21: 136-143.

Osman, I.H., and G. Laporte, 1996, *Metahueristics: Theory and Applications*, Kluwer publishers, Boston, USA.

Osman, I.H., and G. Laporte, 1996, Metaheuristics: A Bibliography. *Annals of Operations Research*, 63, 513-628.

Osman, I.H. and Hasan M. 1997. A Layered Matching-planarity Testing Heuristic for the Machine Layout Planning. *Egyptian Computer Science Journal*, 19: 1-17.

Pesch, E., F.Glover, and I.H. Osman, 1995. *Efficient Facility Layout Planning in a Maximally Planar Graph Model*, Working paper, Graduate School of Business Adminstration, University of Colorado at Boulder, U.S.A.

Poutre, L.A. 1994. *Alph-time Algorithms for Incremental Planarity Testing*, In Proc. ACM Symp. on Theory of Comput., Montreal, CA.

Resende, M.G.C and Riberio C.C. 1997. A GRASP for Graph Planarization. *Networks* 29:173-189.

Takefuji, and Lee. 1989. Parallel Algorithms for Finding a Near-maximum Independent Set of a Circle Graph. *IEEE Trans. on Neural Networks*, 1: 236-267.

Wascher, G. and Merker J. 1997. A Comparative Evaluation of Heuristics for the Adjacency Problem Facility Layout Planning. *Int J Prod Res*, 35: 447-466.

الملخص

استخدام نظرية الشبكات لمعالجة تخطيط الخدمات

ميرزا حسين حسن
جامعة الكويت

مشكلة تخطيط الخدمات يمكن برمجتها باستخدام الشبكات الرقمية، حيث عناصر الشبكة المكونة من مراكز التجميع (NODES) والطرق المؤدية إلى المراكز (ARCS) يمكن ربطها لتكون مطوعة للرسم وتحديد مواقع الخدمات بالواقع العملي. تمثل الخدمات بمراكز التجمع على الشبكة ودرجة الارتباط بينها بالطرق المؤدية بين الخدمات. ابتداءً أفترض بأن جميع مراكز التجمع مرتبطة ببعضها الآخر وتعد هذه الفرضية غير واقعية، لذا يعمل النموذج المقترح بإيجاد شبكة مصغرة واقعية ترتبط ببعضها حسب أهمية وحاجة قرب المركز بمثيلاتها. تسمى الشبكة الفرعية (المصغرة) التي يمكن رسمها ضمن الأحداثيات الثنائية ويتم انعكاسها على سطح الإحداثية دون تقاطع طرقها (ARCS) بـ (PLANAR GRAPH). يعد الأسلوب المطروح في الورقة حديثاً عند برمجتها بالبرمجة الخطية الثنائية التي بدورها تعتمد على نظريات الشبكات. تم استخدام أسلوب البحث الخطي مع الأسلوب الاستكشافي للوصول إلى الحلول المرجوة وعند المقارنة بالأعمال المعروضة بالأدبيات، تبين بأن النموذج المقترح قادر على إيجاد أفضل الحلول وبسرعة عالية للوصول إلى الحل الأمثل في حال استخدام الحاسبات الآلية.

Meraz H. Hasan (Ph. D. Management Science, Imperial College of Science, Technology and Medicine, U.K, 1992) He is an Associate Professor, Quantitative and Information System Department, Kuwait University, His Research Interests are Optimization, Scheduling, Meta-heuristics and Stochastic Processes.