INFORMATICA, 1992, Vol.3, No.4, 524–538

A LOCAL SEARCH ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

Kowtha A. MURTHY, Yong LI
Computer Science Department
The Pennsylvania State University
University Park, PA 16802

Panos M. PARDALOS
303 Weil Hall, Department of Industrial and Systems Engineering
University of Florida, Gainesville, FL 32611

Abstract. In this paper, we present a new local search algorithm for solving the Quadratic Assignment Problem based on the Kernighan-Lin heuristic for the Graph Partitioning Problem. We also prove that finding a local optimum for the Quadratic Assignment Problem, with the neighborhood structure defined in the algorithm, is PLS-complete. The greatest advantages of the algorithm are its simplicity and speed in generating high quality solutions. The algorithm has been implemented and tested on an IBM 3090 computer with a variety of test problems of dimensions up to 100, including many test problems available in the literature and a new set of test problems with known optimal permutations.

Subject classifications: analysis of algorithms: computational complexity, suboptimal algorithms; facilities/equipment planning: discrete location.

Other key words: combinatorial optimization, graph partitioning, polynomial-time local search.

Introduction. The Quadratic Assignment Problem(QAP) belongs to a class of combinatorial optimization problems that have many practical applications but are computationally difficult to solve. Given a set $N = \{1, 2, \ldots, n\}$ and $n \times n$ matrices $F = (f_{ij})$ and $D = (d_{kl})$, the problem is to find a permutation p of the set N that minimizes

$$C(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}.$$

In the framework of the classical location problem, the set N describes a set of n sites on which n facilities are to be located. The matrix $F = (f_{ij})$ is the flow matrix, where f_{ij}, $i, j \in N$, represents the flow of materials from facility i to facility j. The matrix $D = (d_{kl})$ is the distance matrix, where d_{kl}, $i, j \in N$, represents the distance from location k to location l [8]. In addition to its application in facility location problems [2, 3, 8], the QAP has been found useful for such applications as problems in scheduling [5], the backboard wiring problem in electronics [25], and even the assignment of runners to a relay team [6]. Other applications may be found in [4, 9, 11].

The QAP has shown itself to be a very difficult problem computationally. This problem, of which the Traveling Salesman Problem is a special case [10], is NP-hard. Furthermore, the problem of finding an ϵ-approximate solution is also NP-hard [21]. Problems of size $n > 15$ are not practically solvable to optimality [16, 20]. Consequently, many heuristic methods have been developed to solve the QAP [1, 23, 24, 26, 27]. Most of these methods use a local search algorithm where the local search depends on the formulation of the problem and a neighborhood structure. A local search algorithm starts with an initial feasible solution and successively moves to neighboring solutions until no further improvement is possible. To characterize the complexity of solving combinatorial optimization problems such as the QAP with local search algorithms, a Polynomial-time Local Search (PLS) class has been defined [7] that captures the structure of NP problems at the level of their feasible solutions and neighborhoods. Similar to NP-completeness, the concept of PLS-completeness has been defined to capture the class of the hardest problems in PLS. For certain NP-complete problems, the corresponding PLS problems have already been shown to be PLS-complete [7, 22]. In regard to the complexity of local search, see also [17] and [19].

In Section 1 of this paper, we propose a new local search algorithm for the QAP based on the Kernighan-Lin heuristic algorithm for the Graph Partitioning Problem (GP). In Section 2, we show

that the QAP with the neighborhood structure defined by the local search algorithm is PLS-complete by reduction from the GP with the Kernighan-Lin neighborhood. We implemented and tested the algorithm on an IBM 3090 computer using a variety of test problems of sizes up to 100. Computational results are reported in Section 3.

1. A new local search algorithm. In this section, we describe a new local search algorithm for the QAP and establish the connection between the new algorithm and the Kernighan-Lin heuristic algorithm for the Graph Partitioning Problem (GP).

The local search algorithm for the QAP starts with a random permutation as a current permutation. For a current permutation p_0, a sequence of permutations, $p_1, \ldots, p_l$, is constructed in a greedy sense. Each of the permutations in the sequence is obtained from the previous one by swapping (interchanging) two assignments and has cost lower than the current permutation. A local search is performed in the sequence, replacing the current permutation by the permutation with the lowest cost in the sequence (the algorithm stops if the sequence is empty for the current permutation). In the description of the local search algorithm below, instead of using the cost $C(p_k)$ of a permutation p_k in the sequence of permutations corresponding to a current permutation p_0, we use the cumulative gain $G(k)$ of the permutation p_k, where $G(k) = C(p_0) - C(p_k)$. Hence, the larger is the cumulative gain of a permutation, the lower is the cost.

Algorithm 1: A Local Search Algorithm for the QAP.

Input: n, $n \times n$ matrices F, D, and a permutation p of size n.
Output: A local optimal permutation p for the QAP.

1. Set $p_0 = p$ and calculate its cost $C(p_0)$. Set $i = 0$, $g_i = 0$, and $G(i) = 0$, where g_i and $G(i)$ are the step gain and the cumulative gain, respectively.
2. $i = 1$. Initially, select a pair of facilities such that, by exchanging their locations, a positive step gain is obtained, i.e., $g_1 = C(p_0) - C(p_1) > 0$. If no such pair exists then go

to 7, otherwise set $G(1) = g_1$.

3. $i = i + 1$. For each pair of facilities not already selected, evaluate the step gain by exchanging their locations. Then, select the pair with maximum gain $g_i = C(p_{i-1}) - C(p_i)$. If all facilities have been selected then set $i = i - 1$ and go to 5.
4. Compute the cumulative gain, $G(i) = \sum_{k=1}^{k=i} g_k$. If $G(i) > 0$; then go to 3.
5. Select k, such that $G(k)$ is maximum for $0 \leqslant k \leqslant i$.
6. If $k > 0$ then set $p_0 = p_k$ and go to 2.
7. We have reached a local optimum for the QAP. Set $p = p_0$ and output p and $C(p)$.

Now let us review the Kernighan-Lin heuristic algorithm for the GP for an undirected graph $G(V, E)$ (assuming $|V| = 2n$) with edge weights $w(e)$, $e \in E$. For convenience, a partition of the set V always means a partition into two sets (A, B) with $|A| = |B| = |V|/2$ in the rest of the paper. Then, the problem GP is to find a partition (A, B) of the set V with the minimum cost $C(A, B)$, which is defined to be the sum of the weights of all edges between A and B. As one of the most successful heuristic algorithms for the GP, the Kernighan-Lin heuristic starts with a random partition of the set V. A sequence of partitions, $(A_1, B_1), \ldots, (A_l, B_l)$, is constructed for a current partition (A_0, B_0) in a greedy sense. Each partition (A_k, B_k), $1 \leqslant k \leqslant l$, in the sequence is obtained from the previous one (A_{k-1}, B_{k-1}) by swapping one vertex in A_{k-1} with one vertex in B_{k-1} and has cost lower than the current partition. A local search is performed in the set of partitions of this sequence, replacing the current partition by the partition with the lowest cost in the sequence (the algorithm stops if the sequence is empty for the current partition). Similar to the description of Algortihm 1, we use the cumulative gain $G(k)$ for a partition p_k.

Algorithm 2: Kernighan-Lin heuristic for the GP.

Input: $n, G = (V, E)$ with $|V| = 2n, W = (w_{ij})$, and a partition (A, B) of V.

Output: A locally optimal partition (A, B) of V.

1. Set $A_0 = A$ and $B_0 = B$, obtain its cost $C(A_0, B_0)$. Set $i = 0$, $g_i = 0$, and $G(i) = 0$, where g_i and $G(i)$ are step gain and cumulative gain, respectively.
2. i = 1. Initially, select a pair of vertices $a_1 \in A_0$ and $b_1 \in B_0$ such that, by swapping them, the resulting partition (A_1, B_1) produces a positive step gain g_1, i.e., $g_1 = C(A_0, B_0) - C(A_1, B_1) > 0$. If such a pair does not exist then goto 7, otherwise set $G(1) = g_1$.
3. $i = i + 1$. Among the vertices not selected so far, choose a pair $a_i \in A_{i-1}$ and $b_i \in B_{i-1}$ and swap them to obtain A_i and B_i with maximum step gain $g_i = C(A_{i-1}, B_{i-1}) - C(A_i, B_i)$. If all the vertices have been selected then set $i = i - 1$ and go to 5.
4. Compute the cumulative gain $G(i) = \sum_{k=1}^{k=i} g_k$. If $G(i) > 0$; then go to 3.
5. Choose k, such that $G(k)$ is maximum for $0 \leqslant k \leqslant i$.
6. If $k > 0$ then set $A_0 = A_k$ and $B_0 = B_k$ and go to 2.
7. We have reached a local optimum for the GP; set $A = A_0$ and $B = B_0$. Output A, B and $C(A, B)$.

Comparing the above algorithm with the local search algorithm for the QAP, one can easily see the similarity between them. Instead of working with partitions in the GP, we work with permutations in the QAP. The reduction from the GP to the QAP in the next section reveals why the adaptation of Kernighan-Lin algorithm to the QAP can be effective. Furthermore, extensive computational results in Section 3 indicate that the proposed local search algorithm (Algorithm 1) performs very well.

2. PLS-completeness and the QAP. For many combinatorial optimization problems, local search gives rise to some of the most successful heuristics. A classical example in this regard is the Linear Programming Problem for which the Simplex method can be viewed as a local search algorithm, in which a local search step is to go from the current basis to an adjacent basis which differs

from the current one by one column vector. Based on the pivoting rule, worst-case examples can be constructed that force the Simplex method to take exponential time. Whether there can be a pivoting rule under which the Simplex method takes only polynomial time is a major open question.

In order to characterize the complexity of such local search algorithms, a new complexity class, the Polynomial-time Local Search class, was introduced and studied in [7]. A problem P is in PLS if, for each instance $x \in I$ (the set of all instances), we have a set of feasible solutions $F(x)$ such that it is easy to decide whether $s \in F(x)$ for any solution s. Then, given $x \in I$, we can produce a feasible solution $s \in F(x)$ in polynomial time. Next, given $x \in I$ and $s \in F(x)$ we can compute the cost $C(s,x)$ of s in polynomial time. In addition, every solution $s \in F(x)$ has a set of neighboring solutions $N(s,x)$. Finally, given $x \in I$ and $s \in F(x)$, we can test in polynomial time whether s is locally optimal, and if not, produce a solution belonging to $N(s,x)$ with a better cost value (A solution s is *locally optimal* if it does not have a strictly better neighbor).

More formally, a local search problem P in PLS is defined as follows. Given an input x, find a locally optimal solution $s \in F(x)$. For the problem P, the following three polynomial time algorithms should also exist.

1. Algorithm A, on input $x \in I$, computes an initial feasible solution $s_0 \in F(x)$.
2. Algorithm B, on input $x \in I$ and $s \in F(x)$ computes $C(s,x)$.
3. Algorithm C, on input $x \in I$ and $s \in F(x)$, either determines that s is locally optimal or finds a better solution in $N(s,x)$.

A problem $P \in$ PLS is PLS-reducible to another problem $Q \in$ PLS, if there are polynomial time computable functions f and g, such that f maps an instance x of P to an instance $f(x)$ of Q and for any locally optimal solution s for $f(x)$, $g(s,x)$ produces a locally optimal solution for x. A problem P in PLS is PLS-complete, if every other problem in PLS is PLS-reducible to P.

An example of a PLS-complete problem is the GP with the Kernighan-Lin neighborhood structure defined in Algorithm 2 [7]. The Kernighan-Lin neighborhood of a partition for the GP can be defined as follows. A swap of a partition (A, B) is a partition (A', B'), where A and A' have a symmetric difference of 2, i.e., (A', B') is obtained from (A, B) by swapping one element of A with one element of B. (A', B') is a greedy swap if $C(A, B) - C(A', B')$ is maximized over all swaps of (A, B). If in fact (A', B') is the lexicographically smallest over all greedy swaps, we say that (A', B') is the lexicographic greed swap of (A, B). Let (A_i, B_i) be a sequence of partitions, each of which is a swap of the one preceding it, starting from (A_0, B_0). We call it monotonic, if the differences of $A_i - A_0$ and $B_i - B_0$ are monotonically increasing (that is, no vertex is switched back to its original set (A_0, B_0)). Finally, we say that a partition (A', B') is a neighbor of (A, B) if it occurs in the unique maximal monotonic sequence of lexicographically greedy swaps starting with (A, B). Note that such a sequence will consist of $|V|/2+1$ partitions, with the last one equal to (B, A). Thus, each partition has $|V|/2$ neighbors. The algorithm performs local search over this neighborhood structure, replacing the current partition by the partition with the lowest cost in the neighborhood.

In the remaining part of this section, we show that the QAP with the neighborhood structure defined in Algorithm 1 is PLS-complete by reduction from the GP with the Kernighan-Lin neighborhood structure. First, we show that the local search problem for the QAP is in PLS. Since the set of feasible solutions of the QAP is the set of permutations, an initial feasible solution can be produced in linear time. Computing the cost of a permutation for the QAP can be done in polynomial time. The neighborhood structure defined for the QAP in Algorithm 1 is quite similar to the Kernighan-Lin neighborhood structure for the GP. For a given permutation for the QAP, there are $\lfloor n/2 \rfloor$ neighbors. We can determine in polynomial time if the permutation is locally optimal, and if not, produce a better permutation among the $\lfloor n/2 \rfloor$ neighboring permutations. Hence, with this neighborhood structure, finding a local optimum for the QAP is in PLS.

, To prove that the PLS-completeness, we show that the GP is PLS-reducible to the QAP. Given an instance of the GP of size $2n$, we can create an instance of the QAP with the same size in polynomial time. Furthermore, for each local optimal permutation of the QAP, there is a natural local optimal partition for the corresponding GP. More specifically, suppose for the GP, the graph $G = (V, E)$ has edge weights $w(e)$ and vertex set V with $|V| = 2n$. We construct, in polynomial time, an instance of the QAP with $2n \times 2n$ matrices $F = (f_{ij})$ and $D = (d_{kl})$ defined below:

$$f_{ij} = w(i,j) \quad \text{if } (i,j) \in E; \quad \text{otherwise } f_{ij} = 0,$$

$$d_{kl} = 0 \quad \text{if } k,l \in A \text{ or } k,l \in B; \quad \text{otherwise } d_{lk} = 1,$$

$$\text{where} \quad A = \{1,2,\ldots,n\},\ B = \{n+1, n+2, \ldots, 2n\}.$$

This reduction defines a one-one correspondence between a permutation p_k of the QAP with a partition (A_k, B_k) of the vertex set V of the corresponding GP. The set of facilities allocated to locations 1 to n in p_k constitutes the set A_k. The set of facilities allocated to locations $n+1$ to $2n$ in p_k constitutes the set B_k. The cost of p_k for the QAP is exactly twice the cost of the partition (A_k, B_k) for the GP. Let the partition corresponding to a permutation p_0 be (A_0, B_0), then a permutation p_k is a neighboring permutation of p_0 if, and only if, (A_k, B_k) is a neighboring partitions of (A_0, B_0). Hence, for any local optimal permutation of the QAP, the corresponding partition is a local optimal partition for the GP and can be recovered in polynomial time. By definition, the local search problem for the QAP with the neighborhood structure defined in Algorithm 1 is PLS-complete.

3. Computational results. We describe our computational experiments with the local search algorithm. The experiments were designed to test the solution quality and the running time of the algorithm. We used several classes of test problems including test problems from Krarup and Pruzan [9], Nugent et al. [13], Skorin-Kapov [24], and Steinburg [25]. A set of new test problems with known optimal permutations was also used. They were generated

according to the test problem generator provided in [14, 12]. The test problem can be classified as follows:

Table 1. Classification of test problems

Class	Name	n (problem sizes)
1	Krarup	30, 30
2	Nugent	6, 8, 12, 15, 20, 30
3	Palubetskes	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
4	Skorin-Kapov	42, 49, 56, 64, 72, 81, 90
5	Steinburg	36, 36

For the test problems in classes 1, 2, 4, and 5, the optimal permutations are known only for problems of small sizes ($\leqslant 15$). The problems in class 3, with known optimal solutions, were generated with two integer parameters w and z using a test problem generator attached in the appendix. For each problem size in class 3, we generated 8 test problems using $w = 9$ and $z = 1, 2, ..., 8$ respectively. Since the optimal objective function value of the test problem produced by the generator is independent of z, the 8 problems for each fixed size have the same optimal objective function value. The detailed algorithm for the test problem generator is attached in the appendix.

Our algorithm was implemented in Fortran and run on an IBM 3090 computer. For the problems in classes 1, 2, 4, and 5, the algorithm was run with 25 randomly generated starting permutations which were obtained by using a random variable with uniform distribution in (0-1). For each problem size in class 3, the algorithm was run on the 8 test problems with 25 randomly generated starting permutations for each problem. The computational results are summarized in the Table 2.

From Table 2, it can be observed that the results of our algorithm are very promising in terms of both solution quality and running time. The data in the column MIN indicates that, for all the test problems considered, the best solutions found are within

Table 2. Computational results on IBM 3090

Class	n	BKV	TIME	MIN	ERR1	AVG	ERR2	MAX	ERR3
1	30	88900	0.47	90460	1.75	94512	6.31	97060	9.18
	30	91420	0.44	92380	1.05	95271	4.21	99100	8.40
2	6	86	0.00	86	0.00	91	5.81	94	9.30
	8	214	0.01	214	0.00	221	3.27	240	12.15
	12	578	0.02	592	2.42	606	4.84	632	9.34
	15	1150	0.05	1150	0.00	1200	4.35	1286	11.83
	20	2570	0.12	2602	1.25	2670	3.89	2736	6.46
	30	6124	0.73	6180	0.91	6304	2.94	6418	4.80
3	10	1890	0.01	1890	0.00	1989	5.24	2157	14.13
	20	10260	0.16	10260	0.00	10904	6.28	11415	11.26
	30	28710	0.74	28710	0.00	30303	5.55	31427	9.46
	40	60840	2.29	60840	0.00	64041	5.26	66259	8.91
	50	110250	5.55	110564	0.28	115358	4.63	120027	8.87
	60	169920	12.32	171116	0.70	177742	4.60	184129	8.36
	70	246300	23.10	249435	1.27	258437	4.93	266781	8.32
	80	341280	46.95	343239	0.57	358232	4.97	369627	8.31
	90	456570	72.33	460114	0.78	479826	5.09	493792	8.15
	100	594000	116.71	598225	0.71	620442	4.45	642000	8.08
4	42	15812	2.01	16004	1.21	16287	3.00	16520	4.48
	49	23386	3.49	23644	1.10	23932	2.33	24192	3.45
	56	34458	7.40	34802	1.00	35259	2.32	35862	4.07
	64	48498	13.04	48960	0.95	49460	1.98	50038	3.18
	72	66256	25.58	66842	0.88	67379	1.69	67948	2.55
	81	91008	44.36	91752	0.82	92535	1.68	93262	2.48
	90	115534	69.58	116276	0.64	117544	1.74	118404	2.48
5	36	9526	1.58	9738	2.23	10278	7.89	10816	13.54
	36	15852	1.53	16442	3.72	18847	18.89	22294	40.64

- BKV is the best known value for the corresponding problem.
- MIN, AVG, MAX are the minimum, average, maximum values obtained over all the runs for the corresponding problem.
- ERR1, ERR2, ERR2 are the ratios of (MIN-BKV), (AVG-BKV), (MAX-BKV) over BKV in percentages.
- TIME is the average CPU time in seconds over all runs for

4% of the best known values. The average solutions are, in general, within 8% of the best known values. The algorithm displayed stable performance except for the problems in class 5.

The algorithm seems to be very fast for the classes of test problems considered. For example, consider the problems in the class 4. For the problem of size 42 in the class 4, Skorin-Kapov reported that the construction method, the tabu search, and the simulated annealing used in [24] took 1.70, 30.65, and 42.06 seconds respectively on an IBM 3083, while the algorithm proposed here took only 2.01 seconds on an IBM 3090. For the problem of size 90, the construction method, the tabu search, and the simulated annealing used in [24] took 30.18, 727.20, and 538.69 seconds respectively, while the local search algorithm took only 69.58 seconds. However, similar to other heuristic methods for the QAP, the running time of the local search algorithm increases rapidly as the size of the problem increases.

4. Concluding remarks. In this paper, we present a new local search algorithm for the QAP based on the Kernighan-Lin heuristic algorithm for the GP. We have shown that, based on the neighborhood structure defined by the algorithm, the local search for the QAP is PLS-complete. The proposed algorithm was tested on many classes of test problems. The computational results suggest that the proposed local search algorithm is capable of computing high quality solutions quickly. In contrast to heuristics such as tabu search and simulated annealing, another advantage of the algorithm is its simplicity and easy implementation. The algorithm depends only on the starting permutation and does not depend on any other parameters. The success of the algorithm stems from the proper adaptation from the Kernighan-Lin heuristic algorithm for the GP. In addition, the proposed local search algorithm can be parallelized very efficiently. Computational results of a parallel implementation of the local search algorithm and an exact parallel branch and bound algorithm are reported in [18].

We should also mention that, at present, there are no known

relation to the global optimum. From the complexity point of view, it can be shown that, if there exists a polynomial time algorithm for checking whether a given permutation is globally optimal, then P = NP [15].

Appendix: A test problem generator. The problems in class 3 of Table 1, were generated using the test problem generator originally reported in [14] and subsequently appeared and used in [12]. For ready reference, the algorithm to generate test problems is described below (for details, see [12]).

Algorithm 3: Generating Test Problems With Known Optimal Solutions.

Input: n, w, z, positive integers, $0 \leqslant z \leqslant w$.

Output: $n \times n$ matrices F and D, an optimal permutation p and the optimal cost.

1. Construct the matrix $D = (d_{ij})$, of which the elements are the rectilinear distances between the knots of the two dimensional grid $r \times s$, where $r \times s = n$. If (i, j) are neighboring knots, then $d_{ij} = 1$.
2. Set $f_{ij} = w$, $g_{ij} = 2 - d_{ij}$ for $i, j = 1, 2, \ldots, n$.
3. While there exist $i, j \in \{1, \ldots, n\}$ such that $g_{ij} \leqslant 0$ do.
4. Choose the pair l, m, such that $d_{lm} = \max\{d_{ij}\}$, where the max is taken over every (i, j) for which $g_{ij} \leqslant 0$. If no such pair exists, then go to 8.
5. Randomly select a grid point k on one of the shortest ways from l to m, such that, $|d_{lk} - d_{mk}| \leqslant 1$. Then, choose a random integer $\Delta \in [0, z]$.
6. Set $f_{lm} = \Delta$, $f_{lk} = f_{lk} + (w - \Delta)$, $f_{mk} = f_{mk} + (w - \Delta)$ and $g_{lm} = g_{lk} = g_{mk} = 1$.
7. Endwhile.
8. Finally, generate a random permutation p as the optimal permutation. Form the matrix $F = (f_{ij})$ in which $f_{ij} = f_{uv}$, where $i = p(u)$, $j = p(v)$, $i, j = 1, 2, \ldots, n$.
9. Output $F = (f_{ij})$, $D = (d_{ij})$, p, and the optimal cost $w(\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij})$.

REFERENCES

[1] Bazaraa, M.S., and H.D. Sherali, (1980). Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly*, **27**, 29–41.

[2] Dicky, J.W., and J.W. Hopkins (1972). Campus building arrangement using TOPAZ. *Transportation Research*, **6**, 59–68.

[3] Elshafei, A.W. (1977). Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, **28**, 167–179.

[4] Francis, R.L., and J.A. White (1974). *Facility Layout and Location*. Prentice-Hall, Englewood Cliffs, N.J.

[5] Geoffrion, A.M., and G.W. Graves (1976). Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach. *Operations Research*, **24**, 595–610.

[6] Heffley, D.R. (1977). Assigning runners to a relay team. In S.P. Ladany and R.E. Machol (Eds.), *Optimal Strategies in Sports*. North-Holland, Amsterdam. pp. 169–171.

[7] Johnson, D.S., C.H. Papadimitriou, and M. Yannakakis (1988). How easy is local search? *Journal of Computer and System Sciences*, **37**, 79–100.

[8] Koopmans, T.C., and M.J. Beckmann (1957). Assignment problems and the location of economic activities. *Econometrica*, **25**, 53–76.

[9] Krarup, J., and P.M. Pruzan (1978). Computer-aided layout design. *Mathematical Programming Study*, **9**, 75–94.

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

[11] McCormik, E.J. (1970). *Human Factors Engineering*. McGraw-Hill, New York.

[12] Murthy, K.A., and P.M. Pardalos (1990). *A Polynomial-Time Approximation Algorithm for the Quadratic Assignment Problem*. Technical Report CS-33-90, The Pennsylvania State University.

[13] Nugent, C.E., T.E. Vollmann and J. Ruml (1969). An experimental comparison of techniques for the assignment of facilities to locations. *Journal of Operations Research*, **16**, 150–173.

[14] Palubetskes, G.S. (1988). Generation of quadratic assignment test problems with known optimal solutions. *Zh. Vychisl. Mat. Mat. Fiz.*, **28**(11), 1740–

1743 (in Russian).

[15] Papadimitriou, C.H., and D. Wolfe (1985). The complexity of facets resolved. In *Proceedings of the Foundations Of Computer Science,* pp. 74–78.

[16] Pardalos, P.M., and J. Crouse (1989). A parallel algorithm for the quadratic assignment problem. In *Proceedings of the Supercomputing 1989 Conference.* ACM Press. pp. 351–360.

[17] Pardalos, P.M., and S. Jha (1992). Complexity of uniqueness and local search in quadratic 0-1 programming. *Operations Research Letters,* **11**, 119–123.

[18] Pardalos, P.M., K.A. Murthy, and Y. Li (1992). Computational experience with parallel algorithms for solving the quadratic assignment problem. To appear in *Computer Science and Operations Research: New developments in their interfaces,* Williamsburg, VA, January 1992.

[19] Pardalos, P.M., and G. Schnitger (1988). Checking local optimality in constrained quadratic programming is np-hard. *Operations Research Letters,* **7**, 33–35.

[20] Roucairol, C. (1987). A parallel branch and bound algorithm for the quadratic assignment problem. *Discrete Applied Mathematics,* **18**, 211–225.

[21] Sahni, S., T. Gonzalez (1976). P-complete approximation problems.. *Journal of the Association of Computing Machinery,* **23**, 555–565.

[22] Schäffer, A.A., and M. Yannakakis (1989). Simple local search problems that are hard to solve. *Working Paper,* Bell Laboratories. pp. 1–48.

[23] Sherali, Y.D., and P. Rajgopal (1986). A flexible polynomial time construction and improvement heuristic for the quadratic assignment problem. *Computers & Operations Research,* **13**(5), 587–600.

[24] Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing,* **2**(1), 33–45.

[25] Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *SIAM Review,* **3**, 37–50.

[26] Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing,* **17**, 443–455.

[27] Wilhelm, M.R., and T.L. Ward (1987). Solving quadratic assignment problems by simulated annealing. *IEEE Transactions,* **19**, 107–119.

Received August 1992

Y. Li received his Ph. D. degree in Computer Science from the Pennsylvania State University in 1992. He is currently working at the Bell Laboratories. His research interests include parallel algorithms and combinatorial.

K. A. Murthy received his Ph. D. degree in Computer Science from the Pennsylvania State University in 1991. His research interests include software development and combinatorial algorithms. He works as an independent consultant.

P. M. Pardalos was born in Greece in 1954. He received his B. S. degree in Mathematics from Athens University and his Ph. D. degree in Computer Science from the University of Minnesota. Currently he has a visiting associate professor at the University of Florida and a professor at the Technical University of Crete, Greece. His research interests include global and combinatorial optimization, parallel algorithms and software development. He is the author and editor of several books in global optimization and serves as a managing editor of the Journal of Global Optimization.