International Journal of Mechanical Engineering

CONSTRAINED LOAD BALANCING SOLID UNBALANCED ASSIGNMENT PROBLEM: AN EXACT ENUMERATIVE ALGORITHM

Sk.Mastan[1], U.Balakrishna[2], G. Sankar Sekhar Raju[3], T. Jayanth Kumar[4]

[1]Research Scholar, JNTUA, Ananthapuramu, India-515002.

[2]Professor, SITAMS, Chittoor, India-517127.

[3]Professor, JNTUA, Pulivendula,, India-516390.

[4]Assistant Professor, Jain (Deemed to be University), Kanakapura Rd, Bengaluru, Karnataka, India-562112.

ABSTRACT

In this paper, a practical variant of unbalanced assignment problem called Constrained Load Balancing Solid Unbalanced Assignment Problem [CLBSUAP]. The problem can be stated as follows: Let there be n persons, m jobs, p facilities(which is a third independent factor) such that the number of persons should be less than the number of jobs and the cost of performing jobs to persons at different facilities. It is a usual three-dimensional unbalanced assignment problem with the restrictions $m*$ $(\leq m)$ jobs out of m jobs are assigned to $n*$ $(\leq n)$ persons out of n persons and a job cannot be assigned to more than one person, and each of the $n*$ persons can do at least 1 and at most $\left\lceil \frac{m^*}{n^*} \right\rceil$ jobs under the given facilities with or without repetition with the objective that the overall cost on performing the jobs by the persons with the above restrictions is minimum. This problem is formulated using 0-1 integer Linear Programming (0-1 ILP). To deal with the CLBSUAP optimally, a pattern recognition strategy based Lexi-search approach (LSA) is developed. The effectiveness of the proposed LS approach to the CLBSUAP as against different approaches has been examined for a variety of randomly generated test problems.

KEYWORDS

Constrained Load Balancing Solid Unbalanced Assignment Problem, Lexi-search algorithm, Pattern recognition technique, 0-1 Integer linear programming.

1. INTRODUCTION

Assignment-related issues often occur in several engineering, medical and management areas. In reality, this is a well-studied topic in combinatorial optimization problems under the optimization or operations research branches. Furthermore, the problem of assignment is an essential subject that has been used to tackle numerous difficulties all around the world [1]. This issue has arisen frequently in numerous educational initiatives around the world. This study divided the assignment problem into two categories in the education domain: timetabling and allocation.

The investigation of how to allocate things to objects in the best feasible way (optimal way) is known as the assignment issue [2, 3]. The two components of the assignment problem are the matchings and the objective function. The matching denotes the essential combinatorial arrangement, whereas the goal function denotes the desire to be as efficient as feasible. Nonetheless, the challenge is "how to complete an assignment with the best purpose while also meeting all of the constraints?" Several approaches [1, 2] have been presented to answer the question, including the exact method [4, 10, 11, 13 & 14], the heuristics and metaheuristic methods [5,6,7,8, & 12].

The goal of investigating the assignment problem is to find a match between two or more sets of items that will reduce the overall cost of all matched pairs. Based on the specific structure of the matched sets and the cost function, the assignment models can be divided into quadratic, bottleneck, linear, and multidimensional groups [9]. As a result, there is a table or matrix in every assignment problem. The rows are usually made up of objects or persons to assign, whereas the columns are made up of things or jobs to allocate. Meanwhile, the figures in the table represent the costs associated with each individual assignment. The following is the problem that motivated the research described in this paper: student instructors at educational institutions are required to engage in instructional activities on a regular basis. Each group of children will be assigned to one of many schools, with a college tutor overseeing their practices. The fact that determining which school a student will be assigned to and which tutor will be in charge of a student's monitoring is difficult.

You should try to satisfy the requirements and preferences of both students and tutors as much as feasible. Formally Let there are m students, n tutors and p schools. Let I={1,2,3,….,n}, J={1,2,3,….m}, K={1,2,3,….p}. For student i∈ I, j∈ J and school k∈ K let S_{ijk} be a satisfaction value associated with an assignment of student i to school k under super vision by tutor j. Tutor j is willing to supervise no more than t_j students for $j\in J$ and school $k\in K$ can have atmost s_k students assigned to it.

The following is an overview of how this paper is organized: The CLBSUAP's problem statement and mathematical model are detailed in Section 2. The proposed algorithm for resolving the CLBSUAP is described in Section 3. In Section 4, there is an example. Comparable data can be found in Section 5 followed by conclusions in Section 6.

2. PROBLEM STATEMENT AND FORMULATION

The CLBSUAP can be described accordingly:

Let I={1,2,3,….,n}, J={1,2,3,…,m}& K={1,2,3,….p} be the set of n persons, m jobs & p facilities respectively and a non-negative integer C_{ijk} indicates the cost to complete j^{th} job by i^{th} person at k^{th} facility. Let $n^*(<n)$ out of n persons are to be assigned among m^* (<m) jobs such that each person must perform at least 1 and atmost $\left\lceil\frac{m^*}{n^*}\right\rceil$ number of jobs. A person is allowed to do more than one job using same facility, but one after another. The problem CLBSUAP seeks assignment of m* jobs to n^* persons such that atleast one and atmost $\left\lceil\frac{m^*}{n^*}\right\rceil$ jobs are to be done by n^* persons under the given facilities with the objective that the overall cost on performing the given jobs by the persons is minimum.

The CLBSUAP model is formulated using the following assumptions:

- $n^* < m^* < m$
- All the persons has to start working at the same time
- The same job cannot be done by multiple persons
- A person can do multiple jobs with same or distinct facilities
- The elements in the cost matrix take arbitrary values

Using the above assumptions, the CLBSUAP model is developed as 0-1 ILP:

$Min\left(\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} C_{ijk}\, X_{ijk}\right)$	(1)
Subject to	
$\sum_{i\in I}\sum_{k\in K} X_{ijk} = 1, j \in J$	(2)
$1 \le \sum_{j\in J}\sum_{k\in K} X_{ijk} \le \left\lceil\frac{m^*}{n^*}\right\rceil, i \in I$	(3)
$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} X_{ijk} = m^*$	(4)
$X_{ijk} = \{0,1\}; IXJXK = \{(i,j,k)\}$	(5)

In this model, (1) is an objective function which minimizes the cost of performing m^*jobs by n^* machines under p facilities. Constraint (2) represents each job can be assigned to one person under a facility. Constraint (3) represents atleast 1 and atmost $\left\lceil\frac{m^*}{n^*}\right\rceil$ jobs must be done by m* persons. Constraint (4) denotes m* jobs are to be done by n^*persons under p facilities. Finally, In Constraint (5),X_{ijk} takes 1 if j^{th} job is assigned to i^{th} person under k^{th} facility and X_{ijk}assumes 0, otherwise.

3. PRELIMINARIES OF LSA

3.1 Feasible Solution

If CLBSUAP meets all of the constraints of (2-6), the solution is said to be feasible.

3.2 Pattern

A pattern associated with an assignment is referred to as a three-dimensional array. If a pattern is a feasible solution, its value can be determined using (7), which gives the total assignment cost, which is the same as the value of the objective function.

$$V(X) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} C_{ijk}X_{ijk} \qquad (7)$$

3.3 Alphabet Table

The elements of cost matrix $C = [C_{ijk}]$ in an alphabet table are arranged in increasing order and named from 1 to $nxmxp$. Let SR= {1, 2, 3,…, $nxmxp$} be the collection of $nxmxp$ well-organized indices, arrays. t and Ct respectively, express the cost and its cumulative sums of C. Let the arrangements row (R), column (Cl) and Time(T) are the indices of SR. This table includes the ordered index set as the table of alphabets, such as SR, t, Ct, R, C & T. Let $L = \{S_1, S_2, .., S_r\}$, $S_i \in SR$ be a pattern whose elements are free from the order in the sequence. To be unique, the elements of SR are arranged in such a way that $S_i < S_{i+1}, i = 1, 2, 3, ...r-1$.

3.4 Word, partial word and value of a word ($V(L)$)

A systematic sequence $L = \{S_1, S_2, .., S_r\}$ is said to be a word of length r. If $r < n^*$, then the word L is called a partial feasible word and it is the full-length word when $r = n^*$. Any index in SR can be the prime position in L. A partial word L describes a set of words as the leader, with L. If the word block described by it contains at least one possible word, the leader is feasible; otherwise, it is infeasible.

The value of the word L is denoted by $V(L)$ and is calculated using $V(L_{s-1}) + C(S_r)$ with $V(L_0) = 0$ and obviously, $C(S_r)$ be the cost array whose elements are arranged in a way such that $C(S_r) \leq C(S_{r-1}), r = 1,2,3, \ldots, mxnxp$. $V(L)$ is similar to $V(\mathrm{X})$ value (Sundara Murthy).

3.5 Calculation of bounds

Setting appropriate upper and lower bounds is critical for dealing with the NP-hard problem effectively. Initially, the UB of is claimed to be a very large value as an experimental solution $(UB = TS = 999999)$. For block values of words represented by, the lower bound LB(L_s) can be defined as follows::

$$LB(L_S) = V(L_S) + Ct(S_k + n^* - k) - Ct(S_k)$$

3.6 Lexi Search Method:

In dealing with combinatorial optimization models, optimal solutions obtained through exact search methods have become more attractive. It can be shown that the full and implicit search methods use the same approaches. The Branch and Bound(B & B) strategy is one of the most frequent implicate searching approaches.

The LSA is one such implicated method of enumeration, in which only a fragment of the solution space is explored and optimal solutions are provided consistently (Pandit, 1962).

In truth, B & B can be considered a unique LSA situation. The LSA protects all B & B elements, including the design of feasible solutions, viability testing, and determining the limitations of a partially viable solution.

The entire search procedure is carried out in a precise fashion that resembles a dictionary search for the definition of a term, hence the name "Lexi-search." This systemic hunt also protects against stack overflow and search time.

The fundamental problem with complete search methods is determining I whether the answer is viable and (ii) establishing appropriate bounds. Checking the feasibility is quite important for a few difficulties. A Lexi-search strategy was devised and defined based on the pattern-recognition technique (Murthy, 1976) to efficiently cope with this problem:

"Each problem solution is based on a distinct pattern. A partial solution is the same as a partial pattern. The words that represent the pattern in lexicographic or dictionary order are used to describe an alphabet table. When a partial word is employed, the first boundaries are specified, and then the areas for which the value is lower than the trail's meaning are tested for viability when choosing the ideal term ".

3.7 Proposed LSA

The following are the steps involved in LSA:

Step 1: **Input**

Cost matrix $C = [C_{ijk}], n, m, p, n^*, m^*, UB = TS = 999999$ & SJ^* and then Step 2.

Step 2: Create an alphabet table for the Cost matrix C, and then go to Step 3.

Step 3**: Setting the Bounds**

The procedure begins with a partial word $L = (S_k) = 1; S_k \in SR$ which is a single word length, i.e. $k = 1$.

Compute $LB(L)$. If the $LB(L) < \mathrm{TS}$, then move to step 5, else go to Step 4.

Step 4: If the $LB(L) \geq \mathrm{TS}$, then delete the partial word L and suspend the group of words with L as a leader and go to Step 7.

Step 5: **Feasibility verification**

It is said to be feasible if the partial word follows the constraints; otherwise, it is said to be infeasible. If L is feasible, then accept it and continue for next partial word of order $k+1$ and go to Step 6, else proceed with the next partial word of order k by considering another letter that succeeds S_k in its k^{th} position and go to Step 3.

Step 6: **Concatenation**

If L is a full-length feasible word (i.e. $k = n^*$), then store $LB(L)$ value in TS and go to Phase 8. If L is a partial term, it can be concatenated by using $L(S_{k+1}) = L(S) * (S_{k+1})$, where * indicates the operation of concatenation and goes to step 3.

Step 7: If the word length is 1 and all of the order terms have been met, the search procedure is complete, and you can go to Step 9.

Step 8: **Backtracking**

Backtracking is used to explore the search space; the current is assumed to be an upper bound, and the search proceeds with the next letter of the partial word of order (see Step 3). Steps 3–8 should be repeated until there is no more change, and any feasible/infeasible options that are not in the optimal solution should be ignored. Continue to step 9.

Step 9: Move to Step 10 after saving the current word.

Step 10: Stop

4. Numerical illustration

To validate the algorithm, we considered $\mathrm{n} = 4,\ \mathrm{m} = 6, \mathrm{p} = 2, \mathrm{n}^* = 2, \mathrm{m}^* = 4$. The cost matrix $C = C_{ijk}$ of CLBSUAP is shown below.

Table-1

$$C(i,j,1) = \begin{bmatrix} 4 & 7 & 4 & 32 & 10 & 21 \\ 5 & 13 & 32 & 6 & 1 & 16 \\ 8 & 42 & 11 & 1 & 23 & 3 \\ 15 & 7 & 3 & 60 & 4 & 12 \end{bmatrix},\ C(i,j,2) = \begin{bmatrix} 35 & 4 & 12 & 3 & 4 & 6 \\ 5 & 16 & 72 & 9 & 19 & 9 \\ 1 & 21 & 4 & 2 & 5 & 31 \\ 16 & 5 & 3 & 13 & 8 & 10 \end{bmatrix}$$

$C = C_{ijk}$ is taken in the numerical illustration in Table-1 as non-negative. In matrix $= [X_{ijk} / X_{ijk} \in \{0,1\}]$, $X_{ijk} = 1$ means the i^{th} person is assigned to j^{th} job at k^{th} facility and $X_{ijk} = 0$, otherwise. Here X is called a solution.

4.1 Alphabet – Table

Table 2 shows how to make an alphabet table for a cost matrix. The notations in Table 2 are as follows: SR, C, Cc, R, Cl & Ti respectively denotes the serial number, cost, cumulative cost, row, column and time of the indices.

Table - 2: ALPHABET TABLE

SR	*C*	*Cc*	*R*	*Cl*	*Ti*
1	1	1	2	5	1
2	1	2	3	4	1
3	1	3	3	1	2
4	2	5	3	4	2
5	3	8	3	6	1
6	3	11	4	3	1
7	3	14	1	4	2
8	3	17	4	3	2
9	4	21	1	1	1
10	4	25	1	3	1

11	4	29	4	5	1
12	4	33	1	2	2
13	4	37	1	5	2
14	4	41	3	3	2
15	5	46	2	1	1
16	5	51	2	1	2
17	5	56	3	5	2
18	5	61	4	2	2
19	6	67	2	4	1
20	6	73	1	6	2
21	7	80	1	2	1
22	7	87	4	2	1
23	8	95	3	1	1
24	8	103	4	5	2
25	9	112	2	4	2
26	9	121	2	6	2
27	10	131	1	5	1
28	10	141	4	6	2
29	11	152	3	3	1
30	12	164	4	6	1
31	12	176	1	3	2
32	13	189	2	2	1
33	13	202	4	4	2
34	15	217	4	1	1
35	16	233	2	6	1
36	16	249	2	2	2
37	16	265	4	1	2
38	19	284	2	5	2
39	21	305	1	6	1
40	21	326	3	2	2
41	23	349	3	5	1
42	31	380	3	6	2
43	32	412	1	4	1
44	32	444	2	3	1
45	35	479	1	1	2
46	42	521	3	2	1
47	60	581	4	4	1
48	72	653	2	3	2

4.2 Search Table

The logical flow of the created LSA is depicted numerically in Table 3.

Table –3: SEARCH TABLE (ST)

SN	1	2	3	4	V	LB	R	Cl	Ti	REM
1.	1				1	3	2	5	1	A
2.		2			2	4	3	4	1	A
3.			3		3	5	3	1	2	A
4.				4	5	5	3	4	2	R
5.				5	6	6	3	6	1	R
6.				6	6	6	4	3	1	R
7.				7	6	6	1	4	2	R
8.				8	6	6	4	3	2	R
9.				9	7	7	1	1	1	R
10.				10	7	7	1	3	1	R
11.				11	7	7	4	5	1	R
12.				12	7	7	1	2	2	R
13.				13	7	7	1	5	2	R
14.				14	7	7	3	3	2	R
15.				15	8	8	2	1	1	R
16.				16	8	8	2	1	2	R
17.				17	8	8	3	5	2	R
18.				18	8	8	4	2	2	R
19.				19	9	9	2	4	1	R
20.				20	9	9	1	6	1	R
21.				21	10	10	1	2	1	R
22.				22	10	10	4	2	1	R
23.				23	11	11	3	1	1	R
24.				24	11	11	4	5	2	R
25.				25	12	12	2	4	2	R
26.				26	12	12	2	6	2	A, VT=12
27.				21	11	18	1	2	1	R
28.			4		4	7	3	4	2	R
29.			5		5	8	3	6	1	A
30.				6	8	8	4	3	1	R
31.				7	8	8	1	4	2	R
32.				8	8	8	4	3	2	R
33.				9	9	9	1	1	1	R
34.				10	9	9	1	3	1	R
35.				11	9	9	4	5	1	R
36.				12	9	9	1	2	2	R
37.				13	9	9	1	5	2	R

38.				14	9	9	3	3	2	R
39.				15	10	10	2	1	1	A,VT=10
40.			6		5	8	4	3	1	R
41.			7		5	8	1	4	2	R
42.			8		5	9	4	3	2	R
43.			9		6	10	1	1	1	R,=VT
44.		3			2	5	3	1	2	A
45.			4		4	7	3	4	2	A
46.				5	7	7	3	6	1	R
47.				6	7	7	4	3	1	R
48.				7	7	7	1	4	2	R
49.				8	7	7	4	3	2	R
50.				9	8	8	1	1	1	R
51.				10	8	8	1	3	1	R
52.				11	8	8	4	5	1	R
53.				12	8	8	1	2	2	R
54.				13	8	8	1	5	2	R
55.				14	8	8	3	3	2	R
56.				15	9	9	2	1	1	R
57.				16	9	9	2	1	2	R
58.				17	9	9	3	5	2	R
59.				18	9	9	4	2	2	R
60.				19	10	10	2	4	1	R,=VT
61.			5		5	8	3	6	1	A
62.				6	8	8	4	3	1	R
63.				7	8	8	1	4	2	R
64.				8	8	8	4	3	2	R
65.				9	9	9	1	1	1	R
66.				10	9	9	1	3	1	R
67.				11	9	9	4	5	1	R
68.				12	9	9	1	2	2	R
69.				13	9	9	1	5	2	R
70.				14	9	9	3	3	2	R
71.				15	10	10	2	1	1	R,=VT
72.			6		5	8	4	3	1	R
73.			7		5	8	1	4	2	R
74.			8		5	9	4	3	2	R
75.			9		6	10	1	1	1	R,=VT
76.		2			2	4	3	4	1	A
77.			3		3	5	3	1	2	A
78.				4	5	5	3	4	2	R
79.				5	6	6	3	6	1	R
80.				6	6	6	4	3	1	R
81.				7	6	6	1	4	2	R
82.				8	6	6	4	3	2	R
83.				9	7	7	1	1	1	R
84.				10	7	7	1	3	1	R

85.				11	7	7	4	5	1	R
86.				12	7	7	1	2	2	R
87.				13	7	7	1	5	2	R
88.				14	7	7	3	3	2	R
89.				15	8	8	2	1	1	R
90.				16	8	8	2	1	2	R
91.				17	8	8	3	5	2	R
92.				18	8	8	4	2	2	R
93.				19	9	9	2	4	1	R
94.				20	9	9	1	6	2	R
95.				21	10	10	1	2	1	R,=VT
96.			4		4	7	3	4	2	R
97.			5		5	8	3	6	1	A
98.				6	8	8	4	3	1	R
99.				7	8	8	1	4	2	R
100.				8	8	8	4	3	2	R
101.				9	9	9	1	1	1	R
102.				10	9	9	1	3	1	R
103.				11	9	9	4	5	1	R
104.				12	9	9	1	2	2	R
105.				13	9	9	1	5	2	R
106.				14	9	9	3	3	2	R
107.				15	10	10	2	1	1	R,=VT
108.			6		5	8	4	3	1	R
109.			7		5	8	1	4	2	R
110.			8		5	9	4	3	2	R
111.			9		6	10	1	1	1	R,=VT
112.		3			2	6	3	1	2	A
113.			4		4	7	3	4	2	A
114.				5	7	7	3	6	1	R
115.				6	7	7	4	3	1	R
116.				7	7	7	1	4	2	R
117.				8	7	7	4	3	2	R
118.				9	8	8	1	1	1	R
119.				10	8	8	1	3	1	R
120.				11	8	8	4	5	1	R
121.				12	8	8	1	2	2	R
122.				13	8	8	1	5	2	R
123.				14	8	8	3	3	2	R
124.				15	9	9	2	1	1	R
125.				16	9	9	2	1	2	R
126.				17	9	9	3	5	2	R
127.				18	9	9	4	2	2	R
128.				19	10	10	2	4	1	R,=VT
129.			5		5	8	3	6	1	A
130.				6	8	8	4	3	1	R
131.				7	8	8	1	4	2	R

132.				8	8	8	4	3	2	R
133.				9	9	9	1	1	1	R
134.				10	9	9	1	3	1	R
135.				11	9	9	4	5	1	R
136.				12	9	9	1	2	2	R
137.				13	9	9	1	5	2	R
138.				14	9	9	3	3	2	R
139.				15	10	10	2	1	1	R,=VT
140.			6		5	8	4	3	1	R
141.			7		5	8	1	4	2	R
142.			8		5	9	4	3	2	R
143.			9		6	10	1	1	1	R,=VT
144.		4			3	6	3	4	2	A
145.			5		6	9	3	6	1	A
146.				6	9	9	4	3	1	R
147.				7	9	9	1	4	2	R
148.				8	9	9	4	3	2	R
149.				9	10	10	1	1	1	R,=VT
150.			6		6	9	4	3	1	R
151.			7		6	9	1	4	2	R
152.			8		6	10	4	3	2	R,=VT
153.		5			4	8	3	6	1	A
154.			6		7	10	4	3	1	R,=VT
155.		6			4	7	4	3	1	A
156.			7		7	10	1	4	2	R,=VT
157.		7			4	8	1	4	2	A
158.			8		7	11	4	3	2	R,>VT
159.		8			4	8	4	3	2	A
160.			9		8	12	1	1	1	R,>VT
161.		9			5	9	1	1	1	A
162.			10		9	13	1	3	1	R,>VT
163.		10			5	9	1	3	1	A
164.			11		9	13	4	5	1	R,>VT
165.		11			5	9	4	5	1	R
166.		12			5	9	1	2	2	A
167.			13		9	13	1	5	2	R,>VT
168.		13			5	10	1	5	2	R,>VT
169.	2				1	4	3	4	1	A
170.		3			2	5	3	1	2	A
171.			4		4	7	3	4	2	R
172.			5		5	8	3	6	1	R
173.			6		5	8	4	3	1	A
174.				7	8	8	1	4	2	R
175.				8	8	8	4	3	2	R
176.				9	9	9	1	1	1	R
177.				10	9	9	1	3	1	R
178.				11	9	9	4	5	1	A,VT=9

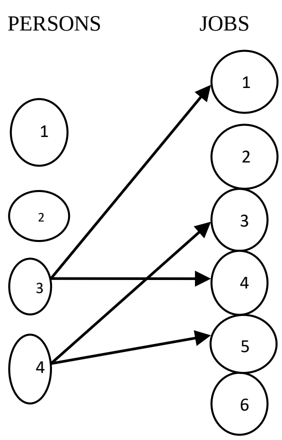
179.			7		5	8	1	4	2	A
180.				8	8	8	4	3	2	R
181.				9	9	9	1	1	1	R,=VT
182.			8		5	9	4	3	2	R,=VT
183.		4			3	7	3	4	2	R
184.		5			4	7	3	6	1	A
185.			6		7	10	4	3	1	R,>VT
186.		6			4	8	4	3	1	A
187.			7		7	10	1	4	2	R,>VT
188.		7			4	8	1	4	2	R
189.		8			4	8	4	3	2	A
190.			9		8	12	1	1	1	R,>VT
191.		9			5	9	1	1	1	R,=VT
192.	3				1	6	3	1	2	A
193.		4			3	6	3	4	2	A
194.			5		6	9	3	6	1	R,=VT
195.		5			4	7	3	6	1	A
196.			6		7	10	4	3	1	R,>VT
197.		6			4	7	4	3	1	A
198.			7		7	10	1	4	2	R,>VT
199.		7			4	8	1	4	2	A
200.			8		7	11	4	3	2	R,>VT
201.		8			4	8	4	3	2	A
202.			9		8	12	1	1	1	R,>VT
203.		9			5	9	1	1	1	R,=VT
204.	4				2	7	3	4	2	A
205.		5			5	8	3	6	1	A
206.			6		8	11	4	3	1	R,>VT
207.		6			5	8	4	3	1	A
208.			7		8	11	1	4	2	R,>VT
209.		7			5	9	1	4	2	R,=VT
210.	5				3	6	3	6	1	A
211.		6			6	9	4	3	1	R,=VT
212.	6				3	7	4	3	1	A
213.		7			6	10	1	4	2	R,>VT
214.	7				3	7	1	4	2	A
215.		8			6	10	4	3	2	R,>VT
216.	8				3	7	4	3	2	A
217.		9			7	11	1	1	1	R,>VT
218.	9				4	8	1	1	1	A
219.		10			8	12	1	3	1	R,>VT
220.	10				4	8	1	3	1	A
221.		11			8	12	4	5	1	R,>VT
222.	11				4	8	4	5	1	A
223.		12			8	12	1	2	2	R,>VT
224.	12				4	9	1	2	2	R,=VT

In the end, the value of TS is 21 lies in 178th row of the table and its corresponding word is given by $L_4 = (2,3,6,11)$, which is also shown as apattern in Table 4.

TABLE-4

$$X(i,j,1)=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&1&0&0\\0&0&1&0&1&0\end{bmatrix} \qquad X(i,j,2)=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&0\\1&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

The assignment represented by the above pattern is [(3,4,1), (3,1,2), (4,3,1), (4,5,1)], where, the 3rd person completed 1st job at first facility 4th job at second facility; 4th person completed 3rd job at first facility and 5th job at first facility. It can also be graphically represented as follows:



5. EXPERIMENTAL RESULTS

In MATLAB, a computer program is created and tested for the intended LSA. The experiments are performed by uniformly generating the cost values (C_{ijk}) between [1, 1000]. We experimented with a variety of challenges in various sizes. Cost random integers are used to build the matrix. The results are summarized in Table-5. It has been demonstrated that finding the best answer takes significantly less time. With the suggested LSA, the CPU takes time to identify the optimal solutions for various hard situations, as seen in the table below.

Table- 5: Computational Results

SN	Number	Number			Number	CPU Run Time (in seconds) by LSA	Total time

	of persons (m)	of jobs (n)	Number of facilities (p)	Number of persons to be done from (m^*)	of jobs to be done from (n^*)	Average time for alphabet table generation (AT)	Average time for getting optimal solution (OS)	Avg. (AT+OS) Seconds
1	4	10	2	2	8	0.0000	0.0000	0.0000
2	5	12	3	3	9	0.0000	0.0000	0.0000
3	7	21	4	4	15	0.0547	0.0000	0.0547
4	10	32	5	6	24	0.1088	0.0000	0.1088
5	15	45	7	9	40	0.2193	0.2751	0.4944
6	22	100	8	10	79	0.2752	0.2192	0.4944

6. CONCLUSION

The assignment issue is a versatile combinatorial optimization problem that can be used to mimic virtually any real-world scenario. Several aspects of the assignment problem, such as the limitations and solution approach employed in the education area, have been investigated. As a result, this study provides a review of assignment issues that have been covered in the prior literature in the context of educational activities. Not only that, but this paper offers a step-by-step way to tackling an assignment problem.

Furthermore, depending on the problem's complexity, it's critical to take the proper strategy to solve the problem in order to arrive at an optimal or sub optimal solution. According to the existing literature, the heuristic and metaheuristic were proved the best for solving the assignment problem since they generated good, but not necessarily optimal, results. Numerous recommended ways that have been considered have also been discovered to be important to employ and improved in various practical circumstances. Because of its versatility, in numerous applications that can be utilized in real-world circumstances, the assignment problem will continue to be an endless riddle in the future.

7. REFERENCES

1. H. Basirzadeh, "Ones assignment method for solving assignment problems," Applied Mathematical Sciences, vol. 6, no. 45-48, pp. 2345–2355, 2012.
2. S. Singh, "A Comparative Analysis of Assignment Problem," IOSR Journal Of Engineering, vol. 2, no. 8, pp. 1–15, 2012.
3. E. Çela, "Assignment Problems," in Handbook of Applied Optimization, Part II-Applications, vol. 6, pp. 667–678, 2002.
4. R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of Scheduling, vol. 12, no. 1, pp. 55–89, 2009.

5. M. Ayob, S. Abdullah, and A. M. A. Malik, "A practical examination timetabling problem at the Universiti Kebangsaan Malaysia," International Journal of Computer Science and Network Security, vol. 7, no. 9, pp. 198–204, 2007.
6. M. Caramia, P. Dell'Olmo, and G. F. Italiano, "Novel local-search-based approaches to university examination timetabling", INFORMS Journal on Computing, vol. 20, no. 1, pp. 86–99, 2008.
7. E. Ersoy, E. Özcan, and A. S. Uyar, "Memetic algorithms and hyperhill-climbers," in Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA07), pp. 159–166, 2007.
8. H. Turabieh and S. Abdullah, "An integrated hybrid approach to the examination timetabling problem," Omega , vol. 39, no. 6, pp. 598–607, 2011.
9. R. E. Burkard, "Selected topics on assignment problems," Discrete Applied Mathematics: The Journal of Combinatorial Algorithms, Informatics and Computational Sciences, vol. 123, no. 1-3, pp. 257–302, 2002.
10. U. Balakrishna, and Sundaramurthy. M,"A Pattern Recognition Lexi Search Approach to Generalized Time Dependent Travelling Salesman Problem", Journal of Operational Research society of India(Springer Publications), 49(3),pp.191-208,2012.
11. N Vasantha kumar, Balakrishna. U,"Constrained Asymmetric Three Dimensional Generalized Travelling Salesman Problem::Pattern Recognition Lexi Search Approach", Journal of Emerging Technologies and Innovative Research(JETIR),6(3),pp.172-176,2019.
12. Purusotham Singamsettya , Jayanth Kumar Thenepalle and Balakrishna Uruturu, "Solving open travelling salesman subset-tour problem through a hybrid genetic algorithm", Journal of Project Management, ISSN 2371-8374 (Online) - ISSN 2371-8366 (Print) 6(4) 209–222,2021
13. Masthan SK, Balakrishna U, Sankar Sekhar Raju and Jayanth Kumar T,"Solving an open k-city travelling salesman problem with ordered constraint: An exact Leaxi-search algorithm", Turkish Online Journal of Qualitative Inquiry(TOJQI), 12(7), 4520-4528,July 2021
14. Thenepalle, J. K., & Singamsetty, P. (2019). Lexi-search algorithm for one to many multidimensional bi-criteria unbalanced assignment problem. International Journal of Bio-Inspired Computation, 14(3), 151-170.