Pareto Optimal Solutions of the Biobjective Bottleneck Assignment Problem

Lasko Laskov
Department of Informatics
New Bulgarian University
Sofia, Bulgaria
llaskov@nbu.bg

Marin Marinov
Department of Informatics
New Bulgarian University
Sofia, Bulgaria
mlmarinov@nbu.bg

***Abstract*—In this paper we describe the computation of all Pareto optimal solutions of a fundamental biobjective variant of the assignment problem (AP). The examined problem is the MAXMIN-MINMAX AP in which both criteria are non-linear bottleneck functions: *maximal capacity* and *minimal time*.**

We introduce the main algorithm that solves the biobjective bottleneck AP, along with its helper functions that are needed for its implementation. We prove the correctness of the algorithms their computational complexity.

The presented method is applicable to majority of the MAXMIN-MINMAX versions of the AP, and with few modifications can be adopted to solve the other standard biobjective bottleneck APs.

***Index Terms*—combinatorial optimization, assignment problem, Pareto optimality, biobjective optimization**

I. Introduction

Along with the network shortest path, the travelling salesman problem and the transportation problem, the *assignment problem* (AP) [1] is one of the fundamental subjects in combinatorial optimization. Since it was one of the problems that were used to establish the discipline operation research, it has been investigated since the 1950s, with the most famous work being [2] that introduces the well-known Hungarian method that solves the basic AP in polynomial time complexity. Since then, many works have been devoted to the AP and its variants based on different combinatorial techniques, graph theory algorithms, and heuristic approaches. Nowadays, AP is a current topic of research in the field (see for example [3], [4], [5]),and particularly the bottleneck AP [6].

The basic variant of AP is formulated as the problem of finding an assignment of n tasks (jobs, projects, processes) to n agents (workers, companies, processors) that is a subject of a given single objective function, most commonly the total cost of the proposed assignment. In this way the goal of the problem is to minimize the sum of the costs determined by the assignment, which is a linear function.

As shown in [1], different objective functions can be used as a subject of the problem. In the case in which for example a time criterion (see (4)) is minimized, or a capacity criterion (see (3)) is maximized, in literature they are referred to as *bottleneck* objective functions [6].

979-8-3503-1291-1/23/$31.00

For many practical problems a single objective function is not enough, and in this case models are defined with two objective functions to form a *biobjective* (or *bicriteria*) AP (see for example [7], [8]). In the case in which the objective functions are more than two, the AP is brought to a multiobjective combinatorial optimization problem [9].

In the case of biobjective AP the optimization problem is a subject of two objective functions. If the two functions are conflicting, this means that it cannot be found a solution that improves either one of the criteria, without making worse the other criterion. In this case we are interested to find the set of non-dominant solutions that are also called *Pareto optimal* [8], [10], [11].

By selecting the concrete functions (see Sec. II), different types of biobjective AP can be formulated. In his notable work [12] Hansen examines ten types of biobjective path problems and their complexity. He proposes two polynomial algorithms along with two fully polynomial approximation schemes. Here we are adopting the notations of [12] applied to the AP.

The goal of our research is to examine completely the problem of computation of all Pareto optimal solutions of the biobjective AP, in the case in which at least one of the objective functions is from the bottleneck type. Here, we focus particularly on the case in which both criteria are bottleneck functions: the MAXMIN and the MINMAX functions. Following the notations introduced in [12], we denote this variant of the biobjective AP as *MAXMIN-MINMAX problem*.

We propose an algorithm and all its helper functions that solve the MAXMIN-MINMAX problem and calculate a list of all Pareto optimal solutions that can be represented:

$$P = P_1 \cup P_2 \cup \ldots \cup P_m, \tag{1}$$

where each P_j, $j = 1, 2, \ldots, m$ is a subset of equivalent Pareto optimal solutions, and m is the number of such sets.

This paper is organized as follows. In Sec. II we introduce the basic mathematical notations and definitions and we formulate the MAXMIN-MINMAX problem. The algorithms are given in Sec. III, and also their correctness and computational complexity are proved. Finally, in Sec. IV we give conclusions and discussions.

II. Basic Notations and Problems Formulation

In this section we give the basic mathematical notations and we formulate the main problem covered in the paper.

A. Basic Notations

With $\mathbb{N}(n) = \{1, 2, \ldots, n\}$ we denote the set of first n natural numbers. For each vector w with $w(j)$ we denote its element on position j, and for each matrix A with $A(i, j)$ we denote its element located on row i and column j.

For an arbitrary matrix A, with

$$[A]^{i_1,i_2,\ldots,i_k}_{j_1,j_2,\ldots,j_s} \tag{2}$$

we denote the matrix that results from A after we delete the rows with indexes $i_1, i_2, \ldots, i_k$, and columns with indexes $j_1, j_2, \ldots, j_s$.

When we describe the execution of loops in algorithms, we will use the notation $a^{(i)}$ to the denote the value of a variable a, and $A^{(i)}$ to denote the value of a matrix A, that is calculated on the i-th iteration.

For each square matrix A of order n, and each vector $w \in \mathbb{N}^k(n)$, $k \in \mathbb{N}(n)$, we define the two functions $H_A(w)$ and $G_A(w)$. As we will clarify, the vector w represents an *assignment*.

The first function is a bottleneck function and it is the minimum of the weights for a given assignment.

$$H_A(w) = \min_{j \in \mathbb{N}(k)} \{A(j, w(j))\} \tag{3}$$

The function $H_A(w)$ represents the *capacity criterion*.

The second function $G_A(w)$ is also a bottleneck function and it is defined as the maximum of the weight for a given assignment.

$$G_A(w) = \max_{j \in \mathbb{N}(k)} \{A(j, w(j))\} \tag{4}$$

The function $G_A(w)$ represents a *time criterion*.

One of the standard approaches to represent an AP is as a problem defied on a bipartite graph [6]. With $G = (U, V, E)$ we denote a bipartite graph with vertices $U \cup V$ and a set of edges E. Since the objective of our research are the balanced APs, without any loss of generality we assume that $U = \mathbb{N}(n)$ and $V = \mathbb{N}(n)$. Then, the edge $\{i, j\}$ from E denotes that the i-th agent from U can be assigned on the j-th position from V. Each assignment is represented with a perfect matching in the bipartite graph G, and vice versa. We denote each perfect matching M with a special vector $w = (j_1, j_2, \ldots, j_n)$, where $\{i, j_i\} \in M$, for each $i \in \mathbb{N}(n)$. Therefore, the components of w are different natural numbers that are less than $n + 1$, in other words $w = (j_1, j_2, \ldots, j_n)$ *is an assignment if and only if it is a permutation of* $\mathbb{N}(n)$.

Definition 1: We will say that the 0-1 matrix A defines the set W of perfect matchings of the bipartite graph G with $w \in W$, if and only if w is a perfect matching of the bipartite graph with an adjacency matrix A.

With W_G we denote the set of all perfect matchings in the bipartite graph G. We assume that $G(U, V, E)$ is a complete bipartite graph that is given along with the two weights matrices A and T, which are defined by the corresponding objective functions.

B. The MAXMIN-MINMAX Problem

In the case of MAXMIN-MINMAX problem, for each edge $\{i, j\} \in E$ the weight $A(i, j) > 0$ represents the *capacity*, and the weight $T(i, j) > 0$ represents the *time* criterion. Then, the number $H_A(w)$ is the *capacity*, and the number $G_T(w)$ is the *time* of the assignment w.

Definition 2: We will say that the assignment $\widehat{w}$ is Pareto optimal with respect to $\{H_A, G_T\}$ when there *does not exist* an assignment w, for which at least one of the following statements holds:

- $H_A(w) > H_A(\widehat{w})$ and $G_T(w) \leq G_T(\widehat{w})$;
- $H_A(w) \geq H_A(\widehat{w})$ and $G_T(w) < G_T(\widehat{w})$.

Definition 3: We will call two assignments w' and w'' equivalent, when $H_A(w') = H_A(w'')$ and $G_T(w') = G_T(w'')$.

The assignment w'' is dominated by the assignment w' when $H_A(w') > H_A(w'')$ and $G_T(w') \leq G_T(w'')$ or $H_A(w') \geq H_A(w'')$ and $G_T(w') < G_T(w'')$.

As given in (1), with P we denote the set of all Pareto optimal assignments.

Definition 4: We will call the subset $P_0 \subseteq P$ a *minimal complete set (MCS)* of Pareto optimal solutions, when $P_0 = \{w_1, w_2, \ldots, w_m\}$, and $w_j \in P_j$ for each $j \in \mathbb{N}(m)$.

Problem 1 (MAXMIN-MINMAX): Calculate the list

$$L = \{(B_1, w_1, a_1, t_1), \ldots, (B_m, w_m, a_m, t_m)\}, \tag{5}$$

where for each $j \in \{1, 2, \ldots, m\}$, B_j is the 0-1 matrix that defines the set P_j from (1), $w_j \in P_j$, $a_j = H_A(w_j)$ and $t_j = G_T(w_j)$.

Corollary 1: Let us suppose that the list (5) is composed. Then $P_0 = \{w_1, w_2, \ldots, w_n\}$ is a MCS.

III. The MAXMIN-MINMAX Algorithm

In this section we describe the proposed algorithms that solves the problem MAXMIN-MINMAX (Prob. 1) which is defined in Sec. II-B. The main method (Alg. 3) is based on a number of helper functions, which we introduce beforehand.

We will use a function $PerfectMatching(Z)$ that takes as an input an arbitrary 0-1 matrix Z, and verifies if the bipartite graph G_Z with adjacency matrix Z has a perfect matching. The function calculates the pair (ind, w_0), where ind is either 0 if G_Z does not have a perfect matching, or 1 if G_Z has a perfect matching. In the case in which $ind = 1$, w_0 stores one perfect matching of G_Z. The $PerfectMatching(Z)$ function is implemented using the Hopcroft-Karp algorithm [13] for maximum matchings in bipartite graphs, which has computational complexity $O(n^{\frac{5}{2}})$.

We define a function $AscSort(T, A, a)$ that sorts in ascending order the elements $T(i, j)$ of the input matrix T, for which $A(i, j) \geq a$. The function returns two lookup tables:

- the sorted *unique* values of the elements of the matrix T, stored in the lookup table u, with p number of elements, $p \leq n^2$ and $u(s) < u(s + 1)$ for $s = 1, 2, \ldots, p$;

- the corresponding indexes of the in the matrix T, stored in the lookup table q, with p number of elements, where $q(s)$ is the list of all coordinates in T whose values are equal to $u(s)$:

$$q(s) = \{(i,j) : T(i,j) = u(s) \text{ and } A(i,j) \geq a\}. \quad (6)$$

The function $TimeMin(A,T,a)$ implements optimization with respect to the time criterion $G_T(w)$, depending on the input parameter a that satisfies the inequality

$$0 \leq a \leq \max_{w \in W_G} \{H_A(w)\}. \quad (7)$$

For its purpose, we define the subset of perfect matchings

$$W^a = \{w \in W_G : H_A(w) \geq a\}, \quad (8)$$

where W_G denotes the set of all perfect matchings in the graph G. Then, the function $TimeMin(A,T,a)$ calculates the tuple (T_0, w_0, t_0), where

- T_0 is a 0-1 matrix that defines the subset of perfect matchings

$$W_0 = \left\{ w_0 \in W^a : G_T(w_0) = \min_{w \in W^a} \{G_T(w)\} \right\}; \quad (9)$$

- $w_0 \in W^a$ and $G_T(w_0) = \min_{w \in W^a} \{G_T(w)\}$;
- $t_0 = G_T(w_0)$.

In the case in which $a = 0$, then $W^a = W_G$ and $G_T(w_0) = \min_{w \in W_G} \{G_T(w)\}$. In other words, w_0 represents the solution of the basic MINMAX problem, and t_0 is the minimal possible time of an assignment with the weight matrix T. Something more, the matrix T_0 defines the set of all optimal solutions by time criterion (see [14]).

Algorithm 1 Time minimum $TimeMin(A,T,a)$

Input: the input matrices A, T and the parameter a
Output: (T_0, w_0, t_0)

```
T_0 ← 0_{n×n}
ind ← 0
s ← 0
(u, q) ← AscSort(T, A, a)
while ind = 0 do
    s ← s + 1
    ∀(i, j) ∈ q(s) set T_0(i, j) ← 1
    (ind, w_0) ← PerfectMatching(T_0)
    if ind = 1 then
        t_0 ← G_T(w_0)
    end if
end while
```

The Alg. 1 us illustrated with the following example.

Example 1: Let the bipartite graph G be defined with the weight matrices, given below.

$$A = \begin{pmatrix} 5 & 7 & 2 & 6 \\ 4 & 7 & 13 & 11 \\ 8 & 11 & 9 & 6 \\ 7 & 9 & 8 & 4 \end{pmatrix} T = \begin{pmatrix} 8 & 7 & 10 & 9 \\ 6 & 9 & 10 & 8 \\ 5 & 11 & 7 & 10 \\ 9 & 8 & 10 & 8 \end{pmatrix} \quad (10)$$

We will trace the execution of Alg. 1 for $a = 5$. Firstly, the algorithm initializes the matrix T_0 with the zero square matrix of order n, denoted by $0_{n\times n}$, and initializes the variables ind and s with zeroes. It calls the sorting function $AscSort(T,A,a)$ that calculates the lookup tables $u = (5,7,8,9,10,11)$ and

$$q = (\{(3,1)\}, \{(1,2),(3,3)\}, \{(1,1),(2,4),(4,2)\}, \{(1,4),(2,2),(4,1)\}, \{(2,3),(3,4),(4,3)\}, \{(3,2)\}). \quad (11)$$

On the next stage, the algorithm enters the **while** loop, and starts iterating. The values $T_0^{(1)}$, $T_0^{(2)}$, $T_0^{(3)}$ of the matrix T_0 for each iteration of the loop are given in Tab. II.

Apparently, the graph G_{T_0}, defined by the adjacency matrix T_0, for the first and second iteration, does not have a perfect matching, and ind will remain equal to 0. In the third iteration the function $PerfectMatching(T_0^{(3)})$ discovers that the adjacency matrix $T_0^{(3)}$ contains at least one perfect matching $w_0^{(3)} = (1,4,3,2)$, and $G_T(w_0^{(3)}) = 8$.

Proposition 1: The Alg. 1 correctly defines the function $TimeMin(A,T,a)$.

Proof: The correctness of the Alg. 1 is proved by induction, and it is based on the correctness of the functions $PerfectMatching(Z)$ and $AscSort(T,A,a)$.

Base case. We assume that after the first iteration of the **while** loop, $ind = 1$. Hence, the matrix T_0 has elements

$$T_0(i,j) = \begin{cases} 1, & \text{if } T(i,j) = u(1) \text{ and } A(i,j) \geq a \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

The function $PerfectMatching(T_0)$ calculates that the bipartite graph G_{T_0} has at least one perfect matching. Let w' is a perfect matching of G_{T_0}. Then $w' \in W^a$, $T(i,w'(i)) = u(1)$ and $A(i,w'(i)) \geq a$, for each $i \in \mathbb{N}(n)$.

Since $u(1)$ is equal to the minimal element of $T(i,j)$, for which $A(i,j) \geq a$, then

$$G_T(w') = u(1) = \min_{w \in W^a} \{G_T(w)\} \quad (13)$$

Therefore, in this case the result (T_0, w_0, t_0) is correctly calculated.

Inductive step. Let $s > 1$ is such that for all $s-1$ iterations of the **while** loop $ind = 0$, but after the s iteration $ind = 1$. This means that the following two observations hold.

1) The matrix $T_0^{(s-1)}$ with elements

$$T_0^{(s-1)}(i,j) = \begin{cases} 1, & \text{if } T(i,j) \leq u(s-1) \text{ and } A(i,j) \geq a \\ 0, & \text{otherwise} \end{cases}$$

is an adjacency matrix of the bipartite graph $G_{T_0^{(s-1)}}$, which does not have perfect matchings.

2) The matrix T_0 with elements

$$T_0(i,j) = \begin{cases} 1, & \text{if } T(i,j) \leq u(s) \text{ and } A(i,j) \geq a \\ 0, & \text{otherwise} \end{cases}$$

is an adjacency matrix of the bipartite graph G_{T_0}, which has at least one perfect matching.

Then, for each perfect matching w' of G_{T_0} it is fulfilled that $w' \in W^a$, $T(i, w'(i)) \leq u(s)$, for each $i \in \mathbb{N}(n)$ and there exists at least one i_0 for which $T(i_0, w'(i_0)) = u(s)$. Therefore,

$$G_T(w') = u(s) = \min_{w \in W^a} \{G_T(w)\}, \tag{14}$$

which proves that in this case Alg. 1 calculates correctly (T_0, w_0, t_0).

To complete the proof, it is enough to notice that from the inequality (7) follows that there exists at least one perfect matching $\widehat{w}$. such that $H_A(\widehat{w}) \geq a$. Then $A(i, \widehat{w}(i)) \geq a$ for each $i \in \mathbb{N}(n)$ and for some s_0 it is fulfilled that $u(s_0) = G_T(\widehat{w})$. This proves that after no more then s_0 number of iterations, the algorithm will reach $ind = 1$. ■

Proposition 2: The complexity of the Alg. 1 is $O(n^{\frac{9}{2}})$.

Proof: The implementation of the algorithm uses the function $PerfectMatching(Z)$ which has complexity $O(n^{\frac{5}{2}})$ because it is implemented using the Hopcroft-Karp algorithm [13]. The complexity of both functions $AscSort(T, A, a)$ and G_T is less than $O(n^{\frac{5}{2}})$. Also, the **while** loop executes less than n^2 number of iterations. Therefore, the computational complexity of the function $TimeMin(A, T, a)$ can be evaluated to $O(n^{\frac{9}{2}})$. ■

Similar to the function $AscSort(T, A, a)$, we define a function $DesSort(A, T, t)$ that sorts in descending order the elements $A(i, j)$ of the matrix A, for which $T(i, j) < t$. The function $DesSort(A, T, t)$ returns as a result two vectors that represent lookup tables u and q, that are defined in analogy to the result of the function $AscSort(T, A, a)$.

The function $CapacityMax(A, T, t)$ implements optimization with respect to the capacity criterion $H_A(w)$ that depends on an input parameter t. For each $t \geq \min_{w \in W_G} \{G_T(w)\}$ we define

$$CapacityMax(A, T, t) = \max_{w \in W_t} \{H_A(w)\}, \tag{15}$$

where $W_t = \{w \in W_G : G_T(w) < t\}$. In other words, the function $CapacityMax(A, T, t)$ calculates the maximal capacity of an assignment $w \in W_t$. Apparently, if $t > \max_{i \in \mathbb{N}(n), j \in \mathbb{N}(n)} \{T(i, j)\}$, then $W_t = W_G$ and (15) implements the solution of the basic single criterion problem for maximal capacity.

The result of the function $CapacityMax(A, T, t)$ is stored in the tuple (A_0, w_0, a_0), where

- A_0 is a 0-1 matrix that defines the subset of perfect matchings

$$W_0 = \left\{ w_0 \in W_t : H_A(w_0) = \max_{w \in W_t} \{H_A(w)\} \right\}; \tag{16}$$

- $w_0 \in W_t$ and $H_A(w_0) = \max_{w \in W_t} \{H_A(w)\}$;
- $a_0 = H_A(w_0)$.

We use the following Alg. 2 to calculate the function $CapacityMax(A, T, t)$.

The two algorithms Alg. 1 and Alg. 2 are similar, however they differ in the objective function that is the subject of optimization, and that Alg. 1 searches for its minimum, while Alg. 2 searches for its maximum. The correctness of Alg. 2 is proved in analogy to Prop. 1, and its complexity is proved to be $O(n^{\frac{9}{2}})$ in analogy to Prop. 2.

Algorithm 2 Capacity maximum $CapacityMax(A, T, t)$

Input: the input matrices A, T and the parameter t
Output: (A_0, w_0, a_0)

```
A_0 ← 0_{n×n}
ind ← 0
s ← 0
(u, q) ← DesSort(A, T, t)
while ind = 0 do
    s ← s + 1
    ∀(i, j) ∈ q(s) set A_0(i, j) ← 1
    (ind, w_0) ← PerfectMatching(A_0)
    if ind = 1 then
        a_0 ← G_T(w_0)
    end if
end while
```

Example 2: Let the bipartite graph G be defined with the weight matrices, given in (10). We will trace the calculations of Alg. 2 when $t = 9$.

The sorting step of the algorithm calculates the lookup tables $u = (11, 9, 8, 7, 5, 4)$ and

$$q = (\{(2,4)\}, \{(3,3), (4,2)\}, \{(3,1)\}, \{(1,2)\}, \{(1,1)\}, \{(2,1), (4,4)\}). \tag{17}$$

Then the algorithm enters the **while** loop which calculates the values $A_0^{(1)}, A_0^{(2)}, \ldots, A_0^{(5)}$ of the of the matrix A_0 in each of the five iterations, as given in Tab. I.

The graph G_{A_0} that is defined by the adjacency matrix A_0 for the first four iterations (see Tab. I) does not contain a perfect matching and the variable ind will stay equal to 0. On the fifth iteration the function $PerfectMatching(A_0^{(5)})$ finds that in this case at least one perfect matching exists, and it is $w_0^{(5)} = (1, 4, 3, 2)$, with maximal capacity $H_A(w_0^{(5)}) = 5$.

Based on the helper functions $TimeMin(A, T, a)$ and $CapacityMax(A, T, t)$ we define the following Alg. 3 that solves MAXMIN-MINMAX problem defined in Prob. 1.

Algorithm 3 Solution of the MAXMIN-MINMAX problem

Input: the input matrices A, T
Output: the list L defined in (5)

```
L ← ∅
t ← ∞
a ← 0
(B^(0), w^(0), t^(0)) ← TimeMin(A, T, a)
while t^(0) < t do
    (·, ·, a) ← CapacityMax(A, T, t)
    (B, w, t) ← TimeMin(A, T, a)
    L ← L ∪ {(B, w, a, t)}
end while
```

TABLE I
THE MATRIX A_0 FOR EACH ITERATION OF ALG. 2

$A_0^{(1)}$	$A_0^{(2)}$	$A_0^{(3)}$	$A_0^{(4)}$	$A_0^{(5)}$
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Example 3: We will trace the calculations of Alg. 3 with the bipartite graph G that is defined with the weight matrices, given in (10).

Before the loop, the algorithm executes a call to the function $TimeMin(A, T, a)$, where $a = 0$. The values stored in the resulting matrix $B^{(0)}$ are given in the column of Tab. II that is labeled with $i = 0$. The values of the other two components of the result of the time optimization function are $w^{(0)} = (1, 4, 3, 2)$ and $t^{(0)} = 8$.

We will point out that the matrix $B^{(0)}$ defines all perfect matchings that have minimal time.

On the *first iteration* of the **while** loop the function $CapacityMax(A, T, t)$ results in maximal capacity $a^{(1)}$ evaluated to 7, which is used as an input of the function $TimeMin(A, T, a^{(1)})$. The resulting matrix $B^{(1)}$ is given in the column of Tab. II that is labeled with $i = 1$. The calculated perfect matching is $w^{(1)} = (2, 4, 3, 1)$, with $t^{(1)} = 9$. The current solution $(B^{(1)}, w^{(1)}, a^{(1)}, t^{(1)})$ is stored into the list L. Since $t^{(0)} = 8 < 9 = t^{(1)}$, the loop continues to the second iteration.

On the *second iteration*, the value of $t^{(1)}$ is the input parameter of $CapacityMax(A, T, t^{(1)})$, which results in maximal capacity $a^{(2)} = 5$ (see also Examp. 2), and it is used as an input of $TimeMin(A, T, a^{(2)})$. The resulting matrix $B^{(2)}$ is given in the last column of Tab. II, and $w^{(2)} = (1, 4, 3, 2)$, $t^{(2)} = 8$ (see also Examp. 1). The current solution is stored into the list L. Since $t^{(0)} = 8 = t^{(2)}$, the loop is terminated, and L contains the solution of Prob. 1, $L = \{(B^{(1)}, w^{(1)}, a^{(1)}, t^{(1)}), (B^{(2)}, w^{(2)}, a^{(2)}, t^{(2)})\}$.

The solution of the Prob. 1 in Examp. 3 gives a detailed characteristic of the Pareto optimal solutions. Indeed:

1) All Pareto optimal solutions are grouped in two classes (P_1 and P_2) of equivalent Pareto optimal solutions, because $|L| = 2$. The equality (1) is fulfilled with $m = 2$.
2) The second element of each solution stored in L is a Pareto optimal solution from the corresponding class. In this particular case, $w^{(1)} \in P_1$ and $w^{(2)} \in P_2$. This observation shows that Alg. 3 also separates one Pareto optimal MCS P_0. In Examp. 3 $P_0 = \{w^{(1)}, w^{(2)}\}$.
3) The matrix $B^{(i)}$ defines the class P_i, $\forall i \in \{1, 2\}$. In the particular Examp. 3 it is directly verified that $per(B^{(1)}) = 1 = per(B^{(2)})$ and therefore $P = P_0$.

We illustrate the solution of Examp. 3 on Fig. 1. The Cartesian plane is defined by the two criteria capacity c, and time t. Each point A_w with coordinates $(H_A(w), G_T(w))$ denotes an assignment w. With a white dot we plot each of the points

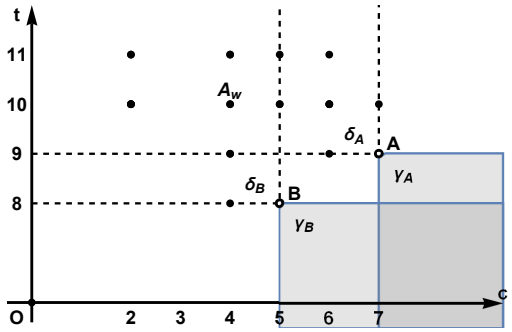

Fig. 1. Plot of the solution of Examp. 3 with capacity criterion c as abscissa, and time criterion t as ordinate. Black points denote assignments, while white points denote Pareto optimal solutions

$A(H_A(w_1), G_T(w_1)) = A(7, 9)$ and $B(H_A(w_2), G_T(w_2)) = B(5, 8)$ that denote Pareto optimal solutions. The point $A(7, 9)$ is the vertex of the angle $\gamma_A = \{(c, t) : c \geq 7 \text{ and } t \leq 9\}$. The point $B(5, 8)$ is the vertex of the angle $\gamma_B = \{(c, t) : c \geq 5 \text{ and } t \leq 8\}$. The angles γ_A and γ_B are colored in gray.

As it can be expected, the points A and B are vertices of the set $F = \gamma_A \cup \gamma_B$, which is colored in gray. From the plot it is clear that there is no other point A_w, that belongs to F. This observation illustrates the fact that the assignments $w^{(1)}$ and $w^{(2)}$ satisfy Def. 2.

We denote the vert. opp. angle of γ_A with $\delta_A = \{(c, t) : c \leq 7 \text{ and } t \geq 9\}$. The rays of δ_A are drawn in dashed lines. Analogously, $\delta_B = \{(c, t) : c \leq 5 \text{ and } t \geq 8\}$ is the vert. opp. angle of γ_B, and its rays are also drawn in dashed lines.

From Def. 3 it follows that an assignment w is dominated by $w^{(1)}$ exactly when $A(H_A(w), G_T(w)) \in \delta_A$. Also, w is dominated by $w^{(2)}$ exactly when $A(H_A(w), G_T(w)) \in \delta_B$. Hence, Fig. 1 shows that each assignment is dominated either by $w^{(1)}$ or by $w^{(2)}$, which illustrates the fact that there are no other Pareto optimal assignments.

Any 0-1 matrix $B^{(i)}$ that is included in the corresponding element of the resulting list L of the Alg. 3 defines all perfect matchings for the i-th solution. In order to extract the list of perfect matchings from matrices, we adopt the function $ListPM(A, k)$ that takes as parameters a 0-1 square matrix A of order n and a natural number k. The result of the function is a list of perfect matchings W_i in the graph G_A with the following properties:

1) If G_A contains more than $k - 1$ perfect matchings, then $|W_i| = k$.
2) If G_A contains less than $k + 1$ perfect matchings, then W_i stores all of them.

TABLE II
THE MATRIX T_0 FOR EACH ITERATION OF ALG. 1 AND THE MATRIX B BEFORE THE LOOP, AND FOR EACH ITERATION OF ALG. 3

$T_0^{(1)}$	$T_0^{(2)}$	$T_0^{(3)}$	$B^{(0)}$	$B^{(1)}$	$B^{(2)}$
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

The algorithm of the function $ListPM(A,k)$ is given as Alg. 2 in [14].

Theorem 1: The Alg. 3 is correct.

Proof: The proof is based on the correctness of the functions $TimeMin(A,T,a)$ and $CapacityMax(A,T,t)$.

Using mathematical induction we prove that each iteration of the **while** loop calculates correctly the corresponding element (B_j, w_j, a_j, t_j) of the list L. At the same time, we prove that for each j the sets $P_j = \{w_0 \in W_G : H_A(w_0) = a_j \text{ and } G_T(w_0) = t_j\}$ are defined by the corresponding matrices B_j, and P_j are sets of Pareto optimal assignments. Besides that, the following equalities hold:

$$a_j > a_{j+1} \text{ and } t_j > t_{j+1},\ \forall j \in \mathbb{N}(m-1), \tag{18}$$

$$W_{t_j} = \{w \in W_G :\ H_A(w) < a_j \text{ and } G_T(w) < t_j\},\ \forall j \in \mathbb{N}(m). \tag{19}$$

■

Proposition 3: The computational complexity of Alg. 3 is $O(n^{\frac{13}{2}})$.

Proof: We have proved that both Alg. 1 and Alg. 2 have computational complexity $O(n^{\frac{9}{2}})$. Therefore, each individual iteration of the **while** loop has complexity $O(n^{\frac{9}{2}})$. The number of iterations is m, which is the number of elements in the list S. Besides that, from the definition of the function $TimeMin(A,T,a)$ it follows that for each $k \in \mathbb{N}(m)$ there exists such an element $T(i_k, j_k)$ of the matrix T, that $t^{(k)} = T(i_k, j_k)$. From Th. 1 we know that $t^{(1)} > t^{(2)} > \ldots > t^{(m)}$. Hence, $m \leq n^2$, which completes the proof. ■

IV. CONCLUSION

In this paper we solve a biobjective version of the AP in which the two objective functions are conflicting, and we look for the set of all Pareto optimal solutions. The examined problem is the MAXMIN-MINMAX problem in which the first criteria represents the maximal capacity, while second criteria represents the minimal time.

The presented Alg. 3 gives a complete description of the set of all Pareto optimal assignments P. The result contains a compact representation of each of the sets P_i of equivalent Pareto optimal assignments. At the same time, if from each P_i is selected a single element, we will get a MSC of Pareto optimal assignments. The correctness of Alg. 3 is proved and its computational complexity is shown. Examp. 3 illustrates the calculations of Alg. 3 for the example given in (10). The graphical representation in Fig. 1 is in fact a geometrical proof of the correctness of the solution of the example.

The functions with corresponding algorithms that implement them $TimeMin(A,T,a)$ and $CapacityMax(A,T,t)$, solve the problems for list composition of the corresponding single criterion problems. The function $TimeMin(A,T,0)$ solves the single criterion problem for calculation of all assignments with minimal time. Analogously, the function $CapacityMax(A,T,t)$ finds all assignments that have maximal capacity.

REFERENCES

[1] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems*, ser. SIAM. University City, Philadelphia: Society for Industrial and Applied Mathematics, 2009.

[2] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, March 1955.

[3] T. Oncan, Z. Şuvak, M. H. Akyüz, and I. K. Altınel, "Assignment problem with conflicts," *Computers & Operations Research*, vol. 111, pp. 214–229, 2019.

[4] K. Morita, S. Shiroshita, Y. Yamaguchi, and Y. Yokoi, "Fast primal-dual update against local weight update in linear assignment problem and its application," *Information Processing Letters*, vol. 183, p. 106432, 2024.

[5] S. Dhouib, "An intelligent assignment problem using novel heuristic: The dhouib-matrix-ap1 (dm-ap1): Novel method for assignment problem," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 1, p. 135–141, 2022.

[6] E. Michael, T. A. Wood, C. Manzie, and I. Shames, "Sensitivity analysis for bottleneck assignment problems," *European Journal of Operational Research*, vol. 303, no. 1, pp. 159–167, 2022.

[7] D. Tuyttens, J. Teghem, P. Fortemps, and K. V. Nieuwenhuyze, "Performance of the MOSA method for the bicriteria assignment problem," *Journal of Heuristics*, vol. 6, pp. 295–310, 2000.

[8] Y. Ge, M. Chen, and H. Ishii, "Bi-criteria bottleneck assignment problem," in *2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, 2012, pp. 1–5.

[9] M. Ehrgott and X. Gandibleux, *Multiobjective Combinatorial Optimization — Theory, Methodology, and Applications*. Boston, MA: Springer US, 2002, pp. 369–444.

[10] O. Berman, D. Einav, and G. Handler, "The constrained bottleneck problem in networks," *Operations Research*, vol. 38, no. 1, pp. 178–181, 1990.

[11] S. Geetha and K. Nair, "A variation of the assignment problem," *European Journal of Operational Research*, vol. 68, no. 3, pp. 422–426, 1993.

[12] P. Hansen, "Bicriterion path problems," *Multiple Criteria Decision Making Theory and Application*, pp. 109–127, 1980.

[13] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[14] L. M. Laskov and M. L. Marinov, "List of selected number of optimal solutions of the assignment problem by time criterion," in *2022 International Conference Automatics and Informatics (ICAI)*. IEEE, 2022, pp. 100–106.