Indian Journal of Science and Technology, *Vol 7(4), 480–490, April 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Data-guided Lexisearch Algorithm for the Quadratic Assignment Problem

Zakir Hussain Ahmed*

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

Abstract

This paper considers the well-known Quadratic Assignment Problem (QAP) for the study. It is NP-hard combinatorial optimizations that can be defined as follows. There is *n* facilities and *n* locations. A *distance* is specified for each pair of locations, and a *flow* is specifiedfor each pair of facilities. The objective of problem is to allocate all facilities to different locations such that the sum of the flows multiplied by the corresponding distances is minimized. We develop a data-guided lexisearch algorithm based on an existing reformulation to find exact solution to the problem. For this we first modify alphabet table according to the number of zeros in the rows of the surplus matrix, thus, renaming rows (facilities), and then we apply lexisearch algorithm. It is shown that before applying lexisearch algorithm, this minor preprocessing of the data improves computational time significantly. Finally, we present a comparative study between data-guided lexisearch algorithm and two existing algorithms on some QAPLIB instances of various sizes. The computational study shows the effectiveness of our proposed data-guided lexisearch algorithm.

Keywords: Alphabet Table, Bound, Data-guided Lexisearch, Quadratic Assignment Problem, Surplus Matrix

1. Introduction

Koopmans and Beckmann[1] introduced the Quadratic Assignment Problem (QAP) for the first time. 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 $a(i)$ is the location of the facility i. The objective of problem is to allocate each facility to exactly one location such that the following total cost is minimized.

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

The QAP is proved to be NP-hard problem and it is one of most difficult combinatorial optimization problems[2]. It has many applications in real-life[3–13]. Due to its real-life application and difficulty, several exact and heuristic algorithms have been developed by many researchers for solving the problem. It is so hard that medium sized instances cannot be solved optimally by an exact algorithm in reasonable time. However, there are some situations where only exact solution is required, for example, placing circuits in a VLSI chip, assigning storage facilities in some location of very large plants that is done once in lifespan for an organization. We, thus, aim to find exact solution to the problem.

The ‘complete enumeration approach’ of solving the QAP is to list all the feasible solutions, evaluate their objective function values, and pick out the best. But, as the number of possible solutions to the QAP is huge, this approach is obviously inefficient and impracticable even for the small sized problem instances, and computational time grows exponentially with the problem size. Quite a few special exact algorithms have been developed, which can solve the problem much more efficiently than this approach. Branch-and-bound[14], Branch-and-cut[15], lexisearch[16] are well-known exact algorithms for solving the QAP. However, it is observed that as the problem size increases using exact method to find exact solution is very difficult, if not impossible.

**Author for correspondence*

The lexisearch algorithm has been effectively applied to many other combinatorial optimization problems[17–20]. In lexisearch algorithm, leader bound plays a vital role in reducing search space, hence, reduce the computational time. Also, before applying the lexisearch, pre-processing of data can reduce the computational effort significantly[18, 19]. In this paper, we apply a data-guided lexisearch algorithm that incorporates a data processing method to find exact solution to the QAP. The effectiveness of our data-guided algorithm against a simple lexisearch algorithm[21] and a discrete linear reformulation[22] have been examined for some QAPLIB instances[23] of different sizes.

We organize the paper as follows: Section 2 gives a literature survey on the QAP. Section 3 presents formulation of the problem by Ahmed[21]. A data-guided lexisearch algorithm is developed for the problem in Section 4. Computational experiment is reported in Section 5. Finally, Section 6 reports comments and concluding remarks.

2. Literature Survey

Numerous methods in the literature are used to find exact solution to the QAP and other combinatorial optimization problems. However, only a few instances of size n≥30 from QAPLIB have been solved optimally and most of them are solved using computers connected in parallel. Out of various methods, many researchers have proposed different branch and bound algorithms for solving the QAP. Branch and bound algorithms are defined from allocation and cutting rules that define lower bounds for the problem. Enumerative schemes using lower bounds to eliminate undesired solutions are developed by Gilmore[24]. Other literatures that use branch and bound algorithms are Lawler[25], Burkard and Derigs[26], Pardalos and Crouse[27], Pardalos et al.[28], Brixius and Anstreicher[29], Hahn et al.[30], etc.

Several reformulations of the QAP as integer or Mixed-Integer Linear Programming Problems (MILPs) have been proposed, which are then solved using different methods. Some literatures studied on special cases of QAP. Christofides and Benavent[31] studied a special case of the QAP that considers the flow matrix as the adjacency matrix of a tree using a MILP approach, which is then, solved using dynamic programming. A similar technique was also used by Urban[32]. Adams and Johnson[33] proposed a level-1 Reformulation-Linearization Technique (RLT-1) for obtaining lower bounds to the problem and found very good bounds. Erdoğan and Tansel[15] proposed two integer programming formulations constructed on the flow-based linearization techniques, which are then used to solve some instances up to size 25 (nug24, chr25a) using a depth-first branch-and-cut algorithm. Adams et al.[34] developed exact solution methods using a RLT-2 formulation for the problem and found solutions up to size of 30. As reported, among the RLT formulations, RLT-2 provides very tighter lower bound that leads to very close exact solution to the problem. Zhang et al.[35] proposed integer programming formulation of the problem and solved some instances up to size of 32 (esc32e-g) using CPLEX 9.0 and found very good results. Hahn et al.[36] proposed another RLT (RLT-3) formulation for calculating lower bounds for the QAP instances. As reported, experimental results project significant runtime improvement over all other published QAP branch-and-bound solvers. Nyberg and Westerlund[22] presented an exact discrete linear formulation of the problem and solved the problem using Gurobi (4.0.1) with default parameter settings. As reported they could solve some instances of size up to 64 (esc64a, tai64a).

The lexicographic search (lexisearch, for short) was developed by Pandit[37], first for Loading Problem (known as Knapsack Problem). It is a systematic branch and bound approach that was developed before branch and bound approach of Little et al.[38]. A lexisearch algorithm was developed to find exact solution to the QAP[16]. However, no any computational experiment was reported. Recently, a reformulation of the QAP has been proposed by Ahmed[21] such that lexisearch algorithm can be applied efficiently. The comparative study of the lexisearch algorithm against implementation of MILP formulation (IPQAPR-IV, therein) by Zhang et al.[35] shows the effectiveness of the proposed lexisearch algorithm based on the reformulation. We are going to use the reformulation by Ahmed[21] and then solve by using a data-guided lexisearch algorithm. In the next section we briefly discuss reformulation by Ahmed[21] with an example.

3. Areformulation by Ahmed[21]

Let $f'_{ij} = f_{ij} - u_i - v_j$ *and* $d'_{ij} = d_{ij} - x_i - y_j$. Then Z_a in equation (1) becomes

$$Z_a = \sum_{i=1}^{n}\sum_{j=1}^{n} f'_{ij} d'_{a(i)a(j)} + s_a \qquad (2)$$

where,

$$u_i = \min_{1 \le j \le n} f_{ij}, \text{ for } i = 1,2,3,\ldots\ldots,n \quad (3a)$$

$$v_j = \min_{1 \le i \le n} (f_{ij} - u_i), \text{ for } j = 1,2,3,\ldots\ldots,n \quad (3b)$$

$$x_i = \min_{1 \le j \le n} d_{ij}, \text{ for } i = 1,2,3,\ldots\ldots,n \quad (3c)$$

$$y_j = \min_{1 \le i \le n} (d_{ij} - x_i), \text{ for } j = 1,2,3,\ldots\ldots,n \quad (3d)$$

$$\alpha_i = \sum_{j=1}^{n} f_{ij}, \text{ for } i = 1,2,3,\ldots\ldots,n \quad (3e)$$

$$\beta_j = \sum_{i=1}^{n} f_{ij}, \text{ for } j = 1,2,3,\ldots\ldots,n \quad (3f)$$

$$\gamma_i = \sum_{j=1}^{n} d'_{a(i)a(j)}, \text{ for } i = 1,2,3,\ldots\ldots,n \quad (3g)$$

$$\delta_j = \sum_{i=1}^{n} d'_{a(i)a(j)}, \text{ for } j = 1,2,3,\ldots\ldots,n \quad (3h)$$

and s_a is the assignment cost with respect to a surplus matrix $S=[s_{ij}]$ with

$$s_{ij} = \alpha_i x_j + \beta_i y_j + u_i \gamma_j + v_i \delta_j \quad (3i)$$

Further Z_a can be reduced to the following form.

$$Z_a = \sum_{i=1}^{n} \sum_{j=1}^{n} f'_{ij} d'_{a(i)a(j)} + s'_a \quad (4)$$

Table 1. The flow matrix F

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

Table 2. The distance matrix D

Location	1	2	3	4	5
1	X	1	1	2	5
2	1	X	4	1	2
3	1	2	X	1	3
4	2	1	1	X	5
5	3	2	2	1	X

where,

$$\phi_i = \min_{1 \le j \le n} s_{ij}, \text{ for } i = 1,2,3,\ldots\ldots,n \quad (5a)$$

$$\psi_j = \min_{1 \le i \le n} (s_{ij} - \phi_i), \text{ for } j = 1,2,3,\ldots\ldots,n \quad (5b)$$

$$s'_{ij} = s_{ij} - \phi_i - \psi_j, 1 \le i, j \le n \quad (5c)$$

and

$$s_a = \sum_{i=1}^{n} \phi_i + \sum_{j=1}^{n} \psi_j + s'_a = c_s + s'_a \quad (5d)$$

In equation (5d), c_s is a constant and s'_a is the assignment cost with regard to the reduced surplus matrix S'. The matrix S' is a non-negative matrix. So, it is sufficient to minimize Z_a in equation (4). For example, let flow and distance matrices are presented in Table 1 and Table 2 respectively, then the corresponding matrices F', D' and S' are calculated and shown in Table 3 to Table 5 [21]. We have c_s=46.

Table 3. The reduced flow matrix F′

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

Table 4. The reduced distance matrix D′

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

Table 5. The reduced surpluss 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

It is to be noted that Edwards[39], and Frieze and Yadegar[41] proposed similar decombination methods to reduce the quadratic coefficients c_{ij}, d_{pq} into $\bar{c}_{ij}$, $\bar{d}_{pq}$ and then applied Gilmore-Lawler method[24, 25] to obtain lower of the QAP instances. However, our method is different from their methods. We are applying lexisearch algorithm which obtains exact solution, whereas their methods give only lower bound that may not be equal to the exact solution.

4. A Data-guided Lexisearch Algorithm

It is reported that in terms of computational times for the same size of instances, simple lexisearch algorithm produces two groups; one takes very low computational time, whereas other takes very high computational time[21]. In the simple algorithm, the nature of the data does not play any role. However, a preliminary scrutiny of the data can suggest some simple preprocessing, after which the algorithm becomes considerably more effective. Ahmed[18] developed a data-guided algorithm by transposing the cost matrix depending variances of rows and columns and then applied to the traveling salesman problem, and found better performance of the algorithm. But, for our problem, modifying the 'alphabet table' according to the variances of rows and columns cannot be applicable. Ahmed[19] developed another data-guided algorithm by modifying 'alphabet table' and applied to the bottleneck traveling salesman problem, which can be applicable to our problem. So, we modify the 'alphabet table' according to the number of zeros in the rows of the 'alphabet table'. We rename the facilities (rows) of the 'alphabet table' and accordingly create a new alphabet table and then apply the simple lexisearch algorithm.

4.1 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 when they are arranged in the non-decreasing order of their values[21]. Alphabet table "$[t(i,j)-s'_{i,t(i,j)}]$" is the mixture of elements of matrix T and their values (Ahmed, 2010a). The alphabet table for the matrix S' is shown in Table 6.

4.2 Lower Bound

The lower bound that is considered in lexisearch algorithm is the bound for leader block. We consider the same lower bound that is used by Ahmed[21]. However, we describe briefly the lower bound. As there are two terms in the equation (4), we have two calculations for the lower bound. For the first term, we sort row-wise elements of F' excluding diagonals in ascending order and store in $F''=[f''_{ij}]$, sort column-wise elements of D' excluding diagonals in descending order and store in $D''=[d''_{ij}]$, and then form an inner product matrix M as follows.

$$m_{ij} = \sum_{k=1}^{n-1} f''_{ik} d''_{kj} \tag{6}$$

Thereafter, elements of M are sorted row-wise in ascending order, which are shown in Table 7.

Now, suppose the location $a(k)$ is selected for concatenating to an incomplete assignment $\{a(1), a(2), \ldots\ldots, a(k-1)\}$. Before concatenation, we must check the bound for the leader $\{a(1), a(2), \ldots\ldots, a(k-1), a(k)\}$. We calculate the lower bound for the leader as follows:

$$L_k = \sum_{i=k+1}^{n} m_{ip} + \sum_{i=k+1}^{n} s'_{i,t(i,p)} \tag{7}$$

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

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

$$Z_k = \sum_{i=1}^{k}\sum_{j=1}^{k} f'_{ij} d'_{a(i)a(j)} + \sum_{i=1}^{k} s'_{i,a(i)} \tag{8}$$

Table 6. The alphabet table (P and V are the location and its value respectively)

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 7. 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

4.3 Modified Alphabet Table

We modify the 'alphabet table' according to the following rule. If all elements of the reduced surplus matrix S' are zero, then consider the matrix M, otherwise consider reduced surplus matrix S' for constructing 'alphabet matrix', and then form the 'alphabet table'. Now, interchange the rows (facilities) of the existing 'alphabet table' so that the rows with maximum zeros are shifted to the bottom while rows with minimum zeros are shifted to the top. In the event of a tie, the first positive values of the locations in the rows are compared, the rows containing largest values are shifted to the top and the rows containing smallest values are shifted to the bottom. The modified alphabet table after preprocessing the existing alphabet table (Table 6) is shown in Table 8.

4.4 The algorithm

Our data-guided lexisearch algorithm can be stated as follows. This algorithm is a modification of the simple lexisearch algorithm for the QAP[21].

Step 0:- Form the 'modified alphabet table'. Initialize the 'best solution value'(Z_a)as big as possible, $k = 1$, and$Z_{k-1} = 0$.

Step 1:- Let the present leader be the assignment of length (k-1) and the first 'legitimate' (i.e., unassigned and unchecked) location in k^{th} row of the alphabet table be the next location with value V. If $((V + Z_{k-1}) \geq Z_a)$, go to step 4, *else*, calculate Z_k (the value of present assignment) and L_k (lower bound for the present leader), and go to step 2. If we do not find any 'legitimate' location in the k^{th} row, go to step 4.

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

Table 8. The modified alphabet table

Facility	P–V	P–V	P–V	P–V	P–V
1(1)	4–0	2–1	1–2	3–3	5–15
2(4)*	1–0	2–0	3–0	4–0	5–8
3(2)	1–0	2–0	3–0	4–0	5–7
4(5)	1–0	2–0	3–0	4–0	5–3
5(3)	1–0	2–0	3–0	4–0	5–0

*Note: The indices in the brackets are the original names given to the facilities, while the indices without parenthesis are new indices, for example, facility that was indexed as 4 is now indexed as 2.

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

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

Step 5:- Z_a is the optimal solution value and the current assignment is the optimal assignment with respect to the facilities as described after preprocessing referred in step 0. Hence for getting the optimal assignment sequence in the required form, restore the facilities and stop.

4.5 Illustration of the Algorithm

Let us illustrate the working of the data-guided lexisearch algorithm using the example presented in Table 1 and Table 2. Let 'best solution value' (Z_a) = 9999. The 'search table' is given in Table 9, and the following symbols are used therein.

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

As seen from the search table, the optimal solution is given by the assignment $\begin{pmatrix} 1 & 4 & 2 & 5 & 3 \\ 4 & 3 & 2 & 1 & 5 \end{pmatrix}$ or equivalently the optimal assignment is {4, 2, 5, 3, 1} with value (cost) $Z_a = 4$. Hence, the optimal assignment cost with regard to the original matrices is $Z_a + C_s = 4 + 46 = 50$.

5. Computational Experience

We have encoded our data-guided lexisearch algorithm (DGLSA) in Visual C++ and run on the same machine used by Ahmed[21], i.e., on a Pentium IV personal computer with speed 3 GHz and 448 MB RAM under MS Windows XP, and tested with some medium sized QAPLIB instances[23]. To show the effectiveness of our DGLSA, a comparative study is carried out against simple Lexisearch Algorithm (LSA) of Ahmed[21]. In Table 10, Best Known solution (BKV) reported in QAPLIB; and Best Solution Value (BSV), percentage of error of the solution (Error(%)), Total computational Time (TotTime) and the computational time when the optimal solution is hit for the First Time (FirstTime) in seconds for solving the instances by

Table 9. The search table

Leaders					Z_k	L_k	Z_a	Remarks
1(1)	2(4)	3(2)	4(5)	5(3)				
4–0 (0)					0	0	9999	GS
	1–0 (0)				9	3	9999	GS
		2–0 (9)			9	0	9999	GS
			3–0 (9)		23	1	9999	GS
				5–0 (23)	26	0	9999	GS
							26	JO
			5–3 (12)		30	0	26	JB, JO
		3–0 (9)			9	0	26	GS
			2–0 (9)		19	1	26	GS
				5–0 (19)	25	0	26	GS
							25	JO
			5–3 (12)		34	0	25	JB, JO
		5–7 (16)			31	0	25	JB, JO
	2–0 (0)				0	0	25	GS
		1–0 (0)			9	0	25	GS
			3–0 (9)		17	1	25	GS
				5–0 (17)	32	0	25	JB, JO
			5–3 (12)		36	0	25	JB, JO
		3–0 (0)			0	0	25	GS
			1–0 (0)		1	1	25	GS
				5–0 (1)	7	0	25	GS
							7	JO
			5–3 (3)		17	0	7	JB, JO
		5–7 (7)					7	JO
	3–0 (0)				0	0	7	GS
		1–0 (0)			9	0	7	JB
		2–0 (0)			0	0	7	GS
			1–0 (0)		1	1	7	GS
				5–0 (1)	4	0	7	GS
							4	JO
			5–3 (3)		14	0	4	JB, JO
		5-7 (7)					4	JO
	5–8 (8)						4	JO
2–1 (1)					1	0	4	GS
	1–0 (1)				1	3	4	JB
	3–0 (1)				22	0	4	JB
	4–0 (1)				1	0	4	GS
		1–0 (1)			1	0	4	GS
			3–0 (1)		4	1	4	JB
			5–3 (4)				4	JO

Table 9. (*Continued*)

Leaders					Z_k	L_k	Z_a	Remarks
1(1)	2(4)	3(2)	4(5)	5(3)				
		3–0 (1)			20	0	4	JB
		5–7 (8)					4	JO
	5–8 (9)						4	JO
1–2 (2)					2	3	4	JB
3–3 (3)					3	0	4	GS
	1–0 (3)				3	3	4	JB
	2–0 (3)				18	0	4	JB
	4–0 (3)				3	0	4	GS
		1–0 (3)			3	0	4	GS
			2–0 (3)		4	1	4	JB
			5–3 (6)				4	JO
		2–0 (3)			20	0	4	JB
		5-7 (10)					4	JO
	5–8 (11)						4	JO
5–15(15)								STOP

Table 10. *Comparison of LSA and DGLSA*

Instance	BKV	LSA				DGLSA			
		BSV	Error(%)	FirstTime	TotTime	BSV	Error(%)	FirstTime	TotTime
esc16a	68	68	0.00	0.60	625.30	68	0.00	0.10	35.70
esc16b	292	292	0.00	29.10	14400.00	292	0.00	11.80	14400.00
esc16c	160	160	0.00	840.50	14400.00	160	0.00	0.00	2984.60
esc16d	16	16	0.00	2.50	14400.00	16	0.00	0.30	33.70
esc16e	28	28	0.00	0.56	14.30	28	0.00	0.00	0.30
esc16f	0	0	0.00	0.00	0.00	0	0.00	0.00	0.00
esc16g	26	26	0.00	0.00	0.50	26	0.00	0.00	0.10
esc16h	996	996	0.00	1.10	6228.20	996	0.00	0.00	1298.50
Partial Average			**0.00**	**109.30**	**6258.54**		**0.00**	**1.53**	**2344.11**
esc32a	130	198	52.31	12058.80	14400.00	142	9.23	1332.50	14400.00
esc32b	168	204	21.43	14093.60	14400.00	168	0.00	495.90	14400.00
esc32c	642	662	3.12	12339.60	14400.00	642	0.00	628.20	14400.00
esc32d	200	234	17.00	4931.80	14400.00	200	0.00	266.36	14400.00
esc32e	2	2	0.00	1.25	26.05	2	0.00	0.00	0.10
esc32f	2	2	0.00	2.03	25.32	2	0.00	0.00	0.10
esc32g	6	6	0.00	0.44	0.45	6	0.00	0.00	8.33
esc32h	438	574	31.05	4099.20	14400.00	460	5.02	2057.66	14400.00
Partial Average			**15.61**	**5940.84**	**9006.48**		**1.78**	**597.58**	**9001.07**

(*Continued*)

Table 10. *(Continued)*

Instance	BKV	LSA				DGLSA			
		BSV	Error(%)	FirstTime	TotTime	BSV	Error(%)	FirstTime	TotTime
kra30a	88900	118820	33.66	3874.31	14400.00	107350	20.75	5118.91	14400.00
kra30b	91420	118930	30.09	4099.42	14400.00	115870	26.74	13924.61	14400.00
kra32	88700	115310	30.00	6170.01	14400.00	101890	14.87	14284.54	14400.00
Partial Average			**31.25**	**4714.58**	**14400.00**		**20.79**	**11109.35**	**14400.00**
scr12	31410	31410	0.00	0.14	0.27	31410	0.00	0.20	0.20
scr15	51140	51140	0.00	77.88	81.70	51140	0.00	36.70	39.10
scr20	110030	118568	7.76	12986.42	14400.00	110030	0.00	6266.00	7886.45
Partial Average			**2.59**	**4354.81**	**4827.32**		**0.00**	**2100.97**	**2641.92**
Total Average			**10.29**	**3436.78**	**8172.82**		**3.48**	**2019.26**	**6449.42**

the algorithms, have been reported. The error (%) is given by the formula $Error(\%) = (BSV - BKV) / BKV \times 100\%$.

Results obtained by the algorithms for twenty two instances of sizes from 12 to 32 have been reported in Table 10. Out of them LSA and DGLSA hit optimal solution to thirteen and seventeen instances respectively within four hours of computational time. However, for three instances optimality could not be proved by DGLSA. On average, DGLSA obtains solutions which are 3.48% away from the optimal solutions, whereas, LSA obtains solutions which are 10.29% away from the optimal solutions. So, DGLSA obtains better solutions. In terms of computational time also, DGLSA is found to be better. It is to be noted that lexisearch algorithm first finds an optimal solution and then proves the optimality of that solution. The table shows that, on average computational time, LSA found optimal solution within at least 42% of the total computational time, whereas DGLSA found the optimal solution within only 31% of the total computational time. That is, LSA spent 58% and DGLSA spent 69% of total computational time on proving optimality of the solutions. So, LSA spends a comparatively large amount of time on finding an optimal solution for these QAPLIB instances compared to DGLSA, and hence, many sub problems are thrown by DGLSA. On the basis of computational time, DGLSA is found to be better than LSA. There is large improvement of DGLSA over LSA for the instances. So, our goal is achieved very well.

In Table 11, we also present another comparative study between our DGLSA and implementation of Discrete Linear Reformulation (DLR) by Nyberg and Westerlund[22] for thirteen QAPLIB instances. The table reports computational times (in seconds), and solutions as were reported by Nyberg and Westerlund[22] on a PC with Intel i7 4-core 2.8 GHz processor and 6 GB RAM and on another PC with Intel i7 6-core 3.2 GHz processor for esc64a using Gurobi (4.0.1) with default parameter settings. So, as regards the computational time, it was not possible to compare them directly as they have been run in different machines, and the machines used by Nyberg and Westerlund[22] are much faster than our machine. From the table it is seen that our algorithm could not hit optimal solution for esc32a and tai64c within four hours of computational time. For the remaining eleven instances, if we consider First Time for our DGLSA, then our algorithm is found to be far better than the DLR. However, among these eleven instances, for four instances our algorithm could not prove optimality of the solution within four hours. At least for the instances of size 12, our algorithm is found to be better. It means that our data-guided lexisearch algorithm can compete with stat-of-art methods in the literature. Also, solution by DGLSA does not rely on commercial math software, whereas solution by DLR relies on Gurobi.

7. Conclusion

We have developed a data-guided lexisearch algorithm to find exact solution to the Quadratic Assignment Problem (QAP). Depending on the number of zeros in the rows of 'alphabet table', we renamed the rows and constructed a new alphabet table. Next, the simple lexisearch algorithm of Ahmed[21] using new alphabet table

Table 11. *Comparison of DRL and DGLSA*

Instance	Size	BKV	Computational Time (in seconds)		
			DGLSA		DLR
			FirstTime	TotTime	TotTime
nug12	12	578	0.77	3.58	59.00
scr12	12	31410	0.20	0.21	9.60
chr12a	12	9552	0.02	0.02	1.60
tai12a	12	224416	4.94	28.13	246.00
rou12	12	235528	12.31	52.16	1187.00
esc16a	16	68	0.10	35.70	11.40
esc16b	16	292	11.80	14400.00	158.00
esc16c	16	160	0.00	2984.60	286.00
esc32a	32	130	–	–	1618580.00
esc32c	32	642	628.20	14400.00	24365.00
esc32d	32	200	266.36	14400.00	36256.00
esc64a	64	116	2.30	14400.00	16370.00
tai64c	64	1855928	–	–	182983.00

is applied. It is shown that before applying the simple lexisearch algorithm, preprocessing of the data improves the computational time as well as solution quality significantly. Finally, the performance of the data-guided lexisearch algorithm is compared with implementation of the Discrete Linear Reformulation (DLR) by Nyberg and Westerlund[22] for some medium sized QAPLIB instances. Among the algorithms, our data-guided algorithm is found to be the better than the simple algorithm, and the data-guided algorithm is competing with DLR using Gurobi.

We have investigated using lexisearch algorithm that only some medium sized instances can be solved optimally within stipulated time limit. For the large sized instances the lexisearch algorithm is not found to be suitable. Also, for some small sized instances, for example esc16b of size 16, our algorithm could not prove optimality of the solution; whereas the instancesesc32e-gof sizes32 could be solved within 8.33seconds only. Also, surprisingly, our algorithm could hit the optimal solution for esc64a of size 64 within 2.3 seconds. 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, a more sophisticated data-guided approach may be used to reduce the computational time further and to find better optimal solution quickly. Also, one can propose a tighter lower bound method which a very important part of the lexisearch algorithm that may find optimal solution for some more instances quickly. Further, combination of lexisearch and genetic algorithm[41] may lead to an efficient way of solving the problem.

8. Acknowledgement

The author is thankful to the respected reviewer for his constructive comments and suggestions. This work has been supported by Deanery of Academic Research, Al Imam Mohammad Ibn Saud Islamic University, Saudi Arabia through Grant no. 320902. The author is very much thankful to the Deanery for its Financial Support.

9. References

1. Koopmans TC, Beckmann MJ. Assignment problems and the location of economic activities. Econometrica. 1957; 25(1):53–76.
2. Sahni S, Gonzales T. P-complete approximation problems. Journal of the Association for Computing Machinery. 1976; 23:555–65.
3. Steinberg L. The backboard wiring problem: a placement algorithm. SIAM Rev. 1961; 3(1):37–50.
4. Geoffrion AM, Graves GW. Scheduling parallel production lines with changeover costs: practical applications of a quadratic assignment/LP approach. Operations Research. 1976 Jul–Aug; 24(4):595–610.

5. Pollatschek MA, Gershoni N, Radday YT. Optimization of the typewriter keyboard by simulation. Angewandte Informatik. 1976; 17:438–9.
6. Elshafei AN. Hospital layout as a quadratic assignment problem. Operations Research Quarterly. 1977; 28(1):167–79.
7. Krarup J, Pruzan PM. Computer-aided layout design. Math Program Stud. 1978; 9:75–94.
8. Heffley DR. Decomposition of the Koopmans–Beckmann problem. Reg Sci Urban Econ. 1980; 10(4):571–80.
9. Hubert LJ. Assignment methods in combinatorial data analysis. Statistics: Textbooks and Monographs Series. New York: Marcel Dekker, Inc. 1987. Book 73.
10. Bos J. A quadratic assignment problem solved by simulated annealing. Journal of Environmental Management. 1993; 37(2):127–45.
11. Forsberg JH, Delaney RM, Zhao Q, Harakas G, Chandran R. 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. 1994; 34:3705–15.
12. Brusco MJ, Stahl S. Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. J Classif. 2000; 17(2):197–223.
13. Duman E, Ilhan O. The quadratic assignment problem in the context of the printed circuit board assembly process. Comput Oper Res. 2007; 34:163–79.
14. Hahn P, Grant T, Hall N. A branch-and-bound algorithm for the quadratic assignment problem based on Hungarian method. Eur J Oper Res. 1998; 108:629–40.
15. Erdoğan G, Tansel, B. A branch-and-cut algorithm for the quadratic assignment problems based on linearizations. Comput Oper Res. 2007; 34(4):1085–106.
16. Das S. Routing and Allied Combinatorial Programming Problems: A Lexicographic Search Approach [Ph.D. Thesis]. Assam, India: Dibrugarh University; 1976.
17. Ahmed ZH. A lexisearch algorithm for the bottleneck traveling salesman problem. Int J Comput Sci Secur. 2010a; 3(6):569–77.
18. Ahmed ZH. A data-guided lexisearch algorithm for the asymmetric traveling salesman problem. Mathematical Problems in Engineering. 2011a; 2011(2011). doi:10.1155/2011/750968.
19. Ahmed ZH. A data-guided lexisearch algorithm for the bottleneck travelling salesman problem. International Journal of Operational Research. 2011b; 12(1):20–33.
20. Ahmed ZH. An exact algorithm for the clustered travelling salesman problem. OPSEARCH. 2013a Jun; 50(2):215–28.
21. Ahmed ZH. A new reformulation and an exact algorithm for the quadratic assignment problem. Indian Journal of Science and Technology. 2013b; 6(4):4368–77.
22. Nyberg A, Westerlund T. A new exact discrete linear reformulation of the quadratic assignment problem. Eur J Oper Res. 2012; 220:314–19.
23. Burkard RE, Karisch SE, Rendl F. QAPLIB - a quadratic assignment problem library. J Global Optim. 1997; 10(4):391–403.
24. Gilmore PC. Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM J Appl Math; 1962; 10:305–13.
25. Lawler EL. The quadratic assignment problem. Manag Sci. 1963; 19:586–90.
26. Burkard RE, Derigs U. Assignment and matching problems: solutions methods with Fortran programs. Lect Notes Econ Math Syst. 1980; 184.
27. Pardalos P, Crouse J. A parallel algorithm for the quadratic assignment problem. Supercomputing '89. Proceedings of the 1989 ACM/IEEE Conference on Supercomputing; 1989 Nov 12–17; Reno, NV, United States. p. 351–60.
28. Pardalos PM, Ramakrishnan KG, Resende MGC, Li Y. Implementation of a variance reduction-based lower bound in a branch-and-bound algorithm for the quadratic assignment problem. SIAM J Optim. 1997; 7(1):280–94.
29. Brixius NW, Anstreicher KM. Solving quadratic assignment problems using convex quadratic programming relaxations. Optim Meth Software. 2001; 16(1–4):49–68.
30. Hahn PM, Hightower WL, Johnson TA, Guignard-Spielberg M, Roucairol C. Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. Yugoslavian Journal of Operational Research. 2001; 11(1):41–60.
31. Christofides N, Benavent E. An exact algorithm for the quadratic assignment problem. Oper Res. 1989; 37(5):760–68.
32. Urban TL. Solution procedures for the dynamic facility layout problem. Ann Oper Res. 1998. 76:323–42.
33. Adams WP, Johnson TA. Improved linear programming-based lower bounds for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1994; 16:43–75.
34. Adams WP, Guignard M, Hahn PM, Hightower WL. A level-2 reformulation–linearization technique bound for the quadratic assignment problem. European Journal of Operational Research. 2007; 180(3):983–96.
35. Zhang H, Beltran-Royo C, Constantino M. Effective formulation reductions for the quadratic assignment problem. Comput Oper Res. 2010; 37:2007–16.
36. Hahn PM, Zhu Y-R, Guignard M, Hightower WL, Saltzman MJ. A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. INFORMS J Comput. 2012; 24(2):202–09.

37. Pandit SNN. Some quantitative combinatorial search problems [Ph.D. Thesis]. Kharagpur, India: Indian Institute of Technology; 1963.
38. Little JDC, Murthy KG, Sweeny DW, Karel C. An Algorithm for the Travelling Salesman Problem. Oper Res. 1963; 11:972–89.
39. Edwards CS. A branch and bound algorithm for the Koopmans-Bechmann quadratic assignment problem. Math Program Stud. 1980; 13:35–52.
40. Frieze AM, Yadegar J. On the quadratic assignment problem. Discrete Appl Math. 1983; 5(1):89–98.
41. Ahmed ZH. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. International Journal of Biometrics & Bioinformatics. 2010b; 3(6):96–105.