Indian Journal of Science and Technology

A New Reformulation and an Exact Algorithm for the Quadratic Assignment Problem

Zakir Hussain Ahmed

Department of Computer Science, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), P.O. Box No. 5701, Riyadh-11432, Kingdom of Saudi Arabia; E-mail: zhahmed@gmail.com

Abstract

In this paper, we consider the quadratic assignment problem (QAP), one of the hardest NP-hard combinatorial optimization problems. We first present a new reformulation of the problem. Then a Lexisearch Algorithm (LSA) is developed for obtaining exact optimal solution to the problem. Finally, a comparative study has been carried out to show the efficiency of the algorithm against an existing algorithm for some medium sized instances from the quadratic assignment problem library, QAPLIB. The comparative study shows the efficiency of the proposed LSA based on the reformulation.

Keywords: Quadratic Assignment Problem, NP-hard, Reformulation, Exact Algorithm, Lexisearch, Alphabet Table, Bound.

1. Introduction

Koopmans and Beckmann [27] first introduced the quadratic assignment problem (QAP) as a mathematical model related to economic activities. The problem can be defined as follows: There are a set of *n* facilities and a set of *n* locations. For each pair of locations, a *distance* is specified and for each pair of facilities a *weight* or *flow* is specified (for example, amount of supplies transported between two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows.

The QAP is one of the most difficult combinatorial optimization problems. Sahni and Gonzales [38] had shown that it is NP-hard and that, unless P = NP, it is not possible to find an f-approximation algorithm, for a constant f. Several NP-hard problems can be modeled as QAPs. It has appeared in several practical applications - backboard wiring [39], scheduling problems [20], typewriter keyboards and control panels [35], hospital planning [17], archeology [28], economic problems [24], statistical analysis [25], forest parks [9], analysis of reaction chemistry [19], numerical analysis [11], placement of electronic components [30, 16] etc.

Due to practical importance and complexity, the QAP has been drawing attention of many researchers – who developed several exact and heuristic algorithms for solving the problem. In general, instances of size n > 30 cannot be solved optimally by an exact algorithm in reasonable time. Though large sized instances cannot easily be solved optimally by an exact algorithm, but there are some situations where only exact optimal solutions are required. Hence, we seek exact optimal solution to the problem. Branch-and-bound [22], Branch-and-cut [18], lexisearch [32, 3] are well-known exact algorithms for solving combinatorial optimization problems. The LSA has been successfully applied to many combinatorial optimization problems [4, 5, 7].

This paper presents a reformulation of the problem based on bias removal method used by Ahmed [5] for the travelling salesman problem (TSP). The TSP can be viewed as a particular case of the QAP when the flow between any pair of facilities is 1. Since the bias-removal method is found

* *Corresponding author:*
Zakir Hussain Ahmed (zhahmed@gmail.com)

to be useful for TSP, a reformulation of the QAP based on the bias removal method is supposed to be effective. Then we apply LSA to obtain exact optimal solution to the problem. The efficiency of our LSA against implementation of integer programming formulation (IPQAPR-IV, therein) using CPLEX 9.0 by Zhang et al. [41] has been examined for medium sized instances from the quadratic assignment problem library, QAPLIB.

This paper is organized as follows: Section 2 presents a brief overview of the literatures on the quadratic assignment problem. Problem definition and reformulation of the problem is presented in Section 3. Section 4 presents a LSA for obtaining exact optimal solution to the problem. Computational experience for the LSA has been reported in Section 5. Finally, Section 6 presents comments and concluding remarks.

2. Literature Review

The Quadratic Assignment Problem (QAP) is one of the hardest NP-hard combinatorial optimization problems. So, finding exact optimal solution for the large instances is almost impossible. However, there are quite situations where exact optimal solutions are very important, for example, laying out circuits in a VLSI chip, location of large plants and storage facilities done once in lifetime for an organization.

Several methods used to achieve exact optimal solution for the QAP as well as combinatorial optimization problem include branch-and-bound, cutting planes or combinations of these methods, like branch-and-cut, dynamic programming, and LSA. Out of them branch-and-bound are the most known and used algorithms and are defined from allocation and cutting rules, which define lower bounds for the problem. Gilmore [21] developed enumerative schemes that use lower bounds to eliminate undesired solutions. Various literatures concerning QAP branch-and-bound algorithms are available [12, 33, 34, 10, 22, 23]. In recent years, procedures that combine branch-and-bound techniques with parallel implementation are being widely used. Due to the combination, the best results for the QAP are being achieved. However, it is also important to observe that the success for the instances of bigger sizes is related to the hardware technological improvements [37, 33, 14].

Dynamic programming is often used for the special cases of QAP, where the flow matrix is the adjacency matrix of a tree. Christofides and Benavent [13] studied this case using a mixed integer linear programming (MILP) approach to the relaxed problem. Then it was solved using dynamic programming by taking advantage of the polynomial complexity of the instances. This technique was also used by Urban [40].Some literature proposed different integer programming formulation to solve the problem [1, 2]. Recently, Zhang et al. [41] proposed integer programming formulation of the problem and solved using CPLEX 9.0 and found very good results.

Bazaraa and Sherali [8] proposed cutting plane methods, but did not find satisfactory results. However, they contributed in the formulation of some heuristics that use MILP and Benders decomposition. The convergence of the employed technique is slow; hence, it is not widely used so far. Of course, it presented good quality of solutions for small sized QAP instances. Miranda et al. [30] use Benders decomposition algorithm to deal with a motherboard design problem, including linear costs in the formulation.

Padberg and Rinaldi [31] proposed a branch-and-cut, a variation of the cutting plane methods, which appears to be an alternative cutting strategy that exploits the polytope defined by the feasible solutions of the problem. Its main advantage over cutting planes is that the cuts are associated with the polytope's facets. Cuts associated with facets are more effective than the ones produced by cutting planes, so the convergence to an optimal solution is accelerated. The lack of knowledge about the QAP polytope is the reason why polyhedral cutting planes are not widely used for this problem. In this context, some researchers have been describing basic properties of the polytope that can contribute to future algorithm development [26, 18]. However, most of the above methods are based on lower bounds.

The lexicographic search (lexisearch, for short) is a systematic branch and bound approach, developed by [32], first for Loading Problem, popularly known as knapsack problem, before branch and bound approach of Little et al. [29] was published. Das [15] developed a LSA for obtaining exact optimal solution to the QAP. But, no computational experience has been reported.

3. Problem Definition and Reformulation

The problem is defined in the context of assigning n facilities to n locations. Let f_{ij} be the flow between facilities i and j, and d_{kl} be the distance between locations k and l. Let $a = \{a(1), a(2), \ldots\ldots, a(n)\}$ be an assignment, where a(i)

represents the location of the facility i. The problem is to assign to each location exactly one facility so as to minimize the cost

$$Z_a = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{a(i)a(j)} \quad (1)$$

Since there are n! assignments, so complete enumeration method is impossible for n>14, even if both flow F = [f_{ij}] and distance D = [d_{kl}] matrices are symmetric. We propose a reformulation of the problem which is based on bias removal method used by Ahmed [5] for the travelling salesman problem (TSP). It is found that bias removal method is effective for the TSP. So, our reformulation is supposed to be effective for the QAP also. Following is a reformulation of the problem.

Let

$$f_{ij} = f_{ij}^{'} + u_i + v_j \quad (2)$$

$$d_{ij} = d_{ij}^{'} + x_i + y_j \quad (3)$$

Then Z_a becomes

$$Z_a = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}^{'} d_{a(i)a(j)}^{'} + \sum_{i=1}^{n} x_{a(i)} \sum_{j=1}^{n} f_{ij} + \sum_{j=1}^{n} y_{a(j)} \sum_{i=1}^{n} f_{ij} + \sum_{i=1}^{n} u_i \sum_{j=1}^{n} d_{a(i)a(j)}^{'} + \sum_{j=1}^{n} v_j \sum_{i=1}^{n} d_{a(i)a(j)}^{'} \quad (4)$$

Also let

$$u_i = \min_{1 \le j \le n} f_{ij}, \; for\; i = 1,2,3,.....,n \quad (5)$$

$$v_j = \min_{1 \le i \le n} (f_{ij} - u_i), \; for\; j = 1,2,3,.....,n \quad (6)$$

$$x_i = \min_{1 \le j \le n} d_{ij}, \; for\; i = 1,2,3,.....,n \quad (7)$$

$$y_j = \min_{1 \le i \le n} (d_{ij} - x_i), \; for\; j = 1,2,3,.....,n \quad (8)$$

$$\alpha_i = \sum_{j=1}^{n} f_{ij}, \; for\; i = 1,2,3,.....,n \quad (9)$$

$$\beta_j = \sum_{i=1}^{n} f_{ij}, \; for\; j = 1,2,3,.....,n \quad (10)$$

$$\gamma_i = \sum_{j=1}^{n} d_{a(i)a(j)}^{'}, \; for\; i = 1,2,3,.....,n \quad (11)$$

$$\delta_j = \sum_{i=1}^{n} d_{a(i)a(j)}^{'}, \; for\; j = 1,2,3,.....,n \quad (12)$$

Then Z_a becomes

$$Z_a = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}^{'} d_{a(i)a(j)}^{'} + \sum_{i=1}^{n} \alpha_i x_{a(i)} + \sum_{j=1}^{n} \beta_j y_{a(j)} + \sum_{i=1}^{n} u_i \gamma_i + \sum_{j=1}^{n} v_j \delta_j = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}^{'} d_{a(i)a(j)}^{'} + s_a \quad (13)$$

where s_a is the cost of an assignment with respect to a surplus matrix S = [s_{ij}] with

$$s_{ij} = \alpha_i x_j + \beta_i y_j + u_i \gamma_j + v_i \delta_j \quad (14)$$

We further reduce the matrix S as follows.

$$\phi_i = \min_{1 \le j \le n} s_{ij}, \; for\; i = 1,2,3,.....,n \quad (15)$$

$$\psi_j = \min_{1 \le i \le n} (s_{ij} - \phi_i), \; for\; j = 1,2,3,.....,n \quad (16)$$

$$s_{ij}^{'} = s_{ij} - \phi_i - \psi_j, 1 \le i, j \le n \quad (17)$$

The reduced matrix S' = [s'_{ij}] becomes non-negative with at least one zero in each row and each column. So, s_a becomes

$$s_a = \sum_{i=1}^{n} \phi_i + \sum_{j=1}^{n} \psi_j + s_a^{'} = c_s + s_a^{'} \quad (18)$$

where c_s is constant and s'_a is the cost of an assignment with respect to the reduced surplus matrix S'. Hence it is enough to minimize

$$Z_a = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}^{'} d_{a(i)a(j)}^{'} + s_a^{'} \quad (19)$$

For the flow and distance matrices given in Table 1 and Table 2 respectively, all the step-by-step calculations to obtain reduced surplus matrix are shown in Table 1 to Table 6. We have c_s = 46.

Table 1. The flow matrix *F* with α and β

Facility	1	2	3	4	5	α_i
1	X	5	0	6	1	12
2	5	X	3	0	4	12
3	2	3	X	0	0	5
4	4	0	0	X	1	5
5	1	2	0	5	X	8
β_j	12	10	3	11	6	

Table 2. The distance matrix D with X and Y

Location	1	2	3	4	5	x_i
1	X	1	1	2	5	1
2	1	X	4	1	2	1
3	1	2	X	1	3	1
4	2	1	1	X	5	1
5	3	2	2	1	X	1
y_j	0	0	0	0	1	

Table 3. The reduced flow matrix F' with U and V

Facility	1	2	3	4	5	u_i
1	X	5	0	6	1	0
2	4	X	3	0	4	0
3	1	3	X	0	0	0
4	3	0	0	X	1	0
5	0	2	0	5	X	0
v_j	1	0	0	0	0	

Table 4. The reduced distance matrix D' with γ and δ

Location	1	2	3	4	5	γ_i
1	X	0	0	1	3	4
2	0	X	3	0	0	3
3	0	1	X	0	1	2
4	1	0	0	X	3	4
5	2	1	1	0	X	4
δ_j	3	2	4	1	7	

Table 5. The surplus matrix S with φ and ψ

Facility\Location	1	2	3	4	5	φ_i
1	15	14	16	13	31	13
2	12	12	12	12	22	12
3	5	5	5	5	8	5
4	5	5	5	5	16	5
5	8	8	8	8	14	8
ψ_j	0	0	0	0	3	

4. An LSA for the QAP

In LSA, the set of all possible solutions to a problem is arranged in a hierarchy, such that each incomplete word represents the block of words with this incomplete word as the leader. For the QAP, each location is considered as a letter in an alphabet and each assignment can be represented as a word with this alphabet. Thus the entire set of words in this dictionary (namely, the set of solutions) is partitioned into blocks. Bounds are computed for the values of the objective function over the blocks of words, which are then compared with the 'best solution value' found so far. If no word in the block can be better than the 'best solution value' found so far, jump over the block to the next one. If the current block, which is to be jumped over, is the last block of the present super-block, then jump out to the next super-block. Further, if the value of the current leader is already greater than or equal to the 'best solution value'; no need for checking the subsequent blocks within this super-block. However, if the bound indicates a possibility of better solutions in the block, enter into the sub block by concatenating the present leader with appropriate letter and set a bound for the new sub-block so obtained [4, 32].

Table 6. The reduced surplus matrix S'

Facility\Location	1	2	3	4	5
1	2	1	3	0	15
2	0	0	0	0	7
3	0	0	0	0	0
4	0	0	0	0	8
5	0	0	0	0	3

4.1 Incomplete Word and Block

An incomplete assignment $\{a(1), a(2), \ldots\ldots, a(k)\}$ of k facilities into first k locations, where k≤n, is called an incomplete word or a leader of length k. This incomplete word is also called a leader of length k. A block Q with a leader $\{a(1), a(2), a(3)\}$ of length three consists of all the words beginning with $\{a(1), a(2), a(3)\}$ as the string of first three letters. The block P with leader $\{a(1), a(2)\}$ of length 2 is the immediate super-block of Q and includes Q as one of its sub-blocks. The block R with leader $\{a(1), a(2), a(3), a(4)\}$ is a sub-block of Q. The block Q consists of many sub-blocks $\{a(1), a(2), a(3), a(r)\}$ one for each a(r). The block Q is the immediate super-block of block R.

4.2 Alphabet Table

Alphabet matrix, T = [t(i,j)], is a square matrix of order n formed by the positions of the elements of the reduced surplus matrix S' of order n. The i^{th} row of the matrix T consists of the positions of the elements in the i^{th} row of the matrix S' when they are arranged in the non-decreasing order of their values. If t(i,r) stands for the r^{th} element in the i^{th} row

of T, then t(i,1) corresponds to the smallest element in the i^{th} row of the matrix S'. Alphabet table "$[t(i,j)-s'_{i,t(i,j)}]$" is the combination of elements of the matrix T and their values [4].For the reduced surplus matrix S′ in Table 6, the alphabet table is shown in Table 7.

4.3 Lower Bound

In LSA, solution does not depend on lower bound, unlike branch-and-bound algorithm. The lower bound for each block leader on the objective function value is set to skip as many subproblems in the search procedure as possible. A subproblem is skipped if its lower bound exceeds the 'best solution value' found so far (i.e., upper bound) in the process. The higher the lower bound the larger the set of subproblems that are skipped. There are two parts in equation (19) – one consists of $f'_{ij}d'_{a(i)a(j)}$ and other s'_a. For the first part, we construct a matrix M from the matrices F' and D' as follows. Sort each row elements of F' excluding diagonals in ascending order and store in F" = [$f"_{ij}$]. Also, sort each column elements of D' excluding diagonals in descending order and store in D" = [$d"_{ij}$]. Then the elements of the matrix M = [m_{ij}] are calculated as

$$m_{ij} = \sum_{k=1}^{n-1} f"_{ik} d"_{kj} \tag{20}$$

It is seen that for some instances either all elements of the matrix M are zero or elements of each row are same. For that case, it will not help to accelerate the search process. So, we add $f'_{ij}d'_{ij} + f'_{ji}d'_{ji}$ to the elements m_{ij}. Thereafter, elements of each row are sorted in ascending order. For the F and D matrices given in Table 1 and Table 2, the sorted matrix M is shown in Table 8.

Now, suppose an incomplete assignment is $\{a(1), a(2), ..., a(k-1)\}$ and the location a(k) is selected for concatenation. Before concatenation, we check the bound for the leader $\{a(1), a(2), ..., a(k-1), a(k)\}$. For that, we start our computation from $(k+1)^{th}$ row of the 'alphabet table'

Table 7. The alphabet table (P is the location and V is the value of the location)

Facility	P - V	P - V	P - V	P - V	P - V
1	4-0	2-1	1-2	3-3	5-15
2	1-0	2-0	3-0	4-0	5-7
3	1-0	2-0	3-0	4-0	5-0
4	1-0	2-0	3-0	4-0	5-8
5	1-0	2-0	3-0	4-0	5-3

Table 8. The matrix M

Facility\Location	1	2	3	4	5
1	0	1	1	1	8
2	0	3	3	3	13
3	0	0	0	0	1
4	0	0	0	0	1
5	0	0	0	0	2

and traverse up to the n^{th} row, consider the value of first 'unassigned' location in each row. The lower bound for the leader can be calculated as:

$$L_k = \sum_{i=k+1}^{n} s'_{i,t(i,p)} + \sum_{i=k+1}^{n} m_{ip} \tag{21}$$

where t(i, p) is the first 'unassigned' location the i^{th} row in matrix T

The value of the incomplete assignment $\{a(1), a(2), ..., a(k)\}$ can be calculated as

$$Z_k = \sum_{i=1}^{k}\sum_{j=1}^{k} f'_{ij} d'_{a(i)a(j)} + \sum_{i=1}^{k} s'_{i,a(i)} \tag{22}$$

4.4 The Algorithm

Let F and D be the flow and distance matrices respectively, and S' be the reduced surplus matrix as discussed in section 3. The steps of the LSA can be written as follows.

Step 0: Form the 'alphabet table'. Initialize the 'best solution value' (Z_a) as large as possible, k as one, and value of assignment (Z_{k-1}) as zero.

Step 1: With the incomplete assignment of length (*k-1*) take as leader; consider the first 'unassigned and unchecked' location in the k^{th} row of the alphabet table. If (value of the location + Z_{k-1}) is greater than or equal to Z_a, go to step 4, *else*, compute the value of present 'incomplete assignment' (Z_k), after temporarily concatenating the present location to the current leader, and the 'lower bound' (L_k) for the temporary leader, go to step 2. If there is no any 'unassigned and unchecked' location found in that row, go to step 4.

Step 2: If '$(Z_k+L_k)<Z_a$', go to step 3, *else*, drop the location which was temporarily concatenated in step 1,and jump over the block, i.e., go to step 1.

Step 3: Go into the sub-block, i.e., augment the current leader; concatenate the considered location permanently to it, lengthening the leader by one, that is, *k* is increased by one. If the current 'incomplete

assignment' is a complete assignment, then replace $Z_a = Z_k$ and go to step 4, *else*, go to step 1.

Step 4: Jump out to the next super-block, i.e., decrement k by 1 (one), rejecting all the subsequent assignments from this block. If $k<1$, go to step 5, *else* go to step 1.

Step 5: Current assignment gives the optimal assignment sequence, with Z_a as the optimal solution value, and stop.

4.5 Illustration of the LSA

Working of the above LSA is explained through the example given in Table 1 and Table 2. We suppose that 'best solution value' (Z_a) is 9999. Table 9 gives the 'search table', and the symbols used therein are listed below:

GS: Go into the sub-block.
JB: Jump over the block.
JO: Jump out to the next super-block.

As illustration of the example, we initialize Z_a=9999, k=1, and Z_0=0. We start from 1st row (i.e., 1st facility) of the 'alphabet table'. Here, t(1,1) = 4, i.e., location 4 with value $b'_{14} = 0$ and $(Z_0 + s'_{14}) < Z_a$. The value for the present incomplete assignment (Z_1)=0. Now we go for bound calculation for the present leader {4}. The bound will guide us whether the facility 1 will be assigned to location 4. Lower bound for the present leader is

$$L_1 = (s'_{21} + s'_{31} + s'_{41} + s'_{51}) + (m_{21} + m_{31} + m_{41} + m_{51})$$
$$= (0+0+0+0)+(0+0+0+0) = 0$$

Since, $(Z_1 + L_1 = 0 + 0 = 0) < Z_a$, we accept the location 4 that leads to the incomplete assignment {4} with $Z_1 = 0$.Next, we go to 2nd row (i.e., 2nd facility) of the 'alphabet table'. Here, t(2,1) = 1, i.e., location 1 with value $=s'_{21} = 0$ and $(Z_1 + s'_{21}) < Z_a$. The value for the present incomplete assignment $(Z_2) = 9$. Now we go for bound calculation for the present leader {4, 1}. The bound will guide us whether the facility 2 will be assigned to location 1. Lower bound for the present leader is

$$L_2 = (s'_{32} + s'_{42} + s'_{52}) + (m_{32} + m_{42} + m_{52})$$
$$= (0+0+0)+(0+0+0) = 0$$

Since, $(Z_2 + L_2 = 9+0=9) < Z_a$, we accept the location 1 that leads to the incomplete assignment {4, 1} with $Z_2 = 9$. Proceeding in this way we obtain the 1st complete assignment as {4, 1, 2, 3, 5} with $Z_5 = 37 < Z_a$, so we replace $Z_a = 37$. Now, we jump out to the next higher order block, i.e.,

Table 9. The search table

Leaders					Z_k	L_k	Z_a	Remarks
1	*2*	*3*	*4*	*5*				
4-0 (0)					0	0	9999	GS
	1-0 (0)				9	0	9999	GS
		2-0 (9)			9	0	9999	GS
			3-0 (9)		9	5	9999	GS
				5-3 (12)	37	0	9999	GS
							37	JO
			5-8 (17)		35	0	37	GS
				3-0 (35)	41	0	37	JB, JO
		3-0 (9)			9	0	37	GS
			2-0 (9)		9	5	37	GS
				5-3 (12)	36	0	37	GS
							36	JO
			5-8 (17)		35	0	36	GS
				2-0 (35)	36	0	36	JB, JO
		5-0 (9)			24	0	36	GS
			2-0 (24)		24	0	36	GS
				3-0 (24)	32	0	36	GS
							32	JO
			3-0 (24)		24	0	32	GS
				2-0 (24)	40	0	32	JB, JO
	2-0 (0)				0	0	32	GS
		1-0 (0)			1	0	32	GS
			3-0 (1)		1	5	32	GS
				5-3 (4)	15	0	32	GS
							15	JO
			5-8 (9)		27	0	15	JB, JO
		3-0 (0)			12	0	15	GS
			1-0 (12)		21	5	15	JB
			5-8 (20)					JO
		5-0 (0)			3	0	15	GS
			1-0 (3)		12	0	15	GS
				3-0 (12)	26	0	15	JB, JO
			3-0 (3)		3	0	15	GS
				1-0 (3)	4	0	15	GS
							4	JO
	3-0 (0)				0	0	4	GS
		1-0 (0)			1	0	4	GS
			2-0 (1)		1	5	4	JB
			5-8 (9)					JO
		2-0 (0)			12	0	4	JB
		5-0 (0)			6	0	4	JB, JO
	5-7 (7)							JO

(*continued*)

Table 9. (*Continued*)

Leaders					Z_k	L_k	Z_a	Remarks
1	*2*	*3*	*4*	*5*				
2-1 (1)					1	0	4	GS
	1-0 (1)				1	0	4	GS
		3-0 (1)			2	0	4	GS
			4-0 (2)		2	5	4	JB
			5-8 (10)					JO
		4-0 (1)			7	0	4	JB
		5-0 (1)			17	0	4	JB, JO
	3-0 (1)				20	0	4	JB
	4-0 (1)				1	0	4	GS
		1-0 (1)			7	0	4	JB
		3-0 (1)			2	0	4	GS
				1-0 (2)	2	5	4	JB
				5-8 (10)				JO
			5-0 (1)		11	0	4	JB, JO
	5-7 (8)							JO
1-2 (2)					2	3	4	JB, JO
3-3 (3)					3	0	4	GS
	1-0 (3)				3	0	4	GS
			2-0 (3)		6	0	4	JB
			4-0 (3)		9	0	4	JB
			5-0 (3)		19	0	4	JB, JO
	2-0 (3)				20	0	4	JB
	4-0 (3)				3	0	4	GS
		1-0 (3)			9	0	4	JB
		2-0 (3)			6	0	4	JB
		5-0 (3)			13	0	4	JB, JO
	5-7 (10)							JO
5-15 (15)								STOP

{4, 1, 2} with $Z_3 = 9$, and try to compute another complete assignment with lesser cost. Proceeding in this way, we obtain the optimal assignment as {4, 2, 5, 3, 1} i.e., facility 1 to location 4, facility 2 to location 2, facility 3 to location 5, facility 4 to location 3, and facility 5 to location 1, with value $Z_a = 4$. Hence, the optimal assignment cost with respect to the given original matrices is $Z_a + C_s = 4 + 46 = 50$.

5. Computational Experience

The LSA has been encoded in Visual C++ on a Pentium IV personal computer with speed 3 GHz and 448 MB RAM under MS Windows XP, and tested with some [36] instances of various sizes. We carry out a comparative study between our LSA and implementation of integer programming formulation by Zhang et al. [41] (IPQAPR-IV, therein) in Table 10. We report known solution reported in QAPLIB; and the solution value, Total Computational Time (TotTime) and the computational time when the optimal solution is seen for the First Time (FirstTime) in seconds for solving the instances by our LSA in the table. The table also copies computational times (formulation setup times + IP CPU times) in seconds, and solutions as were reported by Zhang et al. [41] on a laptop with Intel Pentium M-1.70 GHz processor and 1.23 GB RAM using CPLEX 9.0.Table 10 also presents percentage of errors of the solution obtained by the algorithms for the instances. The error (%) is given by the formula $Error(\%) = (BestSol - OptSol)/OptSol \times 100\%$, where *BestSol* denotes the best solution obtained by the algorithms and *OptSol* denotes the optimal/best known solution reported in QAPLIB.

Table 10 presents results obtained by the algorithm for twenty two instances of sizes from 12 to 32. Both algorithms hit optimal solution to the same thirteen instances. Of course, for some of them optimality could not be proved by both algorithms. For the remaining nine instances, there is a difference in solution qualify. Among these nine instances, maximum percentage of error of the solution is 78.57% (to esc32b) by IPQAPR-IV formulation using CPLEX 9.0 whereas only 52.31% error (to esc32a) by LSA. On the average, LSA obtains solutions which are 10.29% away from the optimal solutions, whereas, IPQAPR-IV formulation obtains solutions which are12.24% away from the optimal solutions. Hence, on the average, our algorithm obtains better solutions.

In terms of computational time, direct comparison is not possible since the algorithms have been run in different machines. If we assume that the performances of the machines are equivalent, then for the instances whose optimality are proved, LSA is found to be better. It shows that our LSA can compete with stat-of-art methods in the literature. Also, solution by our algorithm does not rely on commercial math software, whereas solution by IPQAPR-IV formulation implementation relies on CPLEX.

In general, the LSA first finds an optimal solution and then proves the optimality of that solution, i.e., all the remaining subproblems are discarded. The table shows that, on average computational time, LSA found optimal solution within 42% of the total solution time. That is, LSA spent at least 58% of total time on proving optimality of the solutions. Therefore, for these QAPLIB instances, LSA spends a relatively large amount of time on

Table 10. Comparison of different algorithms

Instance	OptSol	LSA				IPQAPR-IV [41]		
		BestSol	Error(%)	FirstTime	TotTime	BestSol	Error(%)	TotTime
esc16a	68	68	0.00	0.60	625.30	68	0.00	14405.1
esc16b	292	292	0.00	29.10	14400.00	292	0.00	14412.9
esc16c	160	160	0.00	840.50	14400.00	160	0.00	14406.6
esc16d	16	16	0.00	2.50	14400.00	16	0.00	2687.8
esc16e	28	28	0.00	0.56	14.30	28	0.00	1004.6
esc16f	0	0	0.00	0.00	0.00	0	0.00	0.5
esc16g	26	26	0.00	0.00	0.50	26	0.00	351.1
esc16h	996	996	0.00	1.10	6228.20	996	0.00	14430.2
Partial Average			**0.00**	**109.30**	**6258.54**		**0.00**	**7712.4**
esc32a	130	198	52.31	12058.80	14400.00	194	49.23	15128.2
esc32b	168	204	21.43	14093.60	14400.00	300	78.57	15653.3
esc32c	642	662	3.12	12339.60	14400.00	736	14.64	16033.2
esc32d	200	234	17.00	4931.80	14400.00	236	18.00	15543.1
esc32e	2	2	0.00	1.25	26.05	2	0.00	68.7
esc32f	2	2	0.00	2.03	25.32	2	0.00	68.9
esc32g	6	6	0.00	0.44	0.45	6	0.00	327.7
esc32h	438	574	31.05	4099.20	14400.00	532	21.46	16140.6
Partial Average			**15.61**	**5940.84**	**9006.48**		**22.74**	**9870.5**
kra30a	88900	118820	33.66	3874.31	14400.00	107350	20.75	15908.9
kra30b	91420	118930	30.09	4099.42	14400.00	115870	26.74	15812.7
kra32	88700	115310	30.00	6170.01	14400.00	123590	39.33	16693.5
Partial Average			**31.25**	**4714.58**	**14400.00**		**28.94**	**16138.0**
scr12	31410	31410	0.00	0.14	0.27	31410	0.00	31.9
scr15	51140	51140	0.00	77.88	81.70	51140	0.00	611.8
scr20	110030	118568	7.76	12986.42	14400.00	110676	0.59	14417.2
Partial Average			**2.59**	**4354.81**	**4827.32**		**0.20**	**5020.3**
Total Average			**10.29**	**3436.78**	**8172.82**		**12.24**	**9279.02**

proving an optimal solution, and hence, a large number of subproblems are thrown by LSA. Also, it is seen that our algorithm produces two groups of instances of same size in terms of computational times with a big gap between the groups; one group takes very low computational time, whereas other group takes very high computational time. However, similar observations are reported for the traveling salesman problem and its some variations [4, 5].

6. Conclusion

We reformulate the quadratic assignment problem such that LSA can be applied efficiently to the problem, and then apply a LSA to obtain exact optimal solution to the problem. As lower bound plays a vital role in skipping the subproblems and hence, accelerates the search process, a lower bound for leader of blocks on the objective function is proposed. Then the performance of the LSA is compared with implementation of the IPQAPR-IV formulation by Zhang et al. [41] using CPLEX 9.0 for some medium sized QAPLIB instances. Computational experience shows the effectiveness of our algorithm.

We have investigated using LSA that only some medium sized instances can be solved optimally within stipulated time limit. For some small sized instances optimality of the solution could not be proved within four hours by our algorithm, for example, instances esc16b-d of size 16; whereas the instance esc32g of size 32 could be solved within 0.45second. We investigated why some small sized instances could not be solved, whereas some medium sized instances could be solved very quickly, but, we did not come to any conclusion. This definitely, depends on the data structure.

So, like the usual traveling salesman problem [5] and the bottleneck traveling salesman problem [6], a sophisticated data-guided approach may reduce the computational effort drastically and produce better solution quality within stipulated time limit. Also, the proposed lower bound method seems to be very loose; hence, a tighter bound may obtain optimal solution for some more instances within limited time frame.

7. Acknowledgements

The author is very much thankful to the honourable reviewer for his constructive comments and suggestions which help the author to improve the paper. This research was supported by the Deanery of Academic Research, Al Imam Mohammad Ibn Saud Islamic University, Riyadh, Saudi Arabia, via Grant no. 320902. The author is also very much thankful to the University for its financial support.

8. References

1. Adams W P, and Johnson T A (1994). Improved linear programming-based lower bounds for the quadratic assignment problem, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 16, 43–75.
2. Adams W P, Guignard M et al. (2007). A level-2 reformulation–linearization technique bound for the quadratic assignment problem, European Journal of Operational Research, vol 180(3), 983–996.
3. Ahmed Z H (2000). A Sequential Constructive Sampling and Related Approaches to Combinatorial Optimization, Ph.D. thesis, Tezpur University, Assam, India.
4. Ahmed Z H (2010). A lexisearch algorithm for the bottleneck traveling salesman problem, International Journal of Computer Science and Security, vol (6), 569–577.
5. Ahmed Z H (2011a). A data-guided lexisearch algorithm for the asymmetric traveling salesman problem, Mathematical Problems in Engineering, vol 2011, Article ID 750968, doi:10.1155/2011/750968.
6. Ahmed Z H (2011b). A data-guided lexisearch algorithm for the bottleneck travelling salesman problem, International Journal of Operational Research, vol 12, No. 1, 20–33.
7. Ahmed Z H (2012). An exact algorithm for the clustered travelling salesman problem, Opsearch, appeared online, doi: 10.1007/s12597-012-0107-0.
8. Bazaraa M S, Sherali H D (1980). Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem, Naval Research Logistics Quarterly, vol 27(1), 29–41.
9. Bos J (1993). A quadratic assignment problem solved by simulated annealing, Journal of Environmental Management, vol 37(2), 127–145.
10. Brixius N W, and Anstreicher K M (2001). Solving quadratic assignment problems using convex quadratic programming relaxations, Optimization Methods and Software, vol 16(1-4), 49–68.
11. Brusco M J, and Stahl S (2000). Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices, Journal of Classification, vol 17(2), 197–223.
12. Burkard R E, and Derigs U (1980). Assignment and matching problems: Solutions methods with Fortran programs, Lectures Notes in Economics and Mathematical Systems, vol 184, Springer-Verlag, New York, Secaucus, NJ.
13. Christofides N, and Benavent E (1989). An exact algorithm for the quadratic assignment problem, Operations Research, vol 37(5), 760–768.
14. Clausen J, and Perregaard M (1997). Solving large quadratic assignment problems in parallel, Computational Optimization and Applications, vol 8(2), 111–127.
15. Das S (1976). Routing and Allied Combinatorial Programming Problems: A Lexicographic Search Approach. Ph.D. Thesis, Dibrugarh University, Assam, India.
16. Duman E, and Ilhan O (2007). The quadratic assignment problem in the context of the printed circuit board assembly process, Computers and Operations Research, vol 34(1), 163–179.
17. Elshafei A N (1977). Hospital layout as a quadratic assignment problem, Operations Research Quarterly, vol 28(1), 167–179.
18. Erdoğan G, and Tansel B (2007). A branch-and-cut algorithm for the quadratic assignment problems based on linearizations, Computers & Operations Research, vol 34(4), 1085–1106.
19. Forsberg J H, Delaney R M et al. (1994). Analyzing lanthanide-included shifts in the NMR spectra of lanthanide (III) complexes derived from 1,4,7,10-tetrakis (N,N-diethylacetamido)-1,4,7,10-tetraazacyclododecane, Inorganic Chemistry, vol 34, 3705–3715.
20. Geoffrion A M, and Graves G W (1976). Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach, Operations Research, vol 24, No. 4, 595–610.
21. Gilmore P C (1962). Optimal and suboptimal algorithms for the quadratic assignment problem, SIAM Journal on Applied Mathematics, vol 10(2), 305–313.
22. Hahn P, Grant T et al. (1998). A branch-and-bound algorithm for the quadratic assignment problem based on Hungarian method, European Journal of Operational Research, vol 108(3), 629–640.
23. Hahn P M, Hightower W L (2001). Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem, Yugoslavian Journal of Operational Research, vol 11(1), 41–60.

24. Heffley D R (1980). Decomposition of the Koopmans–Beckmann problem, Regional Science and Urban Economics, vol 10(4), 571–580.
25. Hubert L (1987). Assignment methods in combinatorial data analysis, Statistics: Textbooks and Monographs Series, vol 73, Marcel Dekker.
26. Jünger M, and Kaibel V (2001). The QAP-polytope and the star transformation, Discrete Applied Mathematics, vol 111(3), 283–306.
27. Koopmans T C, and Beckmann M J (1957). Assignment problems and the location of economic activities, Econometrica, vol 25(1), 53–76.
28. Krarup J, and Pruzan P M (1978). Computer-aided layout design, Mathematical Programming Study, vol 9, 75–94.
29. Little J D C, Murthy K G et al. (1963). An Algorithm for the Travelling Salesman Problem, Operations Research, vol 11, 972–989.
30. Miranda G, Luna H P L et al. (2005). A performance guarantee heuristic for electronic components placement problems including thermal effects, Computers and Operations Research, vol 32(11), 2937–2957.
31. Padberg M W, and Rinaldi G (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM Review, vol 33(1), 60–100.
32. Pandit S N N (1963). Some quantitative combinatorial search problems, Ph.D, Thesis, Indian Institute of Technology, Kharagpur, India.
33. Pardalos P, and Crouse J (1989). A parallel algorithm for the quadratic assignment problem, Proceedings of the Supercomputing Conference 1989. ACM Press, 351–360.
34. Pardalos PM, Ramakrishnan KG et al. (1997). Implementation of a variance reduction-based lower bound in a branch-and-bound algorithm for the quadratic assignment problem, SIAM Journal on Optimization, vol 7(1), 280–294.
35. Pollatschek M A, Gershoni N et al. (1976). Optimization of the typewriter keyboard by simulation, Angewandte Informatik, vol 17, 438–439.
36. QAPLIB (1997). A quadratic assignment problem library, available from http://qaplib.uwaterloo.ca/inst.html.
37. Roucairol C (1987). A parallel branch and bound algorithm for the quadratic assignment problem, Discrete Applied Mathematics, vol 18, 211–225.
38. Sahni S, and Gonzales T (1976). P-complete approximation problems, Journal of the Association for Computing Machinery, vol 23(3), 555–565.
39. Steinberg L (1961). The backboard wiring problem: A placement algorithm, SIAM Review, vol 3(1), 37–50.
40. Urban T L (1998). Solution procedures for the dynamic facility layout problem, Annals of Operations Research, vol 76(0), 323–342.
41. Zhang H, Beltran-Royo C et al. (2010). Effective formulation reductions for the quadratic assignment problem, Computers & Operations Research, vol 37(11), 2007–2016.