[1,*]Yongwen Hu
[2]Lixun Wang
[3]Zejian Zhang

A Feasible Flow-based Algorithm for Solving Classic Assignment Problem with Bottleneck

JES Journal of Electrical Systems

Abstract: - Classic assignment problem with bottleneck is a fundamental combinatorial optimization problem with numerous practical applications in diverse domains. In order to comprehensively explore the classic assignment problem with bottleneck and to develop efficient algorithm, a feasible flow-based algorithm is proposed for optimizing classic assignment problem with bottleneck. The bottleneck constraints are addressed by updating the bottleneck of the problem, and a feasible-flow approach is leveraged to efficiently identify optimal solutions within the network framework associated with such problems. To assess the effectiveness of our proposed algorithm, a comprehensive set of numerical experiments is carried out across various instances of classic assignment problem with bottleneck. Our numerical results demonstrate the superior performance of our algorithm, particularly when applied to problems with smaller cost scales, making it particularly suitable for addressing medium-sized to smaller-scale problems.

Keywords: Classic Assignment Problem with Bottleneck, Network Flow Model, Admissible Arc, Numerical Experiments.

I. INTRODUCTION

In the past several decades, the assignment problem (AP) has garnered theoretical and practical attention from numerous researchers and practitioners. The AP stands as a widely recognized combinatorial optimization problem, and researchers have extensively investigated various adaptations and variations of this problem. Among the variations of AP presented by researchers were ones to the bottleneck AP [1], the generalized AP, the semi-AP, the quadratic AP, and a variety of others. Also, there were some papers combining two or more of these basic variations [2-4].

The fundamental objective of the classic assignment problem (CAP) is to establish a bijective mapping between *n* agents and *n* jobs with the primary aim of minimizing the overall cost associated with these assignments. However, in real life, multiple jobs may be executed in parallel, and not only the total cost of assignments but also the duration of the jobs will be minimized. This paper concentrates on a multi-objective problem, which is presented as follows:

$$\begin{aligned} &minmax_{x_{ij}=1}\{c_{ij}\} \\ &min\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \end{aligned} \tag{1}$$

subject to the identical constraints and definitions as those in the CAP. The stated problem is a special version of bottleneck assignment problem (BAP), and many applications can be found in real life. For instance, in the context of an assembly line, each operator must be allocated to a workstation for task execution. Given that the cycle-time of the assembly line is determined by the slowest workstation, optimizing system productivity necessitates assigning each operator to a workstation with the shortest processing time, thereby minimizing the longest processing time. Another illustrative scenario involves the allocation of railway emergency crews to areas impacted by natural disasters. In such cases, swift and efficient allocation of resources can be critical in responding to emergency situations effectively. Further applications of the concept of bottleneck assignment can be identified in [5]. In fact, the problem has application in any AP situation where a minimax objective is appropriate.

Ford and Fullkerson [6] originally discussed BAP. After that, a number of researchers studied BAP extensively and proposed several algorithms for solving BAP. Particularly, one stream of researchers seeks to transform the BAP into CAP, and the Hungarian method can be applied to solve it. Pferschy [7] presents a solution method for BAP. Page [8] demonstrated the procedure for transforming the BAP into an equivalent CAP. Bhatia [9] gave a recursive algorithm for BAP by solving a series of special CAP with the Hungarian method. However, this method can result in a large-scale of transformed assignment matrix even for a small BAP. Edmonds and Fulkerson [10] devised an algorithm for solving the general BAP by introducing dummy jobs or works. The threshold algorithm, as described in reference [11], employs an initial threshold cost and examines the possibility of creating an

[1] School of Mechanical Engineering, Hubei University of Arts and Science, Longzhong, Xiangyang, 441053, Hubei, China
[2] School of Mechanical Engineering, Hubei University of Arts and Science, Longzhong, Xiangyang, 441053, Hubei, China
[3] Hubei Institute of Logistics Technology, Xiangyang, Hubei, China
*Corresponding author: Yongwen Hu

assignment with allocation costs below this threshold. The process involves iteratively increasing the threshold until a valid assignment meeting the criteria is discovered. The final assignment represents the optimal solution to the BAP. In [12], introduces enhancements to the complexity of the threshold algorithm by incorporating a binary search pattern for threshold adjustment, contrasting with the gradual incremental approach. Michael [13] provided two methods for the sensitivity of BAP.

Indeed, the CAP can be equally converted into a network flow problem. Derigs [14] presented an optimization method for BAP with a shortest augmenting path method. Hu [15] transformed GAP into a minimum cost flow problem and presented an algorithm with complementary slack condition. Recently, several exact algorithms [16-18] have been developed for the GAP. However, the exact algorithms [16-18] are applicable only under specific assumptions and constraints. Moreover, particularly when dealing with large-scale Assignment Problems (AP), the majority of exact algorithms often fail to provide an optimal solution in a short amount of time. No known polynomial-time algorithm exists for exact solutions to the BAP, but there are several approximate algorithms available for obtaining near-optimal solutions in an acceptable CPU time. Many approximate algorithms [19-22] have been proposed for solving AP.

This paper considers on a CAP with a bottleneck (CAP-B). As mentioned above, an optimal solution for CAP can be found by an algorithm based on network flow theory.

The remainder of this research is structured as follows. Section 2 transforms the CAP-B into a network flow problem; Section 3 introduces a deterministic algorithm designed for CAP-B. Also, the convergence of the algorithm is given in this section; In Section 4, results of the numerical experiments are presented. Section 5 gives concluding remarks and discusses potential avenues for further research.

II. NETWORK-FLOW MODEL FOR CLASSIC ASSIGNMENT PROBLEM WITH BOTTLENECK

Let $\boldsymbol{M} = \{1,2,\dots,n\}$ be the index set of n agents, $\boldsymbol{N} = \{1,2,\dots,n\}$ be the index set of n jobs, and c_{ij} be the cost of assigning the j-th job to the i-th agent. By introducing binary variables x_{ij},with

$$x_{ij} = \begin{cases} 1 & \text{job } j \text{ is assigned to agent } i, \\ 0 & \text{otherwise,} \end{cases}$$

the GAP-B can be stated as

$$CAP-B \quad \begin{cases} & min_{i\in \boldsymbol{M}, j\in \boldsymbol{N}} max_{x_{ij}=1}\{c_{ij}\} \\ & min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ s.t. & \sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\dots,n, \\ & \sum_{j=1}^{n} x_{ij} = 1, i = 1,2,\dots,n, \\ & x_{ij} = 0 \text{ or } 1, i = 1,2,\dots,n; j = 1,2,\dots n. \end{cases} \tag{2}$$

Clearly, CAP-B is a multi-objective problem, and a popular strategy that transforming multi-objective programming into singe-objective programming can be developed for solving it. Suppose the optimal bottleneck of CAP-B is $c^* = \text{minmax}\{c_{ij} \mid x_{ij} = 1, i, j = 1,2,\dots,n\}$. As c^* must be an element in efficiency matrix of the problem, it follows that, if c^* is given, CAP-B can be transformed into the following problem with a single objective.

$$A \quad \begin{cases} & minz = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}x_{ij} \\ & \sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\dots,n, \\ s.t. & \sum_{j=1}^{n} x_{ij} = 1, i = 1,2,\dots,n, \\ & c_{ij}x_{ij} \le c^*, i, j = 1,2,\dots,n, \\ & x_{ij} = 0 \text{ or } 1, i, j = 1,2,\dots,n. \end{cases} \tag{3}$$

Clearly, there is at least one index $i \in M$ and $j \in N$ such that $c_{ij} = c^*$. And c_{ij} can be set to be a big number if $c_{ij} > c^*, \forall i \in M, j \in N$. Furthermore, c^* can be updated with the minimum adjustment method without losing optimal solution.

As problem A is a CAP, there are several algorithms for solving it. It is a widely recognized fact that the CAP can be optimized by equivalently transforming it into a network flow problem.

A network flow model of classic assignment problem is given in Figure 1.

Let $G = (N', A, C, U)$ be a directed network, and $N' = \{s\} \cup M \cup N \cup \{t\}, A, C$ and U are the arc set, cost set and capacity set, respectively. In order to describe the constraints that each agent is required to be assigned to exactly one job, and conversely, every job is required to be assigned to exactly one agent., let $u_{ij} = 1, \forall (i,j) \in A, \forall u_{ij} \in U$, and for a given $c^*, c_{ij} \le c^*$ and $c_{si} = c_{jt} = 0, \forall i \in M, j \in N$. Therefore, for a given c^*, an optimal

solution for the assignment problem with bottleneck is equivalent to a solution of the network flow problem with a flow valued n, and its mathematical model can be presented as follows:

$$LP \left| \begin{array}{ll} & min\, z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ & \sum_{i=1}^{n} x_{si} = n, \\ s.t. & \sum_{j=1}^{n} -x_{jt} = -n, \\ & \sum_{j} x_{ij} - \sum_{k} x_{ki} = 0, \forall i \in M \cup N, i \neq s, t, \\ & 0 \leq x_{ij} \leq 1, \forall i, j \in M \cup U \cup \{s\} \cup \{t\}. \end{array} \right. \tag{4}$$

Hu et al. [23] presented a solution method for solving minimum cost flow problem, where the proposed algorithm identifies augmenting paths in the original network through node potential updates. The next section will give an algorithm by the network flow approach for CAP-B.

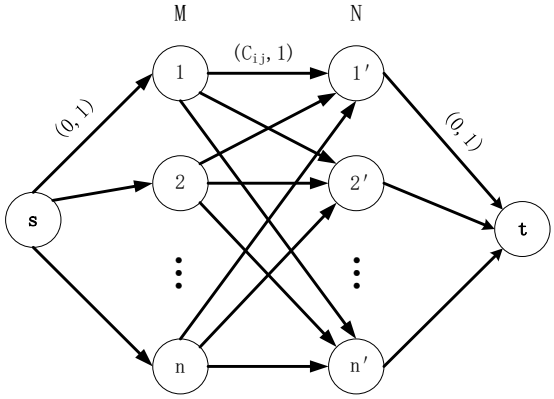

Figure 1: Minimum Cost Flow Model of the Minimum Time Limit Assignment Problem

III. Solution Method of Classic Assignment Problem with Bottleneck

As showed in Figure 1, the modified network flow model exhibits a distinctive feature wherein the capacity of each arc is set to one. This specific characteristic allows for the development of a more efficient algorithm tailored to this network model in comparison to a more general network model. For a given c^*, problem A is a CAP with n agents.

Clearly, the dual problem of (4) is as follows.

$$DP \left| \begin{array}{ll} & max\, \omega = kp_s - kp_t + \sum_{(i,j) \in A} u_{ij} p_{ij} \\ s.t. & p_i - p_j + p_{ij} \leq c_{ij}, \forall (i,j) \in A \\ & p_i\ is\ free, \forall i \in M \cup N \cup \{s\} \cup \{t\} \\ & p_{ij} \leq 0, \forall (i,j) \in A \end{array} \right. \tag{5}$$

where p_i and p_{ij} are the dual variable of LP. Furthermore, let p_i denote the potential of node i, while p_{ij} corresponds to the potential associated with the arc $(i,j) \in A$.

Next the optimality conditions of the minimum cost flow problem will be given. Let x_{ij} and $p = \{p_i, p_{ij}\}$ be a feasible solution of LP and DP, respectively. According to strong duality, if x_{ij} and $p = \{p_i, p_{ij}\}$ are optimal, the following are satisfied.

$$\begin{aligned} &x_{ij} = 0, \text{if } p_i - p_j + p_{ij} < c_{ij}, \forall i \in N, \\ &x_{ij} = u_{ij}, \text{if } p_i - p_j > c_{ij}, \forall i \in N, (i,j) \in A. \end{aligned} \tag{6}$$

As p_i is free $\forall i \in N$, let $p_{ij} = \min\{0, c_{ij} + p_j - p_i\}$, then $p_{ij} \leq 0$ will be satisfied. Thus, the conditions (6) can be stated equally as follows.

$$\begin{aligned} &x_{ij} = 0, \textit{if } p_i - p_j < c_{ij}, \forall i \in \boldsymbol{N} \\ &x_{ij} = u_{ij}, \textit{if } p_i - p_j > c_{ij}, \forall i \in \boldsymbol{N}, (i,j) \in \boldsymbol{A}. \end{aligned} \tag{7}$$

Theorem 1. *Let x_{ij} be a feasible solution of MCF, then x_{ij} is optimal if and only if the following conditions hold.*

$$\begin{cases} p_i - p_j < c_{ij} \Rightarrow x_{ij} = 0, \\ p_i - p_j > c_{ij} \Rightarrow x_{ij} = u_{ij}. \end{cases} \tag{8}$$

For any arc $(i,j) \in A$ is defined as an admissible arc if it satisfies the condition $p_i - p_j = c_{ij}$. And a network is defined admissible network if each arc contained within it qualifies as an admissible arc. Obviously, a solution $x_{ij} = 0, \forall (i,j) \in A$ is optimal with a total flow value of 0. The proposed algorithm commences by initiating the transmission of 0 units of flow from s to t. Throughout the iterations, the algorithm can identify at least one

augmenting path by updating node potentials within a finite number of iterations, while satisfying the condition (8). Hence, flow augmentation is possible when the current flow amount is not at its maximum capacity.

Let μ^+ represent the forward arc set and μ^- represent the backward arc set on an augmenting path μ. An algorithm, according to duality theory, for solving CAP corresponding the network $G' = (N', A, C, U)$ is described as Algorithm 1, where R is an admissible network.

Algorithm 1 Algorithm for solving CAP with duality theory:

Initial settings: $p_i = 0, \forall i \in \boldsymbol{N'}, x_{ij} = 0, \forall (i,j) \in A', \boldsymbol{S} = \{s\} \cup \boldsymbol{M}, \bar{\boldsymbol{s}} = \boldsymbol{N'} \setminus \boldsymbol{S}$. Label source node (0,1).

while $\sum_{i=1}^{n} x_{si} < n, i \in \boldsymbol{M}$ **do**

Remove all labels associated node $i, \forall i \in \boldsymbol{M} \cup \boldsymbol{N} \cup \{t\}$.

while $t \notin \boldsymbol{S}$ **do**

$p_i = p_i + \theta, \forall i \in \boldsymbol{S}$, where $\theta = \min_{k \in \boldsymbol{S} \cap \boldsymbol{M}, q \in \bar{\boldsymbol{S}} \cap \boldsymbol{N}} c_{kq} - p_k + p_q$.

if $p_i - p_j = c_{ij}, \forall i \in \boldsymbol{S}, j \in \bar{\boldsymbol{S}}$ **then**

$(i,j) \in R$

end if

if $(i,j) \in R, x_{ij} = 0, \forall i \in \boldsymbol{S}, j \in \bar{\boldsymbol{S}}$ **then**

Node j is assigned the label $(i,1), \boldsymbol{S} = \boldsymbol{S} \cup \{j\}, \bar{\boldsymbol{S}} = \bar{\boldsymbol{S}} \setminus \{j\}$.

end if

if $(j,i) \in R, x_{ji} = 1, \forall i \in \boldsymbol{S}, j \in \bar{\boldsymbol{S}}$ **then**

Node j is labeled with $(-i,1), \boldsymbol{S} = \boldsymbol{S} \cup \{j\}, \bar{\boldsymbol{S}} = \bar{\boldsymbol{S}} \setminus \{j\}$.

end if

end while

$$x_{ij} = \begin{cases} x_{ij} + 1, & (i,j) \in \mu^+, \\ x_{ij} - 1, & (i,j) \in \mu^-, \\ x_{ij}. \end{cases}$$

end while

Therefore, according to the above analysis, an optimal solution of CAP-B will be found if c^* is given. And c^* can be updated as follows without discarding optimal solution.

Let

$$c_{i'j} = min\{c_{ij} | i = j', j = 1,2,\dots,n\}, i' = 1,2,\dots,n, \tag{9}$$

$$c_{ij'} = min\{c_{ij} | j = j', i = 1,2,\dots,n\}, j' = 1,2,\dots,n, \tag{10}$$

$$T_1 = max\{c_{i'j} \in C_1, c_{ij'} \in C_2\}. \tag{11}$$

Now consider a set $C_r = \{c_{i'j}, i' = 1,2,\dots,n\}, \|C_r\| = n$, and $\forall c_{i_1'j}, c_{i_2'j} \in C_r, i_1' \neq i_2'$. Similarly, let $C_c = \{c_{ij'}, j' = 1,2,\dots,n\}, \|C_c\| = n$, and $\forall c_{ij_1'}, c_{ij_2'} \in C_c, j_1' \neq j_2'$. Evidently, it is possible to find an optimal solution, denoted as c_1^* and x_1^*, if $\forall c_{i_1 j_1}, c_{i_2 j_2} \in C_1$, and $j_1 \neq j_2, \forall i_1, j_1, i_2, j_2 = 1,2,\dots,n$. Furthermore, $c_1{}^* = \max\{c_{i'j} \mid c_{i'j} \in C_1\}$,and $x_{i'j}^* = 1$ if $c_{i'j} \in C_r; x_{i'j}^* = 0$, if $c_{i'j} \notin C_r$; Similarly, if the relation $\forall c_{i_3 j_3}, c_{i_4 j_4} \in C_c$, and $j_3 \neq j_4, \forall i_3, j_3, i_4, j_4 = 1,2,\dots,n$ hold, an optimal solution, an optimal solution c_2^* and x_2^* will be given, and $c_2{}^* = \max\{c_{ij'} \mid c_{ij'} \in C_c\}$ and $x_{ij}^* = 1$, if $c_{ij'} \in C_2; x_{ij'}{}^* = 0$ if $c_{ij'} \notin C_2$. If the minimum element of each row(column) does not belong to a different column(row), which indicates that T_1 defined in (11) is not an optimal solution of problem (2). T_1 will be updated by the following rules without discarding optimal solution.

$$T_1 = T_1 + min\{c_{ij} - T_1 | c_{ij} > T_1, i,j = 1,2,\dots,n\} \tag{12}$$

Obviously, after the updating process, T_1 is an element in efficiency matrix $[C_{ij}]$. Therefore, a new network corresponding to T_1 will be created, and let

$$c_{ij} = \begin{cases} M, c_{ij} > T_1 \\ c_{ij}, c_{ij} \leq T_1 \end{cases} \tag{13}$$

where M is a big positive number. An algorithm for solving problem (2) can be described as follows.

Algorithm 2 Algorithm for solving CAP-B:

Calculating T_1 by the rule (9) to (11) and creating a network $G = (N, A, C, U)$, where C is defined as (13). Let f be the maximal flow of the created network problem solved by Algorithm (1)

while $f < n$ **do**

Update T_1 according to (12);

Update $c_{ij} \in C$ by the rule of (13) and get a new network denoted as G.

Solve network G with Algorithm (1) and get a maximal flow f.

end while

Theorem 2 *Algorithm 2 will conclude within a finite number of iterations.*

$Proof$. Clearly, the sink node t cannot be labeled in association with node j if $x_{jt} = 1, \forall j \in \boldsymbol{N}$. Otherwise, as $c_{si} = c_{jt} = 0, \forall i \in \boldsymbol{M}, j \in \boldsymbol{N}$, node t will receive a label while node j is labeled. This implies that an augmenting path from s to t can be obtained, allowing for an additional unit of flow to be transmitted from s to t during the first iteration. Furthermore, node i can be assigned a label if node j is labeled and $x_{ij} = 1$. Hence, an augmenting path will be identified after updating node potentials in a finite number of iterations. Consequently, at least one additional unit of flow is dispatched from s to t. For CAP-B with n agnets, the total number of iterations is $\frac{n^2+n}{2}$, which implies the Theorem 2.

The following section will present a series of numerical experiments to evaluate the performance of Algorithm 2 as proposed.

IV. NUMERICAL EXPERIMENTS

This section presents a series of numerical experiments to evaluate the algorithm's effectiveness, efficiency, and behavior under various conditions. Our algorithm was implemented in MATLAB® R2016b, running on computer equipped with an Intel® Core i5-7300HQ CPU @ 2.50GHz, 8GB of RAM.

The proposed algorithm has been effectively evaluated with varying input data, and c_{ij} follows a uniform distribution within the intervals $[1, 10^2], [1, 10^4]$. The numerical experiments were conducted on networks of varying scales, where n ranged from 10 to 300). For a given n, The average CPU times were computed across 20 problem instances. The performance of the proposed algorithm is compared with a deterministic algorithm (turnpike approach proposed by Kuo [24]) and a meta-heuristic algorithm. For simplicity, the proposed algorithm is referred to as A. A meta-heuristic algorithm designed by Woodcock and Wilson [19] and a deterministic method by Kuo [24] for assignment problem are called B and C. The CPU time for each algorithm is given in Table 1.

Table 1: The Computational Performance of Randomly Generated CAP-B Instances, Varying in N and c_{ij} Intervals, is Evaluated in Terms of Execution Time (Measured in Seconds)

Number of agents	Average number. of arcs	Average CPU time $c \in [0,100]$			Average CPU time $c \in [0,10000]$		
		A	B	C	A	B	C
10	91	0.04	6.10	0.05	0.04	6.11	0.08
50	2243	0.47	9.59	0.75	0.58	10.49	0.83
100	9013	4.62	14.23	5.62	6.71	16.24	6.72
150	20239	24.86	28.56	13.48	40.53	51.57	18.09
200	36024	107.00	136.37	62.76	156.00	212.42	80.86
300	81021	610.00	849.31	157.39	1027.76	1245.94	201.76

The results show that our algorithm outperformed algorithm C by achieving an average CPU time reduction of 29% across problem instances when $n \leq 100$. Although our algorithm requires more CPU time compared to the bees algorithm when $n > 100$, our algorithm can find an exact optimal solution, while the bees algorithm finds a near-optimal solution of CAP-B.

In addition, our algorithm runs approximately 1.14-150.30 times faster than algorithm B across all problem instances. However, our algorithm will take more CPU time when $n > 100$ (compared with algorithm C) and $n > 150$ (compared with algorithm B). For a smaller range of c_{ij} (e.g. $c_{ij} \in [0, 10^2]$), our algorithm has a high probability to get more admissible arcs after updating the nodes potentials, which will accelerate the process for finding a flow valued n. On the contrast, our algorithm finds an optimal solution in more iterations for a larger range of c_{ij} (e.g. $c_{ij} \in [0, 10^4]$). Therefore, for the same number of agents, obtaining an optimal solution will require more CPU time when dealing with a larger range of c_{ij}. And for a larger range of c_{ij}, as the problem size increases, algorithm A experiences a greater increase in CPU time.

V. CONCLUTION AND FURTHER WORK

This study examines the CAP-B which is a variant of the CAP. An algorithm is proposed for solving the CAP-B. Our exact algorithm formulates the problem as a network model with a distinctive structure and solves it by iteratively updating the 'bottleneck' and node potentials.

Numerical experiments conclusively illustrate that the proposed algorithm can efficiently find optimal solutions, especially for a small-scale assignment problem with bottleneck. Further, the numerical experiments reveal that the algorithm's performance is sensitive to the range of c_{ij}. For smaller range of c_{ij}, the algorithm performs well, but its performance degrades rapidly as the range of c_{ij} becomes large.

While the proposed algorithm successfully identifies optimal solutions for the CAP-B, several potential directions for future research can be explored. Firstly, it is meaningful to investigate the performance of our algorithm on more complex and realistic instances of the problem, such as those with uncertain or dynamic demand and capacity constraints. This would require the development of more sophisticated optimization models and algorithms that can adapt to changing conditions over time. Another interesting direction for future work would be to explore machine learning and data-driven techniques for solving CAP-B.

ACKNOWLEDGMENT

This research was supported by the Hubei Provincial Natural Science Foundation Joint Fund for Innovation and Development (No. 2022CFD081), Hubei Provincial Philosophy and Social Science Project (Youth Project) (No.21Q238) and Xiangyang City Science and Technology Plan Project (Basic Research) (No. 2022ABH006436).

REFERENCES

[1] Mathur, K, Puri, M C. A bilevel bottleneck programming problem. European Journal of Operational Research, 1995, 86(2),.337–344.
[2] Beheshti, B, Prokopyev, O A, Pasiliao, E.L.. Exact solution approaches for bilevel assignment problems. Computational Optimization and Applications,2016,64, 215–242.
[3] Vasko F J, Dellinger A, Lu Y, et al. A simple and efficient technique to generate bounded solutions for the generalized assignment problem: A guide for OR practitioners. Research Reports on Computer Science, 2022, 13-34.
[4] Spivey M Z, Powell W B. The dynamic assignment problem. Transportation science, 2004, 38(4): 399-419.
[5] Pentico, D W. Assignment problems: A golden anniversary survey. European Journal of Operational Research,2007,176(2), 774–793.
[6] Fulkerson, D R. A Production Line Assignment Problem. Rand Corporation, Illinois,1953.
[7] Pferschy U. Solution methods and computational investigations for the linear bottleneck assignment problem. Computing, 1997, 59: 237-258.
[8] Page, E S. A note on assignment problems. The Computer Journal,1963,6(3), 241–243 (1963).
[9] Bhatia, H L. Time minimizing assignment problem. Systems and cybernetics in management,1977, 6(3),75–83.
[10] Edmonds, J, Fulkerson, D R. Bottleneck extrema. Journal of Combinatorial Theory,1970,8(3), 299–306.
[11] Garfinkel, R S. An improved algorithm for the bottleneck assignment problem. Operations Research,1971,19(7),1747–1751.
[12] Gabow, H N, Tarjan, R E. Algorithms for two bottleneck optimization problems. Journal of Algorithms,1988,9(3), 411–417.
[13] Michael, E, Wood, T A, Manzie, C, Shames, I. Sensitivity analysis for bottleneck assignment problems. European Journal of Operational Research,2022,303(1),159–167.
[14] Derigs, U, Zimmermann, U. An augmenting path method for solving Linear Bottleneck Transportation problems. Computing,1979,22(1), 1–15.
[15] Hu, Y, Liu, Q. A Network Flow Algorithm for Solving Generalized Assignment Problem. Mathematical Problems in Engineering, 2021, 1–8
[16] Andersen A C, Pavlikov K, Toffolo T A M. Weapon-target assignment problem: Exact and approximate solution algorithms. Annals of Operations Research, 2022, 312(2): 581-606.
[17] Ghoniem A, Flamand T, Haouari M. Exact solution methods for a generalized assignment problem with location/allocation considerations. INFORMS Journal on Computing, 2016, 28(3): 589-602.
[18] Beheshti B, Prokopyev O A, Pasiliao E L. Exact solution approaches for bilevel assignment problems. Computational Optimization and Applications, 2016, 64: 215-242.
[19] Woodcock, A J, Wilson, J M. A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. European journal of operational research,2010, 207(2), 566–578.
[20] Chu, P.C., Beasley, J E. A genetic algorithm for the generalised assignment problem. Computers & Operations Research,1997,24(1), 17–23.
[21] Tasgetiren, M F, Suganthan, P N, Chua, T J, Al-Hajri, A. Differential evolution algorithms for the generalized assignment problem. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2606–2613. IEEE, Turkey (2009)
[22] Sethanan, K, Pitakaso, R. Improved differential evolution algorithms for solving generalized assignment problem. Expert Systems with Applications,2016,45, 450–459.
[23] Hu, Y, Zhao, X, Liu, J, Liang, B, Ma, C. An Efficient Algorithm for Solving Minimum Cost Flow Problem with Complementarity Slack Conditions. Mathematical Problems in Engineering, 2020, 1–5.

[24] Kuo, C, Nicholls, G. A turnpike approach to solving the linear bottleneck assignment problem. The International Journal of Advanced Manufacturing Technology,2014,71(5-8),1059–1068.