Affectation multicritère de tâches à des unités hétérogènes de traitement avec contraintes d'incompatibilité et de capacité

Bernard Roy, Roman Slowinski

MULTI-CRITERIA ASSIGNMENT OF TASKS TO HETEROGENEOUS PROCESSING UNITS WITH INCOMPATIBILITY AND CAPACITY CONSTRAINTS

Bernard Roy[1], **Roman Słowiński**[2]

[1]LAMSADE, University of Paris Dauphine, 75775 Paris, France;
roy@lamsade.dauphine.fr

[2]Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland; slowinsk@sol.put.poznan.pl

1. Introduction and problem statement

We are considering the following assignment problem. A finite set T of *tasks* t_h, h=1,...,n, has to be assigned to a finite set P of *processing units* p_j, j=1,...,m. Each task t_h has to be assigned to exactly one processing unit, however, more than one processing unit is able to process t_h. Each task t_h has, moreover, a different "preference" for being assigned to a particular processing unit; this "preference" is expressed by a *dissatisfaction degree* of t_h for being assigned to p_j, denoted by d_{hj}. The dissatisfaction degree $d_{hj} \geq 0$ for h=1,...,n; j=1,...,m; the greater d_{hj} the higher the dissatisfaction; $d_{hj}=\infty$ means impossibility of assigning t_h to processing unit p_j.

Each processing unit p_j has a regular *capacity* of processing m_j tasks. This capacity is flexible to some extent and can be increased up to $M_j \geq m_j$ tasks, however, only m_j tasks can be processed on p_j without any cost. One additional task over m_j augments the *processing cost* of p_j by a marginal cost $c_{j1}>0$, similarly, the second additional task by c_{j2}, etc., until the (M_j-m_j)-th additional task that augments the processing cost by $c_{j(M_j-m_j)}$, where these marginal costs are non-decreasing. Obviously, $\sum_{j=1}^{m} M_j \geq n$.

The assignment of tasks to processing units has to respect some *incompatibility constraints*. Let $Q_{ij} \subseteq T$, i=1,...,u_j, j=1,...,m, be a subset of tasks such that each $t_h \in Q_{ij}$ can be processed by processing unit p_j, however, in a feasible assignment, at most one $t_h \in Q_{ij}$ can be assigned to p_j, for i=1,...,u_j and j=1,...,m. Q_{ij} is called the i-th *incompatibility group* of processing unit p_j. Different incompatibility groups of the same processing unit must be disjoint. If a task t_h can be processed by a processing unit p_j and if $t_h \notin Q_{ij}$, i=1,...,u_j, then p_j is said to be able to process t_h *directly*; if t_h belongs to one of incompatibility groups of p_j, then p_j is said to be able to process t_h *indirectly*. It follows from the definition of Q_{ij} that a task t_h which can be processed directly by a processing unit p_j does not enter any incompatibility group of p_j.

The quality of feasible assignments is evaluated by three criteria: G_1 - the maximum dissatisfaction of tasks, G_2 - the total dissatisfaction of tasks, G_3 - the total cost of processing units.

One can meet assignment problems of this type in many real life situations. Let us give a few examples:

a). Organization of the students' choice of optional courses[1]

Each processing unit corresponds here to a course (or seminar) that students may have to follow in order to obtain a diploma. Each student e_i has to follow a fixed number u of courses chosen from among set P of courses given by different professors. Student e_i is thus associated with u tasks: $t_{i1},\ldots,t_{iu}$.

For each task $t_{i1},\ldots,t_{iu}$, student e_i presents two possibilities: the first possibility points out a first choice course (dissatisfaction zero) while the other, a second choice course (dissatisfaction greater than zero). All u first choices must be different. As to second choices, they do not need to be different neither among them nor from the first choices. It is thus possible that a course p_j is pointed out many times by student e_i in the secondary choice and also once in the first choice (some reasonable constraints impose a minimum diversity). Let Q_{ij} be a set of tasks for which student e_i has pointed out course p_j; as this course may be followed only once, Q_{ij} is the i-th incompatibility group of processing unit p_j; in this case, p_j can process indirectly each task included in Q_{ij}. If student e_i has pointed out course p_j only once, then the corresponding task can be processed directly by p_j.

The way in which some courses are organized (students' oral reports or limited access to some facilities) can lead to fixing an upper bound on the number of students accepted for the course. It is then realistic to represent this constraint by two limits: the first one, that would be desirable to keep (regular capacity m_j), and the second one, that is rather impossible to overstep (maximal capacity M_j). Each additional student over m_j deteriorates the conditions in which the course is carried out, and the deterioration rate increases with the number of additional students.

b). Timetabling

One is considering here a set of activities (courses, exams, meetings, exhibitions, etc.) that should be scheduled over time. These activities should be performed in places called rooms. Moreover, they have the same unit duration that fits a time slot. For a given activity t_h, many time slots are possible but they are more or less satisfactory. During each time slot p_j, m_j rooms are normally available, but it is also possible to adapt some extra rooms at additional cost depending on the room.

[1] A problem of this type arises when students (preparing a diploma in scientific methods of management at the University of Paris Dauphine) are to be assigned to one major and one minor seminar, taking into account their preferences and pedagogic capacity of professors. The case of Dauphine students was at the origin of the present study, however, before the present model was set up, a less general one existed together with preliminary software made in collaboration with several students. B. Roy used this supporting tool during the last decade, as a director of this diploma.

In this case, the tasks represent activities, and the processing units – time slots with available rooms. The incompatibility groups of time slot p_j are composed of activities that can be performed in this time slot but assignment of one activity to that slot excludes assignment of any other from this group, as they all require the same resource, for example a professor, a facility or the same group of participants.

c). Scheduling on machines

Set T of n tasks has to be processed on set P of m parallel heterogeneous machines (e.g. processors of a parallel computer system). The processing of each task on a particular machine is more or less efficient (dissatisfaction degree) and some tasks, in addition to a machine, require a unique additional resource (e.g. input-output channel, printer etc.). Each machine p_j is normally available in m_j copies which can be enlarged to M_j at some extra cost (e.g. spent for communication). The incompatibility groups of a given machine are composed of tasks that can be processed on this machine and require the same additional resource.

Many other applications can fit the considered model, in particular, from the field of personnel management, distribution and transportation. In all these contexts two types of questions arise:

i). If all tasks from set T can be assigned, how to define and choose the "best assignment"? An assignment that accepts some dissatisfaction of tasks will be, in general, more economic than an assignment which tends to perform each task in most satisfactory conditions. The latter may introduce indeed high costs of overloading some processing units. In consequence, it seems reasonable to search for a compromise between dissatisfaction of tasks (maximum dissatisfaction degree and total dissatisfaction) and the cost of overloading the processing units.

ii). If it is impossible to assign all tasks from set T, how to characterize the reasons of this impossibility? After they have been clearly identified, what actions can be undertaken (modifications of some input data) to enable additional assignments?

The paper is aiming at answering the above questions. It is organized in the following way. In section 2, the problem of finding a feasible assignment is formulated as a problem of finding a maximum integer flow in a special network. Section 3 is devoted to analysis of the case of impossible assignment; subsets of tasks, incompatibility groups and processing units that are responsible for this impossibility constitute so-called blocking configuration; to solve the impossibility problem, six actions of unblocking are identified; it is then demonstrated that they are the only ones which can create possibilities of additional assignments. The multi-criteria

assignment problem is considered in section 4; a strategy for selection of the best compromise assignment with respect to the three criteria G_1, G_2, G_3 defined above is outlined. Section 5 presents procedures for exploration of a set of non-dominated assignments on the plane ($G_2 \times G_3$); they are specialized in searching for supported and non-supported solutions, respectively. Numerical examples with and without blocking configuration are presented in section 6; they are solved using a specialized software called MASCOME (Multi-criteria Assignment Subject to Constraints Of Mutual Exclusion) (Slowinski, 2001). Conclusions are drawn in section 7.

2. Modeling the set of feasible assignments

The problem of finding a feasible assignment of tasks from T to processing units from P can be formulated as a problem of finding a maximum integer flow in the network **N**, shown in Fig. 1, being characterized by:

1) a set of vertices V in which two special vertices are distinguished: a *source* s and *destination* d,
2) a set of directed arcs $W \subseteq V \times V$,
3) integer numbers $\kappa(w) \geq 0$, associated with each arc $w \in W$, called the *capacity* of arc w, giving the upper bound for the flow on this arc; the lower bound on all arc flows is equal to zero.

Specifically, apart from s and d, the vertex set V is composed of three subsets:

- T, including n vertices t_h corresponding to the tasks (h=1,..,n),
- Q, including u vertices q_{ij} corresponding to incompatibility groups Q_{ij} (i=1,...,u_j, j=1,...,m; $u = \sum_{j=1}^{m} u_j$),
- P, including m vertices p_j corresponding to the processing units (j=1,...,m).

Below, when speaking about incompatibility groups in the network context, we will use the vertex notation q_{ij} instead of the set notation Q_{ij}. The arc set W is composed of five types of arcs:

- arcs (s, t_h), called *entering arcs*, with capacity equal to one,
- arcs (t_h, p_j), called *directly connecting arcs*, with unlimited capacity,
- arcs (t_h, q_{ij}), called *indirectly connecting arcs*, with unlimited capacity,
- arcs (q_{ij}, p_j), called *incompatibility arcs*, with capacity equal to one,
- arcs (p_j, d), called *leaving arcs*, with capacity equal to M_j.

The network **N** has n entering arcs, m leaving arcs and u incompatibility arcs. Moreover, the directly connecting arc (t_h, p_j) exists in **N** if and only if task t_h can be processed directly by processing unit p_j. Finally, the indirectly connecting arc (t_h, q_{ij}) exists in **N** if and only if processing unit p_j can process task t_h indirectly, that is if t_h belongs to the i-th incompatibility group of p_j. Thus, there is at most one path from t_h to p_j.

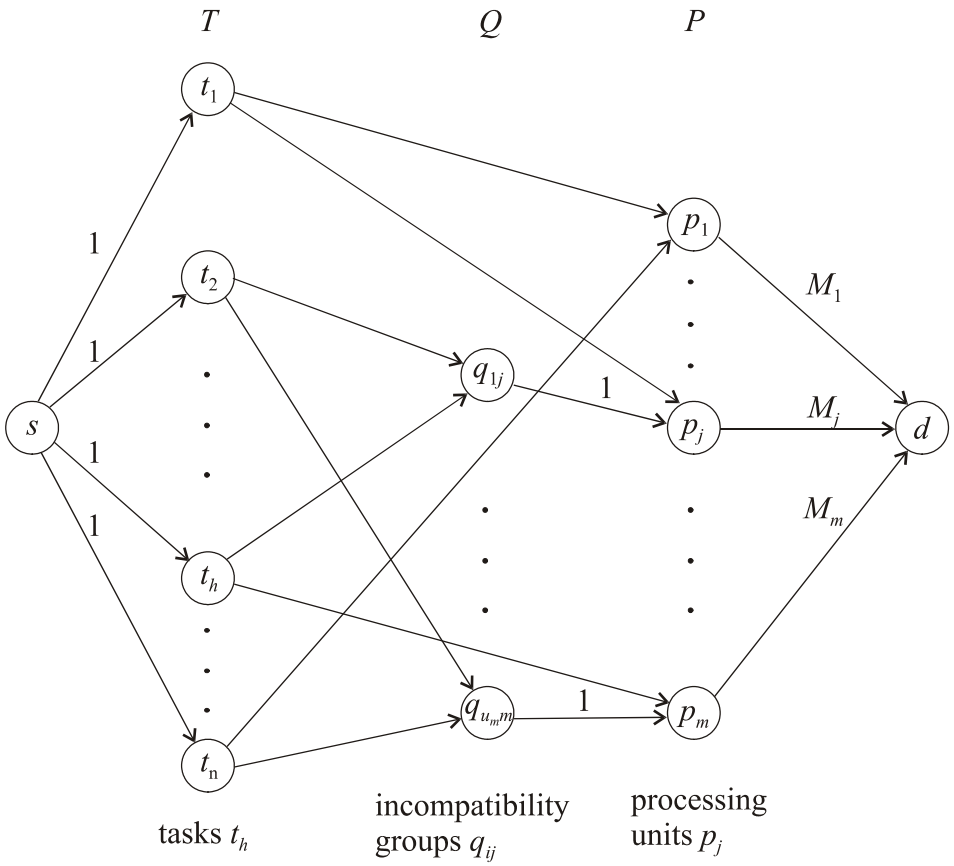


Fig. 1. Finding a feasible assignment as a problem of finding a maximum integer flow in network **N**

Obviously, every integer flow in the network **N**, having value equal to n, represents an assignment respecting the constraints of our problem and, vice versa, for any assignment respecting the constraints of our problem, one can find an integer flow equal to n in the network **N**. Clearly, each flow in **N** equal to n is a maximum flow in **N**, but the converse is not true.

If a maximum flow in **N** does not saturate all entering arcs, that is if the value of the maximum flow is n-ρ, and ρ>0, then this means that it is impossible to assign ρ tasks to available processing units, given the imposed constraints. In section 3, we will show that this impossibility comes from an existence in **N** of a configuration of tasks, processing units and incompatibility

groups called *blocking configuration*. We will then analyze blocking configurations in order to formulate all possible relaxations of the constraints enabling a complete assignment.

3. Analysis of blocking configurations and actions of unblocking

3.1. A brief reminder of network flow properties

Let us consider a maximum flow in network **N** and the associated labeling procedure defined as follows:

1) vertex *s* is labeled,

2) any vertex following a labeled vertex via a non-saturated arc is also labeled,

3) any vertex preceding a labeled vertex via a non-empty arc is also labeled.

According to the well-known maximum-flow minimum-cut theorem of Ford and Fulkerson (1962), we know that:

- the flow in a network has a maximum value if and only if vertex *d* is unlabeled,
- if vertex *d* has been labeled then the labels give a hint how to augment the flow in the network.

The theorem of Ford and Fulkerson states, moreover, that the value of the maximum flow from *s* to *d* is equal to the value of a minimum cut-set separating *s* from *d*. Such a cut-set is revealed by the labeling procedure: it is composed of all the arcs whose initial vertices are labeled and final vertices are unlabeled; the sum of the capacities of these arcs defines what is called the capacity of the cut-set. The arcs of the cut-set are necessarily saturated and each path connecting *s* and *d* necessarily includes one arc from the cut-set.

3.2. Analysis of the minimum cut-set of network N

Given a maximum flow, the minimum cut-set of network **N**, revealed by the Ford-Fulkerson labeling procedure, contains the following arcs:

(i) entering arcs whose final vertices are unlabeled – their total capacity is equal to x,

(ii) incompatibility arcs whose initial vertices are labeled and final vertices are unlabeled – their total capacity is equal to z; the initial vertices of these saturated arcs are called *saturated incompatibility groups*,

(iii) leaving arcs whose initial vertices are labeled – their total capacity is equal to μ; the initial vertices of these saturated arcs are called *saturated processing units*.

Observe that there are no directly nor indirectly connecting arcs in the minimum cut-set since they have an unlimited capacity.

The total capacity of saturated arcs (i), (ii) and (iii) is equal to $x+z+\mu$. Thus, according to the theorem of Ford and Fulkerson,

$$x + z + \mu = n - \rho. \tag{3.1}$$

The number of tasks that it is impossible to assign, given the imposed constraints, is equal to

$$\rho = n - x - z - \mu.$$

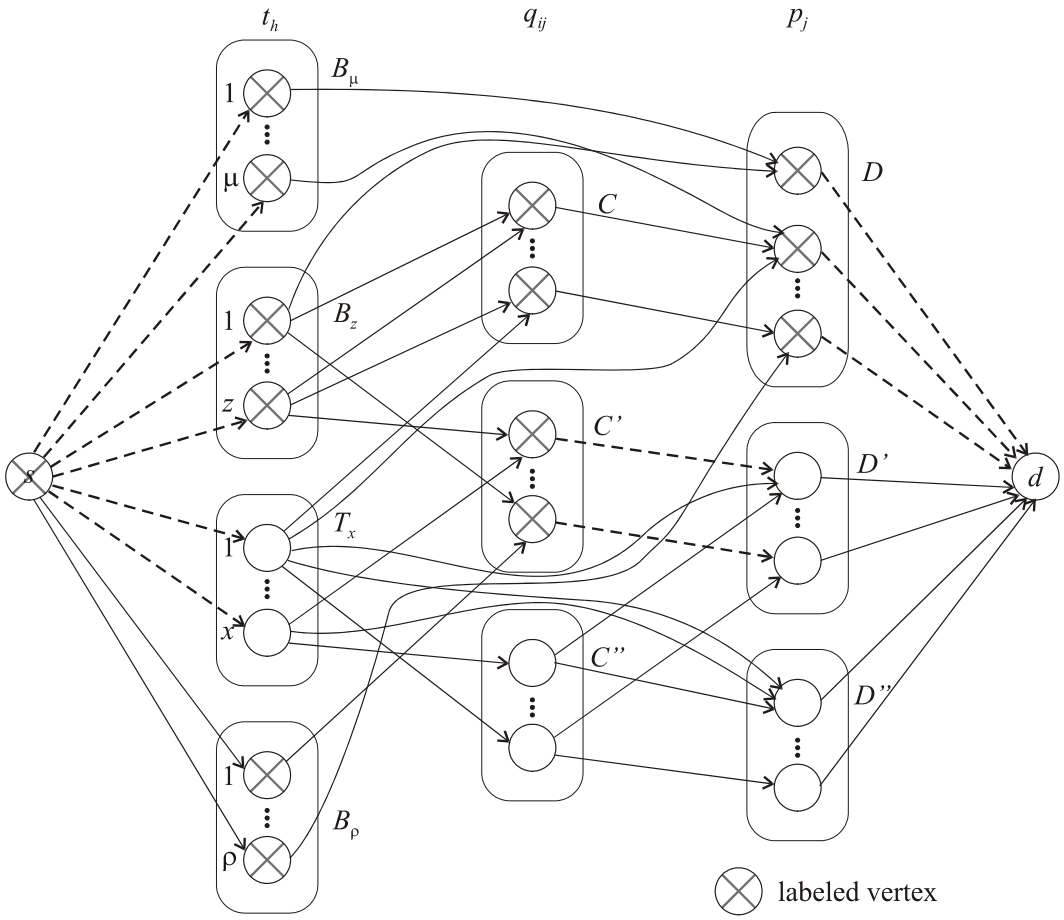


Fig. 2. Example of partition of sets T, Q, P

To understand why the flow cannot be greater than $n - \rho$, it is useful to introduce the following notation.

One can distinguish the following subsets of tasks in the set T:

T_x – set of x unlabeled tasks t_h (final vertices of arcs (i) in the minimum cut-set),

B_z – set of z tasks t_h processed via z saturated incompatibility groups (initial vertices of arcs (ii) in the minimum cut-set),

B_μ – set of μ tasks t_h processed by saturated processing units (initial vertices of arcs (iii) in the minimum cut-set),

B_ρ – set of ρ non-assigned tasks t_h,

$B = B_z \cup B_\mu \cup B_\rho$ – set of labeled tasks.

It follows from (3.1) that T_x, B_z, B_μ and B_ρ are four disjoint sets. Therefore, B_z, B_μ, B_ρ is a partition of B.

The set Q of incompatibility groups can be partitioned into the following subsets:

C – set of labeled incompatibility groups q_{ij} connected to labeled processing units p_j,

C' – set of z labeled incompatibility groups q_{ij} connected to unlabeled processing units p_j (saturated incompatibility groups being initial vertices of arcs (ii) in the minimum cut-set),

C'' – set of unlabeled incompatibility groups q_{ij}.

The set P of processing units can be partitioned into the following subsets (see Fig. 2):

D – set of labeled processing units p_j with total capacity of outgoing arcs equal to μ (saturated processing units being initial vertices of arcs (iii) in the minimum cut-set),

D' – set of unlabeled processing units p_j having at least one predecessor in C'; observe that $\text{card}(D') \leq z$, and $p_j \in D'$ can also have some predecessors in C'' and in T_x, but not in B or in C, because otherwise it would be labeled,

D'' – set of unlabeled processing units p_j having all their predecessors in C'' or in T_x.

3.3. The concept of the blocking configuration

Definition 1: The triplet $<B, C', D>$ revealed by the Ford-Fulkerson labeling procedure, associated with a maximum flow in **N**, is called a *blocking configuration*.

This definition is justified by the result given below.

Theorem 1: Consider a maximum flow in **N** and the associated blocking configuration <B, C', D>. The following statements are true:

1. From among the n-x tasks of B, μ tasks can be processed by saturated processing units $p_j \in D$, and z other tasks by processing units $p_j \notin D$ via z saturated incompatibility groups $q_{ij} \in C'$. No other processing unit p_j can process any of the n-x tasks. Thus, the maximum number of tasks $t_h \in B$ that can be assigned to processing units from P is equal to $\sum_{j \in \{j : p_j \in D\}} M_j + \text{card}(C') = z + \mu$.
2. The blocking configuration <B, C', D> is independent of the maximum flow used to reveal this configuration.

Proof of statement 1: No processing unit $p_j \in D''$ is adjacent to a task $t_h \in B$ because, otherwise, it would be labeled. For the same reason, no incompatibility group $q_{ij} \in C''$ of a processing unit $p_j \in D''$ is adjacent to a task $t_h \in B$. Thus, no processing unit $p_j \in D''$ can process directly nor indirectly a task $t_h \in B$, so the only processing units that could do it are in D and D'. Moreover, for the same reason as above, a processing unit $p_j \in D'$ is not adjacent to a task $t_h \in B$. If a processing unit $p_j \in D'$ is adjacent to an incompatibility group $q_{ij} \in C''$, then this group is not adjacent to a task $t_h \in B$ because, otherwise, it would be labeled. Thus, the processing unit $p_j \in D'$ can process a task $t_h \in B$ only indirectly via an incompatibility group $q_{ij} \in C'$. As there are z incompatibility groups in C', the processing units $p_j \in D'$ cannot process more than z tasks $t_h \in B$, whatever their total capacity. The total capacity of processing units $p_j \in D$ is limited by $\sum_{j \in \{j : p_j \in D\}} M_j = \mu$ because all the leaving arcs outgoing $p_j \in D$ are saturated. Consequently, the processing units from P can process at most $z+\mu$ tasks $t_h \in B$. This can be done exclusively by the processing units $p_j \in D$ or by those processing units which have an incompatibility group in C'. □

Proof of statement 2: Let us start by proving[2] the following lemma.

Lemma: The set of vertices labeled by the procedure of Ford and Fulkerson is independent of the maximum flow that is used to perform this labeling.

Proof of the lemma: Each subset of vertices $M \subset V$, such that $s \in M$ and $d \notin M$, defines a cut-set $\Omega(M)$ composed of arcs (x,y) such that $x \in M$ and $y \notin M$. Let $K[\Omega(M)]$ denote the capacity of this cut-set.

[2] This result has possibly been proved already, however, we did not find any trace of that in classic books. We wish to thank Daniel Vanderpooten who gave us the basis of the following demonstration, shorter and more elegant than the one we have initially made.

Let us consider two distinct maximum flows Φ_1 and Φ_2. Let also M_1 denote the set of vertices labeled with Φ_1, and M_2, the set of vertices labeled with Φ_2. Being maximum, these flows have the same capacity equal to m, so that:

$$m = K[\Omega(M_1)] = K[\Omega(M_2)]. \tag{3.2}$$

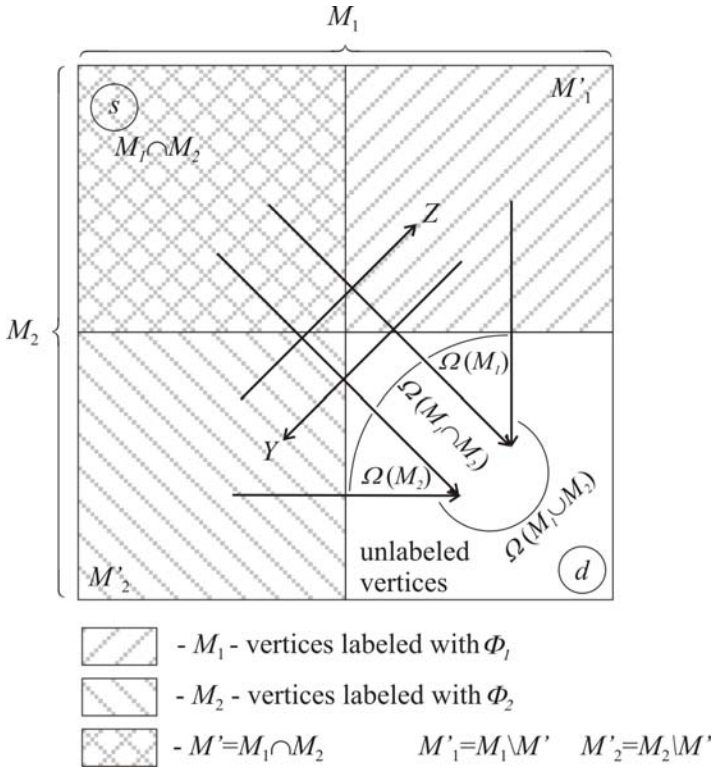


Fig. 3. Graphical support for the proof of the lemma

The subsets $M_1 \cap M_2$ and $M_1 \cup M_2$ include s but not d. Thus, their associated cut-sets verify:

$$K[\Omega(M_1 \cap M_2)] \geq m, \;\; K[\Omega(M_1 \cup M_2)] \geq m. \tag{3.3}$$

Let $M'_1 = M_1 \backslash M_2$ and $M'_2 = M_2 \backslash M_1$. Using these subsets one can define the two following subsets of arcs:

- Y, the set of arcs (x,y) such that $x \in M'_1$, $y \in M'_2$,
- Z, the set of arcs (y,x) such that $y \in M'_2$, $x \in M'_1$.

It is sufficient to analyze the subsets of arcs belonging to the cut-sets $\Omega(M_1)$, $\Omega(M_2)$, $\Omega(M_1 \cap M_2)$, $\Omega(M_1 \cup M_2)$ (see Fig. 3) to verify that:

$$K[\Omega(M_1)] + K[\Omega(M_2)] = K[\Omega(M_1 \cap M_2)] + K[\Omega(M_1 \cup M_2)] + K(Y) + K(Z)$$

and

$$K[\Omega(M_1)] + K[\Omega(M_2)] \geq K[\Omega(M_1 \cap M_2)] + K[\Omega(M_1 \cup M_2)]. \qquad (3.4)$$

It follows from (3.2), (3.3) and (3.4) that:

$$K[\Omega(M_1 \cap M_2)] = K[\Omega(M_1 \cup M_2)] = m.$$

$\Omega(M_1 \cap M_2)$ is thus a minimum cut-set. In consequence, independently of the flow considered (Φ_1 or Φ_2), once the vertices $M_1 \cap M_2$ are labeled, any other vertex cannot be labeled. We have thus $M_1 = M_2$. □

The lemma being proved, one can derive immediately that set B of labeled tasks does not depend on the maximum flow considered. It is so for the set D of labeled processing units and for the set $C \cup C'$ of labeled incompatibility groups. A labeled incompatibility group is in C' if and only if its unique successor in D' is not labeled. This is also independent of the maximum flow considered. This ends the proof of statement 2. □

Corollary:

1). The partition [C, C', C''] of incompatibility groups and the partition [D, D', D''] of processing units do not depend on the maximum flow considered.

2). The x tasks not belonging to B are assigned to processing units in each maximum flow; they can only be processed by processing units from $D' \cup D''$; if a processing unit $p_j \in D'$ is able to process one of these tasks, then this task can be assigned to p_j only directly or indirectly through incompatibility group $q_{ij} \in C''$ such that $Q_{ij} \in T_x$.

3). The processing units from $D \cup D'$ are the only ones that can process the tasks from set B; if a processing unit $p_j \in D'$ is able to process one of these tasks, then this task can be assigned to p_j only indirectly through incompatibility group q_{ij} such that $Q_{ij} \subset B$; moreover, in each maximum flow, one task of each such incompatibility group is assigned to the processing unit $p_j \in D'$; this means that this processing unit cannot process more tasks from B.

4). The partition [B_μ, B_z, B_ρ] of B is not independent of the maximum flow considered.

Proof of the corollary:

1). This result follows immediately from the second statement of the theorem.

2). Whatever the maximum flow considered, the only tasks being labeled are in B, thus the tasks from T_x cannot be labeled. Let $t_h \in T_x$. The arc (s, t_h) is necessarily saturated because otherwise t_h would be labeled. This means that t_h is assigned to a processing unit. This

processing unit is certainly not from set D because $p_j \in D$ are labeled, so t_h would also be labeled. Thus, only the processing units from $D' \cup D''$ are able to process t_h. Consider any processing unit $p_j \in D'$; since this processing unit is unlabeled, it can process t_h directly. Now, consider an incompatibility group Q_{ij} represented by vertex q_{ij} ; if q_{ij} is labeled ($q_{ij} \in C'$), then $t_h \notin Q_{ij}$. As soon as Q_{ij} contains at least one task from B, q_{ij} is labeled. Thus, if $t_h \in Q_{ij}$, then $Q_{ij} \in T_x$.

3). If $p_j \in D''$ would be able to process directly a task from set B, then p_j would be labeled because all tasks from B are labeled. If p_j would be able to process indirectly a task from set B, then the corresponding incompatibility group would be labeled. $p_j \in D''$ has, however, no predecessor among labeled incompatibility groups. For the same reason, $p_j \in D'$ cannot process directly any task from set B. If it is able to process one task from B through incompatibility group q_{ij}, then one task of this incompatibility group is necessarily assigned to p_j because, q_{ij} being labeled and p_j being not, the arc (q_{ij}, p_j) is saturated. Thus, $p_j \in D'$ is processing a number of tasks from B equal to the number of all its incompatibility groups q_{ij}, $i=1,\ldots,u_j$. In any flow, it cannot process more.

4). The following example is sufficient to show that the partition $[B_\mu, B_z, B_\rho]$ of B is not independent of the maximum flow considered.

Let $T=\{t_1, t_2, t_3\}$ and $P=\{p_1, p_2\}$ such that:

t_1 can only be processed by p_1 directly,

t_2 can be processed by p_1 directly and by p_2 indirectly, because of the 2-incompatibility group $Q_{12}=\{t_2, t_3\}$,

t_3 can only be processed by p_2 indirectly, because of the 2-incompatibility group Q_{12},

p_1 and p_2 can process one task each.

Consider two maximum flows presented in Fig. 4, giving the following assignments:

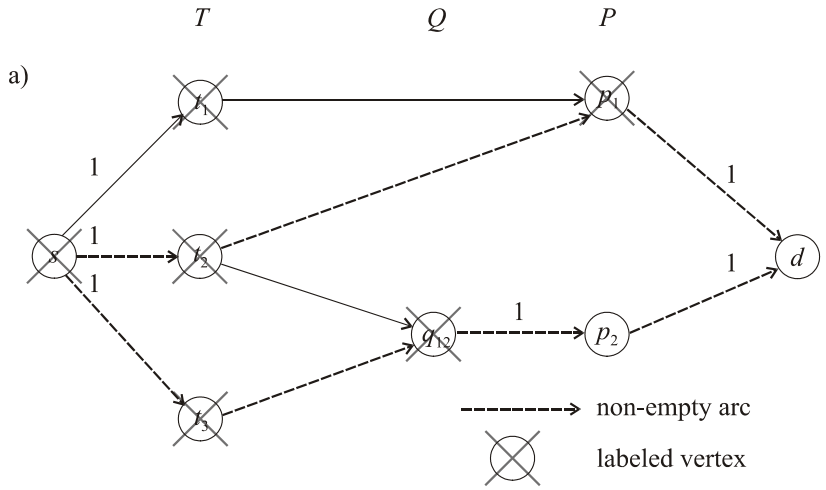
a) t_1 is assigned to p_1 and t_2 is assigned to p_2 through incompatibility group q_{12}, which results in

$$B_\mu=\{t_1\}, B_z=\{t_2\}, B_\rho=\{t_3\},$$

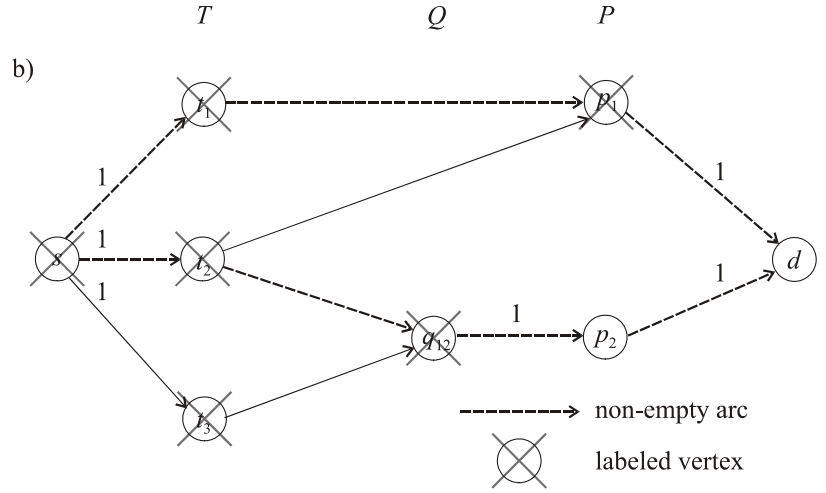
b) t_2 is assigned to p_1 and t_3 is assigned to p_2 through incompatibility group q_{12}, which results in

$$B_\mu=\{t_2\}, B_z=\{t_3\}, B_\rho=\{t_1\}.$$ □

The example presented in point 6.1 illustrates all properties connected with the concept of a blocking configuration.



$B=T$, $C'=\{q_{12}\}$, $D=\{p_1\}$, $D'=\{p_2\}$
$B_\mu=\{t_2\}$, $B_z=\{t_3\}$, $B_\rho=\{t_1\}$



$B=T$, $C'=\{q_{12}\}$, $D=\{p_1\}$, $D'=\{p_2\}$
$B_\mu=\{t_1\}$, $B_z=\{t_2\}$, $B_\rho=\{t_3\}$

Fig. 4. For alternative maximum flows, the labeling does not change but B_μ, B_z and B_ρ do change

3.4. Actions of unblocking

What action should be made in order to relax the imposed constraints such that one of the tasks $t_h \in B_\rho$ and, progressively, all the tasks, can be assigned to the processing units?

We suppose that the only possible modifications of initial data aiming at processing one additional task are of the following types:

1° Augment the capacity of a processing unit p_j.

2° Allow a processing unit p_j to process directly a task t_h that it was unable to process, neither directly nor indirectly.

3° Allow a processing unit p_j to process indirectly a task t_h that it was unable to process, neither directly nor indirectly, putting t_h in one of incompatibility groups of p_j.

4° Allow a processing unit p_j to process directly a task t_h that it was able to process indirectly only.

It follows that any action able to unblock the impossibility of assigning one additional task, consists in making one or more modifications of the above types. The modifications 2°, 3°, 4° are mutually exclusive, however, each of them can be combined with the first modification. In order to augment the assignment of tasks to processing units, one has to consider these modifications with respect to each maximum flow.

The above four modifications correspond to the following four modifications of the network **N**.

I° Augment the capacity of a leaving arc.

II° Add a directly connecting arc (t_h, p_j).

III° Add an indirectly connecting arc (t_h, q_{ij}).

IV° Replace the indirectly connecting arc (t_h, q_{ij}) by a directly connecting arc (t_h, p_j).

Taking into account the four possible modifications, we propose a set of six actions of unblocking and we prove that they are effective and the only ones that can increase by one the number of tasks assigned to processing units. The first four actions operate without changing the initial assignment (maximum flow) that revealed the blocking configuration, while the last two actions use the possibility of changing the initial assignment.

Action 1: (*use of modification 1°*) Augment by one the capacity of any saturated processing unit $p_j \in D$.

Proof of unblocking: The increase of M_j by one permits to label vertex d and thus to augment the maximum flow by one. Consequently, one more task $t_h \in B$ can be assigned to the processing unit p_j. □

Action 2: (*use of modification 2° or 3°*) Consider a non-saturated processing unit p_j ($p_j \notin D$) and a task $t_h \in B$, such that p_j was unable to process t_h neither directly nor indirectly; allow p_j to process

t_h either directly or indirectly through an incompatibility group q_{ij} such that no task of this incompatibility group is assigned to p_j.

Proof of unblocking: Case 1 – action corresponding to modification 2°; it adds a directly connecting arc (t_h, p_j), thus $p_j \in D' \cup D''$ can be labeled; p_j being non-saturated, also d can be labeled. Case 2- action corresponding to modification 3°; it adds an indirectly connecting arc (t_h, q_{ij}); since no task of q_{ij} is assigned to p_j, then $q_{ij} \in C''$ and the arc (q_{ij}, p_j) is empty; consequently, q_{ij}, p_j and d can be labeled. □

Action 3: (*use of modification 4°*) Allow a non-saturated processing unit p_j ($p_j \notin D$) to process directly a task $t_h \in B$ that it was able to process indirectly only.

Proof of unblocking: This action substitutes an indirectly connecting arc (t_h, q_{ij}) by a directly connecting arc (t_h, p_j). If the arc (t_h, q_{ij}) is empty, then its deletion keeps the initial maximum flow unchanged; since t_h is labeled, then p_j and d can also be labeled. If the arc (t_h, q_{ij}) is not empty, then the flow passing through (t_h, q_{ij}) and (q_{ij}, p_j) can be transferred to the arc (t_h, p_j), so the value n-ρ of the flow remains unchanged; as above, p_j and d can then be labeled. □

Action 4: (*use of modification 1° in conjunction with 2° or 3° or 4°*) Apply action 2 or action 3 with respect to a saturated processing unit $p_j \notin D$, after augmenting its capacity by one.

Proof of unblocking: The increase of the capacity of a saturated $p_j \notin D$ by one, boils down this action to action 2 or 3. □

Action 5: (*use of modification 3°*) Suppose that for the initial flow there exists a non-saturated processing unit p_j ($p_j \notin D$) such that at least one task $t_k \notin B$ is assigned to p_j through one of its incompatibility groups q_{ij} ($q_{ij} \in C''$), $i \in \{1,\ldots,u_j\}$. If there exists another flow of the same value n - ρ, such that p_j is non-saturated and no task of the incompatibility group q_{ij} is assigned to p_j, then allow p_j to process indirectly any $t_h \in B$ (that p_j was unable to process neither directly nor indirectly) through the incompatibility group q_{ij}.

Proof of unblocking: This action adds the arc (t_h, q_{ij}). Considering the other flow, q_{ij} can be labeled; since the arc (q_{ij}, p_j) is now empty, p_j can be labeled; p_j being non-saturated, also d can be labeled. □

Action 6: (*use of modification 2° or 3°*) Suppose that for the initial flow there exists a saturated processing unit $p_j \notin D$ such that at least one task $t_k \notin B$ is assigned to p_j. If there exists another flow of the same value n - ρ, where p_j is non-saturated because it has assigned one task $t_h \notin B$ less, then apply (if possible) action 2 or action 3 or action 5 with respect to the latter flow.

Proof of unblocking: As soon as action 2 or 3 or 5 is applied to an alternative flow meeting the required conditions, the unblocking is evident. □

Theorem 2: If it was impossible to assign all the tasks to the processing units, a successful application of any one of the six actions, involving the four possible modifications of the input data, permits to assign one additional task. These six actions are sufficient to exploit all possibilities offered by the four modifications of the input data when one is looking to assign one additional task.

Proof: The efficiency of each action has been demonstrated, so it is sufficient to prove that these actions exploit all possibilities offered by the four modifications of the input data, taken separately or in combination, in order to assign one additional task. First, we will demonstrate that all possibilities of assigning one additional task without changing the initial assignment that revealed the blocking configuration, are taken into account in actions 1 to 4. Then, we will demonstrate that all additional possibilities offered by an alternative assignment are taken into account in actions 5 and 6.

a). Assigning one additional task with respect to the initial assignment.

One is considering here the initial assignment of $n - \rho$ tasks obtained by the procedure for finding a maximum flow in network **N**. Given the $n - \rho$ initial assignments, one is interested in ways of using modifications 1° to 4° to get an additional assignment. In order to augment the initial flow by one, the modification of the input data should permit (given the initial maximum flow) to label the destination d. This is possible only if the modification leads to one of two following transformations:

i). An arc (p_j, d), with p_j initially labeled, changes from saturated to non-saturated: this transformation can only be obtained by modification 1° used in conformity with action 1.

ii). A processing unit p_j, that was initially unlabeled, becomes labeled and the arc (p_j, d) remains or becomes non-saturated: we will demonstrate that this transformation can be obtained by actions 2, 3 or 4 only.

Suppose first that before modification, the arc (p_j, d) is non-saturated. With the initial flow, it will continue to be non-saturated whatever modification or combination of modifications of the input data will be used. In order to get the required transformation, it is necessary to use one of the modifications from 2° to 4°. With modification 2° or 3°, the labeling of p_j takes place only if task t_h concerned by one of these modifications is labeled. Moreover, in case of modification 3°, it is necessary for the labeling of p_j that

the j-th incompatibility group concerned by this modification is such that the arc (q_{ij}, p_j) is non-saturated. Thus, the required transformation can only be obtained with action 2. With modification 4°, the labeling of p_j takes place only if task t_h concerned by this modification is labeled, and this result can only be obtained with action 3.

Suppose now that the arc (p_j, d) is saturated initially. In order to make it non-saturated, it is necessary to use modification 1° and then, only one of the modifications 2°, 3° or 4° can permit the labeling of p_j. It follows that the required transformation can only be obtained with action 4.

b). Assigning one additional task with respect to an alternative assignment.

Application of actions 1 to 4 to the alternative assignment (maximum flow) can exhibit the possibilities of unblocking that it was impossible to see with the initial assignment. We have to prove now that all additional possibilities offered by an assignment being alternative to any initial assignment are taken into account in actions 5 and 6.

Let us recall (see point 1) of the Corollary) that, independently of the maximum flow considered in actions 1 to 4, the vertices labeled and unlabeled will be the same. For an alternative flow, to pass through a modified network an additional unit of flow, that actions 1 to 4 applied to the initial flow were unable to do, it is necessary that there exists:

- either an arc (q_{ij}, p_j) saturated in the initial flow and non-saturated in the alternative one,
- or an arc (p_j, d) saturated in the initial flow and non-saturated in the alternative one.

Actions 5 and 6 concern the first and the second case, respectively. For each of them, they specify actions that can be applied to the alternative flow. Simple algorithms can be proposed to find an alternative flow meeting the required conditions or stating non-existence of such flow. □

4. The multi-criteria assignment problem

4.1. Definition of the family of criteria

The quality of feasible assignments defined in section 2 will be evaluated by three criteria:

- G_1 - the maximum dissatisfaction of tasks,
- G_2 - the total dissatisfaction of tasks,
- G_3 - the total cost of processing units.

The value of G_1 indicates what is the worst case in a given assignment, that is the maximum dissatisfaction among the n tasks assigned to processing units. G_2 measures the quality of assignment in terms of total dissatisfaction over all tasks. Finally, G_3 measures the quality of assignment in terms of total cost over all processing units.

All these criteria are to be minimized. Let us remark that the dissatisfaction criteria G_1 and G_2 are generally in conflict with the cost criterion G_3 because an assignment that accepts high dissatisfaction is usually less costly and vice versa. The conflict may also appear between individual dissatisfaction represented by G_1 and total dissatisfaction represented by G_2.

For the above reasons, except trivial cases, there is no assignment that would be minimal on all criteria, so one has to search for a *best compromise assignment* with respect to one's own preferences (see, e.g. (Roy, 1996)). This best compromise assignment should, moreover, be *non-dominated*, that is such that there could not exist another feasible assignment that would be better on at least one criterion and not worse on others.

4.2. Strategy for selection of the best compromise assignment

Let us remark that criterion G_1 plays a particular role in the family of criteria because it shows the maximum dissatisfaction in an assignment while G_2 and G_3 are integral measures indicating the total dissatisfaction and the total cost. It is thus reasonable to search for the best compromise among those assignments that are not worse on G_1 than some $\overline{G}_1$ and that are non-dominated with respect to G_2 and G_3. Parameterization of G_1 boils down the search for the best compromise to a bi-criteria selection problem. The non-dominated bi-criteria assignments can be shown conveniently on the plane in the system of co-ordinates corresponding to G_2 and G_3.

As to satisfaction of the parametric requirement $G_1 \leq \overline{G}_1$, it is sufficient to forbid all assignments of tasks t_h to processing units p_j for which $d_{hj} > \overline{G}_1$, h=1,…,n; j=1,…,m, and check if there exists at least one feasible assignment. If the answer if positive, one can pass to exploration of a set of non-dominated assignments on the plane ($G_2 \times G_3$). Otherwise, $\overline{G}_1$ should be increased to allow at least one feasible assignment; let G_1^* the corresponding value.

Remark that increasing progressively $\overline{G}_1$ from G_1^* through all values of $d_{hj} > G_1^*$ until $\max_{h,j}\{d_{hj}\}$, one gets all non-dominated assignments with respect to all three criteria.

5. Exploration of a set of non-dominated assignments on the plane ($G_2 \times G_3$)

5.1. The bi-criteria assignment problem with incompatibility constraints

Provided there exists at least one feasible assignment for $G_1 \leq \overline{G_1}$, the problem of finding the best assignment with respect to G_2 and G_3, called problem (P), can be formulated as a bi-criteria network flow problem in network **N'** (see Fig. 5) obtained from **N** (shown in Fig. 1) by the following transformation:

- the lower bounds on flows transmitted along all n entering arcs (s, t_h) are equal to one;
- each vertex corresponding to processing unit p_j (j=1,…,m) is followed by a new layer composed of M_j–m_j+1 vertices p_j^v (v=0,…,M_j–m_j); each new vertex p_j^v is connected by an ingoing arc with p_j and by an outgoing arc with d; the vertex p_j^0 corresponds to the regular capacity m_j of the processing unit p_j, while vertices p_j^v (v=1,…,M_j–m_j) correspond to each unit capacity of p_j, over m_j and up to M_j; a unit capacity of p_j is considered as a *copy* of the processing unit p_j; the capacities of new arcs are unlimited for (p_j, p_j^v) (j=1,…,m; v=0,…,M_j–m_j) and $\kappa(p_j^0, d)=m_j$, $\kappa(p_j^v, d)=1$ (j=1,…,m; v=1,…,M_j–m_j); the lower bound on all new arc flows is equal to zero;
- two types of costs are associated with unit flows through arcs of network **N'**:
  - costs of *processing type*, that is costs of unit flows through arcs (p_j^v, d); they are equal to 0 for (p_j^0, d), and to marginal cost $c_{j(m_j+v)}>0$ for (p_j^v, d), such that $c_{j(m_j+v)} \geq c_{j(m_j+v-1)}$ (j=1,…,m; v=1,…,M_j–m_j); these costs correspond to processing costs of p_j for its regular capacity (v=0) and for each additional unit capacity (v=1,…,M_j–m_j), respectively; the costs of the processing type on all other arcs are, obviously, equal to zero;
  - costs of *dissatisfaction type*, that is costs of unit flows through directly connecting arcs (t_h, p_j) and indirectly connecting arcs (t_h, q_{ij}), equal to d_{hj} (h=1,…,n, i=1,...,u_j, j=1,…,m); these costs correspond to dissatisfaction degrees of t_h for being assigned to p_j; the costs of the dissatisfaction type on all other arcs are, obviously, equal to zero.

The two criteria associated with feasible flow Φ in network **N'** are defined, respectively, as:

$$G_2 = \sum_{w' \in W'} \Phi(w') c(w')$$

$$G_3 = \sum_{w' \in W'} \Phi(w') d(w')$$

where W' is the set of arcs w' of **N'**, $\Phi(w')$ is a flow through arc w', $c(w')$ is a cost of the processing type for a unit flow through arc w', and $d(w')$ is a cost of the dissatisfaction type for a unit flow through arc w'.

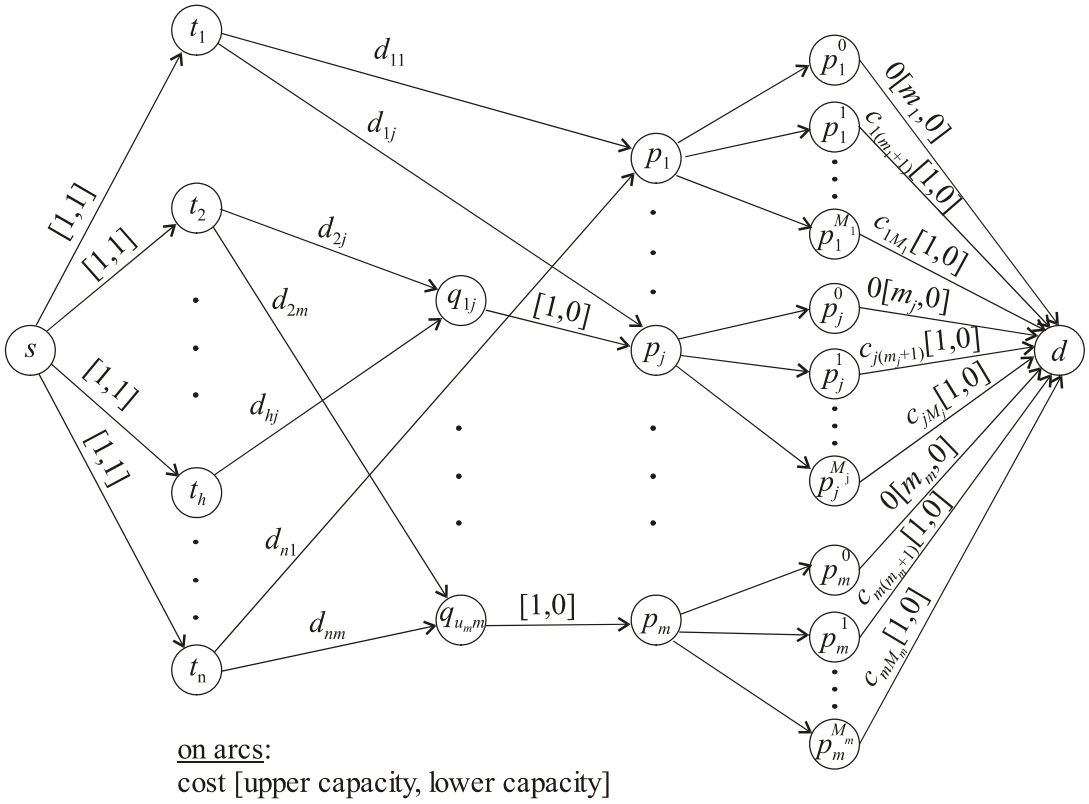


Fig. 5. The bi-criteria integer minimum cost flow problem in network **N'**

To verify that the bi-criteria network flow problem in network **N'** corresponds to our problem (P), the following remarks may be useful:

1. The non-zero costs of the processing type and the non-zero costs of the dissatisfaction type concern different arcs in **N'**, that is, if $c(w')\neq 0$ then $d(w')=0$ and vice versa.
2. As the marginal cost of all additional copies of p_j (j=1,…,m) is non-decreasing, it is impossible that for a feasible flow Φ minimizing G_3, and for any p_j, there is

$\Phi(p_j, p_j^v)=\Phi(p_j^v, d)=1$, while $\Phi(p_j, p_j^z)=\Phi(p_j^z, d)=0$ for some $z<v$; this proves that G_3 represents adequately the total cost of processing units.

Formally, a solution (flow) $\boldsymbol{x}^*$ of problem (P) is *non-dominated* if there does not exist any other feasible solution $\boldsymbol{x}$ such that $G_l(\boldsymbol{x}) \le G_l(\boldsymbol{x}^*)$, l=1,2, with at least one strict inequality. The set of all non-dominated solutions of problem (P) will be denoted by *ND*(P). Each non-dominated solution $\boldsymbol{x}^*$ has its image in the criterion plane ($G_2 \times G_3$), denoted by $G^{(*)}$. For the sake of simplicity, we will use the notion of solution with respect to both $\boldsymbol{x}^*$ and its image $G^{(*)}$, unless a distinction will be necessary to avoid misunderstanding.

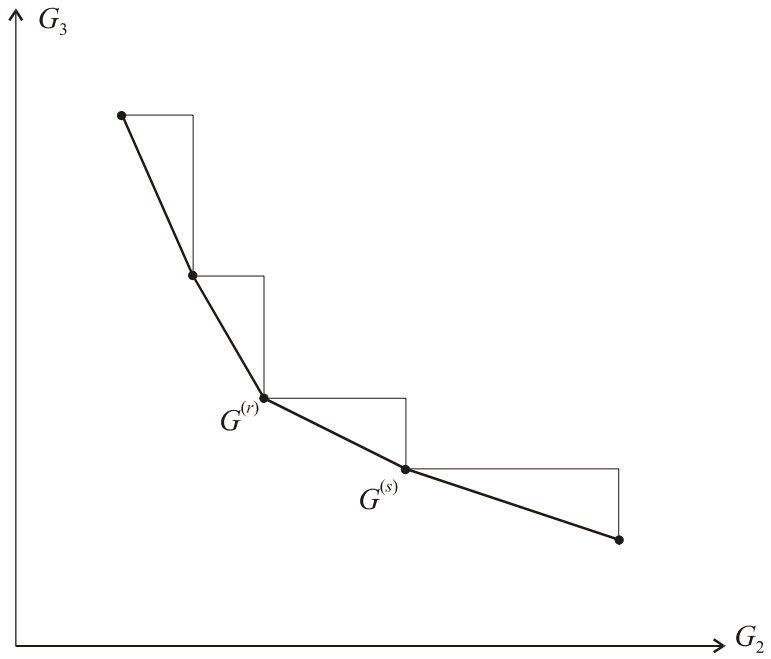


Fig. 6. Supported non-dominated solutions and potential regions $\Delta G^{(r)}G^{(s)}$ of non-supported non-dominated solutions of problem (P)

It is well known that in multi-criteria discrete optimization problems, thus also in case of our problem (P), it is necessary to distinguish two kinds of non-dominated solutions (see, e.g. (Ulungu & Teghem, 1994)):

- the set *S*(P) of *supported* non-dominated solutions which are optimal solutions of the parameterized single-objective problem (P_λ):

  Minimize $G_\lambda(\boldsymbol{x}) = \lambda\, G_2(\boldsymbol{x}) + (1-\lambda)\, G_3(\boldsymbol{x})$ (P_λ)

  subject to the constraints of problem (P) and $\lambda \in (0, 1)$.

- the set $NS(P) = ND(P) \setminus S(P)$ of *non-supported* non-dominated solutions which cannot be found by solving problem (P_λ) for any $\lambda \in (0, 1)$. These non-supported non-dominated solutions are necessarily located in the triangles $\Delta G^{(r)}G^{(s)}$ created in the criterion plane ($G_2 \times G_3$) by any two successive supported non-dominated points, as shown in Fig. 6.

In the following we show how to obtain the supported and the non-supported non-dominated solutions of problem (P) by exact methods. Specifically, in point 5.2, we use the bi-criteria network flow formulation of problem (P) in order to find all supported non-dominated solutions of (P), that is, optimal solutions of problem (P_λ). In order to find all non-supported non-dominated solutions of (P) we propose in point 5.4 to use the well known exact method of Ulungu and Teghem (1995), for a bi-criteria basic assignment problem, within a branch and bound procedure. As the method of Ulungu and Teghem (1995) works on a matrix representation of the bi-criteria basic assignment problem, we represent our problem (P) in matrix terms in point 5.3.

Let us mention that, recently, Sodeno-Noda and Gonzalez-Martin (2001), and Figueira (2002) have proposed general algorithms for the bi-criteria integer minimum cost flow problem. While the first one involves a classic tool of network flow problems, such as the network simplex method, the second one proposes a branch and bound method for searching both supported and non-supported non-dominated solutions of a bi-criteria integer minimum cost flow problem. We decided to use, however, an intermediate, two-stage approach: a parametric network flow algorithm in order to find all supported non-dominated solutions, and then, a branch and bound method involving a simple procedure for bi-criteria basic assignment with a systematic check of incompatibility constraints in order to find all non-supported non-dominated solutions. In this way we are able to better exploit the specific structure of our problem including the constraints of a basic assignment problem and the specific incompatibility constraints.

5.2. Searching for supported non-dominated solutions of problem (P)

In order to find an optimal solution of the parameterized problem (P_λ), we will solve the minimum cost network flow problem in network **N'**, where all the costs of the dissatisfaction type are multiplied by λ and all the costs of the processing type are multiplied by $(1-\lambda)$. The obtained integer flow equal to n minimizes the objective function $G_\lambda(\boldsymbol{x}) = \lambda\, G_2(\boldsymbol{x}) + (1-\lambda)\, G_3(\boldsymbol{x})$.

One can solve this network flow problem using a polynomial time algorithm, for example the simplex-based algorithm of Orlin (1997). It should be noticed, however, that if there exist alternative optimal solutions of problem (P_λ), they should all be discovered because they may

correspond to different supported non-dominated solutions of problem (P). Unfortunately, finding all alternative optima of problem (P_λ) may be time consuming.

Searching for supported non-dominated solutions of problem (P) proceeds as follows (see Ulungu and Teghem, 1995).

Let S denote a list of all supported non-dominated solutions already found. S is initialized by finding two non-dominated solutions of problem (P_λ), first for λ being very close to 1 (say 0.999) and then for λ being very close to 0 (say 0.001).

In S, solutions are ordered according to increasing value of G_2. Let $\{G^{(r)}, G^{(s)}\}$ denote two consecutive solutions in S. Thus, $G_2^{(r)} < G_2^{(s)}$ and $G_3^{(r)} > G_3^{(s)}$. In order to find new supported non-dominated solutions, one has to find all optimal solutions of problem (P_λ) with the following value of λ:

$$\lambda = \frac{G_3^{(r)} - G_3^{(s)}}{\left(G_2^{(s)} - G_2^{(r)}\right) + \left(G_3^{(r)} - G_3^{(s)}\right)}.$$

Let $\{G^{(t)}, t=1,\ldots,y\}$ be a set of optimal solutions obtained in this way, ordered according to increasing value of G_2. Two possible cases can arise:

- $\{G^{(r)},G^{(s)}\} \cap \{G^{(t)}, t=1,\ldots,y\} = \varnothing$; then $G^{(t)}$, $t=1,\ldots,y$, are new supported non-dominated solutions to be put in S; if $y>2$, then in further steps it will be necessary to consider the pairs $\{G^{(r)}, G^{(1)}\}$ and $\{G^{(y)}, G^{(s)}\}$, while it will be possible to skip other pairs $\{G^{(t)}, G^{(t+1)}\}$, $t=2,\ldots,y-1$;
- $\{G^{(r)},G^{(s)}\} \subset \{G^{(t)}, t=1,\ldots,y\}$; if $y>2$, then $G^{(t)}$, $t=2,\ldots,y-1$, are new supported non-dominated solutions to be put in S, however, they are not entering the pairs that should be further checked for existence of new supported non-dominated solutions.

After solving (P_λ) for all consecutive pairs of solutions from S that may yield new supported non-dominated solutions, the final list $S = S(P)$, that is, it contains all supported non-dominated solutions of problem (P).

5.3. Matrix representation of problem (P)

Problem (P) can also be represented as a generalized assignment problem, where assignments are done directly in a matrix whose rows correspond to tasks and columns to processing units. Each entry of the matrix is a cost of assignment of a task to a processing unit. The complete minimum

cost assignment corresponds to selection of n entries in the matrix that minimize the sum of the entries while observing the imposed constraints.

As we are considering two criteria, each of them will be associated with a different cost matrix: criterion G_2 with a matrix D of dissatisfaction type costs, and criterion G_3 with a matrix C of processing type costs. Both matrices are of the same dimension: $n \times \pi$, where $\pi = \sum_{j=1}^{m} M_j$. In this representation, the first M_1 columns of C and D correspond to copies of p_1 , the following M_2 columns to copies of p_2, etc.

The entries of matrix D are defined as: $D_{hi} = d_{h1}$ for $i=1,\ldots,M_1$, and $D_{hi} = d_{h2}$ for $i=M_1+1,\ldots,M_1+M_2$, and … $D_{hi} = d_{hm}$ for $i=\pi - M_m + 1,\ldots,\pi$, $h=1,\ldots,n$. Remark that dissatisfaction degrees of each particular task are identical for all copies of the same processing unit.

As the cost of each particular copy of a processing unit is the same for all tasks, the rows of matrix C are identical. Each row of matrix C is defined as: $C_h = [0,\ldots,0, c_{11},\ldots, c_{1(M_1-m_1)},0,\ldots,0, c_{21},\ldots, c_{2(M_2-m_2)},\ldots,0,\ldots,0, c_{m1},\ldots, c_{m(M_m-m_m)}]$, $h=1,\ldots,n$, where the number of zeroes is equal, respectively, to $m_1, m_2,\ldots, m_m$.

Our bi-criteria assignment problem is clearly a generalization of the basic assignment problem, because in the former there are two, instead of one, objective functions and additional incompatibility constraints have to be considered. The constraints of the basic assignment problem impose that each task is assigned to exactly one processing unit and vice versa. This implies consideration of a square cost matrix. Therefore, if $n<\pi$, in order to represent our assignment problem as a generalization of the basic assignment problem, one has to augment the matrices D and C to the dimension $\pi \times \pi$, by putting zeros in the empty rows. Of course, the assignments made in the rows $n+1,\ldots,\pi$ are dummy ones. As the costs of dummy tasks are all equal to zero, it is obvious that they do not affect the assignment of real tasks in the optimal solution.

We can use the matrix representation of problem (P) to give an equivalent formulation of (P) as a mathematical programming problem. We will give this equivalent formulation below in order to show the incompatibility constraints in terms of the matrix representation.

Let us define the decision variables as:

$$x_{hi} = \begin{cases} 1, \text{ if task } t_h \text{ has been assigned to } r\text{ - th copy of processor } p_j, \text{ where } r + \sum_{k=1}^{j-1} M_k = i, \\ 0, \text{ otherwise.} \end{cases}$$

$h=1,\ldots, \pi$, $i=1,\ldots, \pi$,

Problem (P) can be expressed as follows:

$$\text{Minimize} \quad \begin{bmatrix} G_2(\mathbf{x}) = \sum_{h=1}^{\pi} \sum_{i=1}^{\pi} D_{hi} x_{hi} \\ G_3(\mathbf{x}) = \sum_{h=1}^{\pi} \sum_{i=1}^{\pi} C_{hi} x_{hi} \end{bmatrix}$$

subject to $\sum_{i=1}^{\pi} x_{hi} = 1$, $h=1,\ldots,\pi,$

$$\sum_{h=1}^{\pi} x_{hi} = 1, \quad i=1,\ldots,\pi,$$

$$x_{hi} \in \{0, 1\}, \quad h=1,\ldots,\pi, \quad i=1,\ldots, \pi,$$

and the incompatibility constraints

$$\sum_{h \in H_{lj}} \sum_{i \in J_j} x_{hi} \le 1, \; J_j = \left\{ \sum_{k=1}^{j-1} M_k + 1, \ldots, \sum_{k=1}^{j} M_k \right\}, \; H_{lj} = \left\{ h : t_h \in Q_{lj} \right\}, \; j=1,\ldots,m, \; l=1,\ldots,u_j,$$

where Q_{lj} is the l-th incompatibility group of processing unit p_j, $\sum_{k=1}^{0} M_k = 0$, and $\boldsymbol{x}$ is the vector of decision variables defining the assignment.

Problem (P) is a bi-criteria 0-1 integer linear programming problem.

5.4. Searching for non-supported non-dominated solutions of problem (P)

After finding all supported non-dominated solutions contained in the list $S = S(\mathrm{P})$, one can proceed to the search of non-supported non-dominated solutions of problem (P). These solutions are necessarily located in the triangles $\Delta G^{(r)} G^{(s)}$ created in the criterion plane ($G_2 \times G_3$) by any two supported non-dominated solutions being successive on the list S (see Fig. 6 and 7).

Ulungu and Teghem (1995) proposed an exact method of searching for non-supported non-dominated solutions of a bi-criteria basic assignment problem. It examines systematically the interior of all triangles $\Delta G^{(r)} G^{(s)}$ using the well-known Hungarian method for optimization of some reduced assignment problems in the $\pi \times \pi$ matrix:

$$DC_\lambda = \lambda D + (1-\lambda) C,$$

where, for the triangle $\Delta G^{(r)} G^{(s)}$, $\lambda = \dfrac{G_3^{(r)} - G_3^{(s)}}{\left(G_2^{(s)} - G_2^{(r)}\right) + \left(G_3^{(r)} - G_3^{(s)}\right)}$. To set up a reduced assignment problem, Ulungu and Teghem use properties of dual variables of the linear relaxation of problem (P_λ) considered without the incompatibility constraints. Each time the Hungarian method is applied, it is necessary to obtain all alternative optimal solutions because they may correspond to different values of G_2, G_3. An enumeration procedure is thus needed to examine all feasible combinations of zero reduced costs of the linear relaxation problem, if the number of

such costs is greater than π. If the number of zero reduced costs is much greater than π, this enumeration procedure is time consuming.

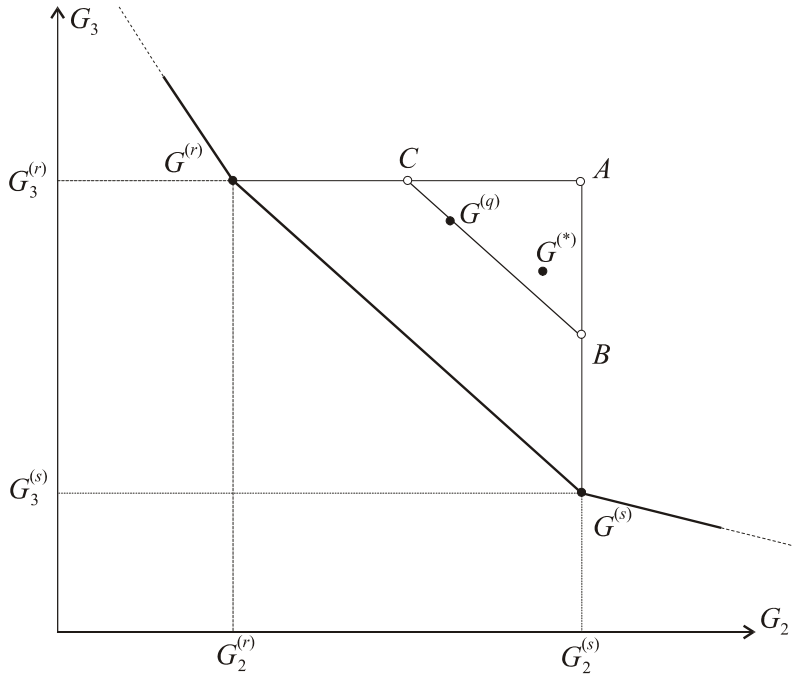


Fig. 7. Searching for non-supported non-dominated solutions of problem (P)

Each non-supported non-dominated solution resulting from the above procedure, say $G^{(q)}$, included in a triangle $\Delta G^{(r)}G^{(s)}$, has to be checked if it satisfies the incompatibility constraints. If it is the case, $G^{(q)}$ is included in set *NS*(P) of non-supported non-dominated solutions of problem (P). Otherwise, $G^{(q)}$ becomes a starting solution for a branch and bound procedure that tries to find a feasible solution of problem (P) located in the corresponding triangle $\Delta G^{(r)}G^{(s)}$. This procedure, called '*Explore NS*', is described below.

Procedure '*Explore NS*'

Step 1. Take a non-supported non-dominated solution obtained by the method of Ulungu and Teghem, say $G^{(q)}$, included in the triangle $\Delta G^{(r)}G^{(s)}$ and such that $G^{(q)}$ defines in the corresponding matrix DC_λ an assignment $\boldsymbol{x}^{(q)}$ which does not satisfy the incompatibility constraints.

Step 2. Let *CLUB* represent the current least upper bound on the optimal assignment in matrix DC_λ which satisfies the incompatibility constraints and belongs to the triangle $\Delta G^{(r)}G^{(s)}$. Set $CLUB = \lambda G_2^{(s)} + (1-\lambda)G_3^{(r)}$, that is to the value of the worst point in the triangle

$\Delta G^{(r)}G^{(s)}$ (point A in Fig. 7). Make solution $G^{(q)}$ and the corresponding assignment in matrix DC_λ the root of the branching tree. Let also $L(DC_\lambda)$ represent a lower bound on the optimal assignment in matrix DC_λ which satisfies the incompatibility constraints and belongs to the triangle $\Delta G^{(r)}G^{(s)}$. Set $L(DC_\lambda)=G_\lambda(\boldsymbol{x}^{(q)})$.

Step 3. Identify the incompatibility constraints that are not satisfied by solution $G^{(q)}$, that is such $j^* \in \{1,\ldots,m\}$ and $l^* \in \{1,...,u_j\}$ that $\sum_{h \in H_{l^* j^*}} \sum_{i \in J_{j^*}} x_{hi}^{(q)} > 1$. Let $\min_{j^*,l^*} \left\{ \sum_{h \in H_{l^* j^*}} \sum_{i \in J_{j^*}} x_{hi}^{(q)} \right\} = k > 1$ and denote the corresponding j^* by ϕ, and l^* by ψ.

Step 4. Branch into k assignment subproblems and set for subproblem 1, matrix element $DC_\lambda^1\left(h^1, i^1\right) = \infty$, for subproblem 2, $DC_\lambda^2\left(h^2, i^2\right) = \infty$, and so on, until subproblem k, $DC_\lambda^k\left(h^k, i^k\right) = \infty$. Setting a matrix element on infinity means that the corresponding assignment is forbidden. The co-ordinates of the assignments forbidden in particular subproblems are different and numbered pairs of indices (h^1,i^1), (h^2,i^2), …, (h^k,i^k) for which $x_{hi}^{(q)} = 1,\ h \in H_{\psi\phi},\ i \in J_\phi$.

Step 5. Solve the k new assignment problems without the incompatibility constraints by the Hungarian method. Each optimal solution is a lower bound $L\left(DC_\lambda^f\right)$, f=1,…,k, for the corresponding subproblem. If for a subproblem there are alternative optima, they should all be identified.

Step 6. If there are one or more solutions from *step* 5 that satisfy the incompatibility constraints and are located in the triangle $\Delta G^{(r)}G^{(s)}$, and if the smallest $L\left(DC_\lambda^f\right)$ for these solutions, say $L\left(DC_\lambda^*\right)$, is smaller than *CLUB*, set *CLUB*= $L\left(DC_\lambda^*\right)$ and save the corresponding solution $G^{(*)}$. Otherwise, *CLUB* remains unchanged.

Step 7. If *CLUB* is less than the lower bounds on all unexplored subproblems, then solution $G^{(*)}$ corresponding to *CLUB* is a candidate to be included in set *NS*(P), so stop; otherwise, go to *step* 8. By unexplored subproblems, we mean subproblems, corresponding to solutions with non-satisfied incompatibility constraints, that have not been branched into further subproblems.

Step 8. From the set of all unexplored subproblems with lower bounds less than *CLUB*, select subproblem $G^{(\neq)}$ with the smallest lower bound for further branching. Substitute $G^{(q)}$ and assignment $\boldsymbol{x}^{(q)}$ by solution $G^{(\neq)}$ and assignment $\boldsymbol{x}^{(\neq)}$, and go to *step* 3.

The above branch and bound procedure needs some comments:

1. '*Explore NS*' solves the easier assignment problem that does not respect the incompatibility constraints and then systematically forbids assignments violating the incompatibility constraints until finally an optimal assignment satisfying the incompatibility constraints is obtained (if one exists in the triangle).

2. The branching into subproblems in *Step* 3 is organized such that each subproblem gets one forbidden assignment, from among assignments of one incompatibility constraint that was violated to a minimum extent in solution $G^{(q)}$, that is a incompatibility constraint having the minimum number $k>1$ of tasks assigned to processing units. Note that there is no need to check out the subproblems arising from other incompatibility constraints violated in solution $G^{(q)}$ (if there are any). It would also lead to the optimal solution, however, the search strategy of the solution space would be of the "*breadth-first-search*" type. Our strategy of branching for only one violated incompatibility constraint is of the "*depth-first-search*" type – it has been acknowledged to be more efficient and less memory-demanding.

3. If subproblems in *Step* 4 have many alternative optima then this step can be time-consuming. In practice, however, the initial interval between the current least upper bound *CLUB* and the lower bound on the optimal assignment provided by the Hungarian method is rather narrow which permits to eliminate many subproblems from further exploration.

4. The systematic search of the solution space by '*Explore NS*' guarantees that if for a given $G^{(q)}$ there exists a non-supported non-dominated solution of problem (P) in the triangle *ABC* (see Fig. 6), then it will be found.

5. If *CLUB* has not been changed during the branch and bound procedure, this means that no non-supported non-dominated solution of problem (P) can be found starting from $G^{(q)}$. Otherwise, solution $G^{(*)}$ corresponding to a final value of *CLUB* is included in *NS*(P) if it is not dominated by any other solution from this set; at the same time, the solutions from *NS*(P) dominated by $G^{(*)}$ are eliminated.

6. Numerical examples

6.1. Example with a blocking configuration

Let us consider the assignment of 5 tasks to 3 processing units with regular capacity equal to 1. The capacity of processing unit p_1 cannot be increased, while the capacity of p_2 and p_3 can be augmented by one. The processing costs of tasks by the processing units are given in Table 1.

Table 1. The processing costs of tasks by the processing units

	p_1	p_2	p_3
Until the regular capacity:	0	0	0
The first additional task:	∞	10	25

The dissatisfaction degrees of particular tasks for being assigned to particular processing units are given in Table 2.

Table 2. The dissatisfaction degrees of tasks for being assigned to processing units

Task vs. processing unit	p_1	p_2	p_3
t_1	∞	5	∞
t_2	4	∞	2
t_3	5	∞	∞
t_4	∞	3	6
t_5	∞	4	5

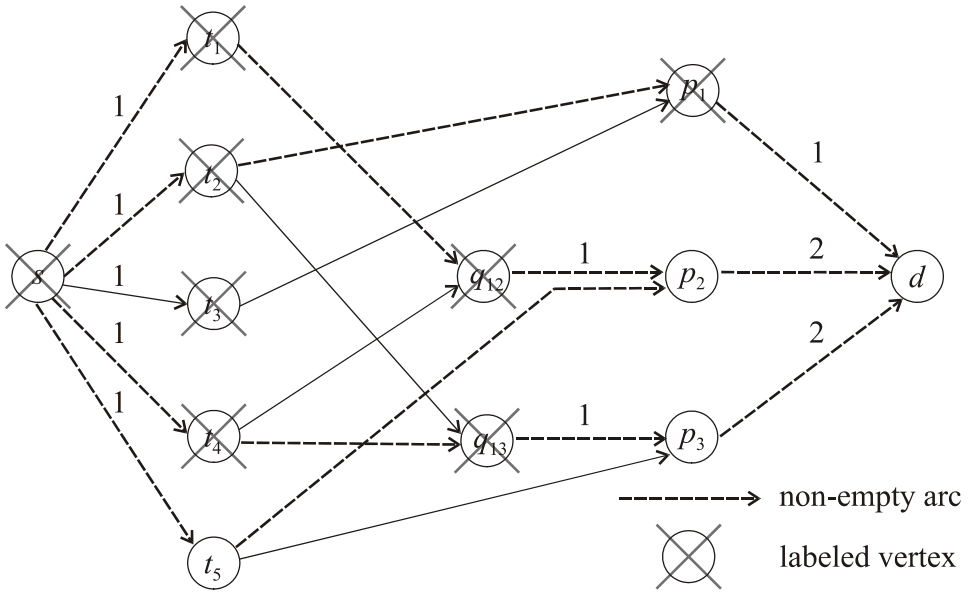


Fig. 8. Finding a feasible assignment as a problem of finding a maximum integer flow in network $\mathbf{N}^{\bullet}$

The assignment of tasks to processing units has to respect some incompatibility constraints. One incompatibility group is defined for processing unit p_2 and another for p_3; they are as follows:

$$Q_{12} = \{t_1, t_4\},\ Q_{13} = \{t_2, t_4\}.$$

The problem of finding a feasible assignment of tasks to processing units, defined above, can be formulated as a problem of finding a maximum integer flow in network $\mathbf{N}^{\bullet}$ shown in Fig. 8.

The maximum integer flow in network $\mathbf{N}^{\bullet}$ has value 4, which is less than the number of all tasks. Thus, the assignment problem has no feasible solution for the above constraints. There are 8 alternative flows of value 4, corresponding to one non-assigned task t_1, t_2, t_3 or t_4, combined with a possibility of assigning task t_5 either to p_2 or p_3; let us denote them by t_1-t_5/p_2, t_1-t_5/p_3, t_2-t_5/p_2, t_2-t_5/p_3, t_3-t_5/p_2, t_3-t_5/p_3, t_4-t_5/p_2, t_4-t_5/p_3, respectively; the flow t_3-t_5/p_2 is shown in Fig. 8.

We will show the blocking configuration for the maximum flow presented in Fig. 8. The characteristic sets introduced in point 3.2 are as follows:

$T_x=\{t_5\},\ B_\mu=\{t_2\},\ B_z=\{t_1, t_4\},\ B_\rho=\{t_3\},$

$C=\varnothing,\ C'=\{q_{12}, q_{13}\},\ C''=\varnothing,$

$D=\{p_1\},\ D'=\{p_2, p_3\},\ D''=\varnothing.$

The blocking configuration $<B, C', D>$ for any maximum flow in $\mathbf{N}^{\bullet}$ is equal to:

$$<\{t_1, t_2, t_3, t_4\},\ \{q_{12}, q_{13}\},\ \{p_1\}>$$

According to point 3.4, the following actions of unblocking are possible (the initial flow is t_3-t_5/p_2):

(a) action of type 1: augment by one the capacity of p_1,
(b) action of type 2: allow p_3 to process t_1 directly,
(c) action of type 2: allow p_3 to process t_3 directly,
(d) action of type 3: allow p_3 to process directly t_2 that it was able to process indirectly only,
(e) action of type 3: allow p_3 to process directly t_4 that it was able to process indirectly only,
(f) action of type 4: augment by one the capacity of p_2 and allow it to process t_2 directly,
(g) action of type 4: augment by one the capacity of p_2 and allow it to process t_3 directly,
(h) action of type 4: augment by one the capacity of p_2 and allow it to process directly t_1 that it was able to process indirectly only,
(i) action of type 4: augment by one the capacity of p_2 and allow it to process directly t_4 that it was able to process indirectly only,
(j) action of type 6 (action 2 for alternative flow t_2-t_5/p_3): allow p_2 to process t_2 directly,
(k) action of type 6 (action 2 for alternative flow t_3-t_5/p_3): allow p_2 to process t_3 directly,

(l) action of type 6 (action 3 for alternative flow t_1-t_5/p_3): allow p_2 to process directly t_1 that it was able to process indirectly only,

(m) action of type 6 (action 3 for alternative flow t_4-t_5/p_3): allow p_2 to process directly t_4 that it was able to process indirectly only.

Actions of type 5 are impossible because $C''=\varnothing$.

After performing any action from among (a) to (m), one gets a maximum flow equal to 5 corresponding to a feasible assignment. For example, action (d) leads to a the following assignment:

$$t_1 \rightarrow p_2$$
$$t_2 \rightarrow p_3$$
$$t_3 \rightarrow p_1$$
$$t_4 \rightarrow p_3$$
$$t_5 \rightarrow p_2$$

with maximum dissatisfaction of tasks G_1=6, total dissatisfaction of tasks G_2=22 and total cost of processing units G_3=35.

6.2. Example without blocking configuration

Let us consider another assignment problem with 4 tasks and 3 processing units whose regular capacity is equal to 1. The capacity of processing unit p_3 cannot be increased, while the capacity of p_1 and p_2 can be augmented by one. The processing costs of tasks by the processing units are given in Table 3.

Table 3. The processing costs of tasks by the processing units

	p_1	p_2	p_3
Until the regular capacity:	0	0	0
The first additional task:	∞	15	3

The dissatisfaction degrees of particular tasks for being assigned to particular processing units are given in Table 4.

The assignment of tasks to processing units has to respect some incompatibility constraints. One incompatibility group is defined for processing unit p_1 and another for p_3; they are as follows:

$$Q_{11} = \{t_1, t_3, t_4\},\ Q_{13} = \{t_2, t_4\}.$$

Table 4. The dissatisfaction degrees of tasks for being assigned to processing units

Task vs. processing unit	p_1	p_2	p_3
t_1	∞	1	3
t_2	10	5	12
t_3	∞	0	2
t_4	7	3	0

The maximum integer flow in the corresponding network is equal to 4, thus there exists at least one feasible assignment.

Suppose that the decision maker wants to analyze feasible assignments for which the maximum dissatisfaction of tasks $G_1 \leq \overline{G}_1 = 7$. According to point 4.2, it is then sufficient to forbid all assignments of tasks to processing units for which the dissatisfaction degree is greater than 7 and check if there exists at least one feasible assignment. As the answer if positive, one can pass to exploration of a set of non-dominated assignments on the plane ($G_2 \times G_3$).

According to point 5.3, the basic assignment problem can be explained on a matrix whose rows correspond to tasks and columns to processing units multiplied by their corresponding capacities. When the quality of feasible assignments is measured by G_2, the matrix is composed of dissatisfaction degrees and denoted by D. In the case of G_3, the matrix is composed of processing costs and denoted by C. Both matrices are of the same dimension: $n \times \pi$, where $n=4$ and $\pi = 5$. They are presented in Tables 5 and 6 (p_2' and p_3' correspond to the first copy of p_2 and p_3, respectively).

Table 5. Assignment matrix D of dissatisfaction degrees

	p_1	p_2	p_2'	p_3	p_3'
t_1	∞	1	1	3	3
t_2	∞	5	5	∞	∞
t_3	∞	0	0	2	2
t_4	7	3	3	0	0

Table 6. Assignment matrix C of processing costs

	p_1	p_2	p_2'	p_3	p_3'
t_1	∞	0	15	0	3
t_2	∞	0	15	∞	∞
t_3	∞	0	15	0	3
t_4	0	0	15	0	3

We wish to find the set ND(P) of all non-dominated solutions $\boldsymbol{x}$ of the bi-objective assignment problem (P) defined in point 5.1. These solutions correspond to non-dominated assignments with respect to G_2 and G_3, respecting the incompatibility constraints.

Using the method presented in point 5.2, one gets two supported non-dominated assignments, corresponding to λ=1 and λ=0, respectively, in the objective function of the parameterized single-objective problem (P_λ). These assignments, denoted by G^A and G^B, are shown in Table 7 and in Fig. 9.

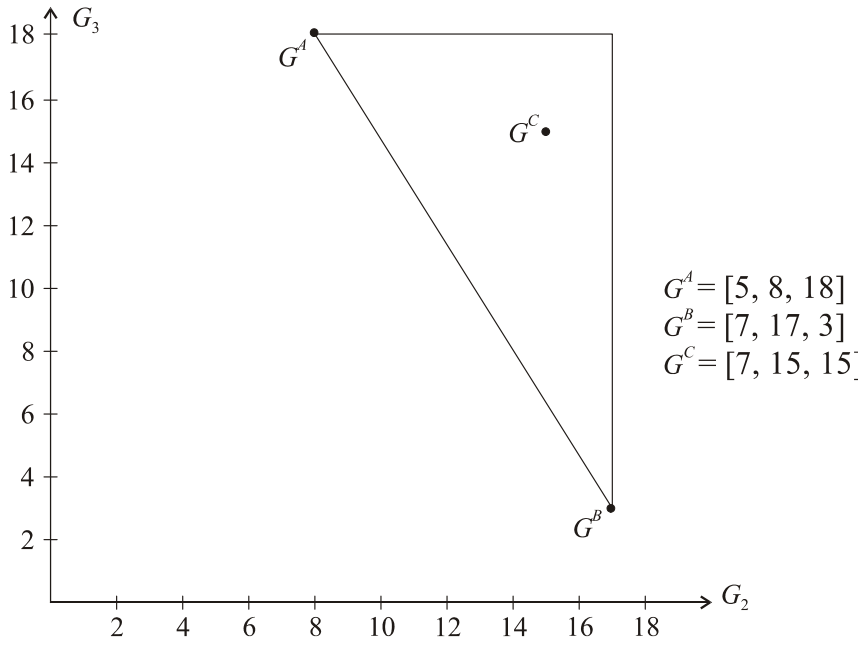


Fig. 9. Supported and non-supported non-dominated assignments for $G_1 \leq 7$

Let us mention that there exists an alternative solution to G^A, where assignment of t_1 to p_3 and t_3 to p_2 is permuted while giving the same value of the three criteria.

Table 7. Supported non-dominated assignments

Assignment G^A	Assignment G^B
$t_1 \rightarrow p_3$ $t_2 \rightarrow p_2$ $t_3 \rightarrow p_2$ $t_4 \rightarrow p_3$	$t_1 \rightarrow p_3$ $t_2 \rightarrow p_2$ $t_3 \rightarrow p_3$ $t_4 \rightarrow p_1$
$G_1^A = 5$ $G_2^A = 8$ $G_3^A = 18$	$G_1^B = 7$ $G_2^B = 17$ $G_3^B = 3$

Minimization of $G_\lambda(\boldsymbol{x}) = \lambda G_2(\boldsymbol{x}) + (1- \lambda)G_3(\boldsymbol{x})$, subject to constraints of problem (P), for $\lambda = \dfrac{G_3^A - G_3^B}{\left(G_2^B - G_2^A\right) + \left(G_3^A - G_3^B\right)} = 0.625$, gives no other supported non-dominated assignments.

Then, using the method of Ulungu and Teghem (1995) and the procedure '*Explore NS*', presented in point 5.4, non-supported non-dominated assignments are searched in the triangle $\Delta G^A G^B$. In result, one gets only one non-supported non-dominated assignment, denoted by G^C. It is presented in Table 8 and in Fig. 9.

Finally, $ND(\mathrm{P}) = \{G^A, G^B, G^C\}$.

Table 8. Non-supported non-dominated assignment

Assignment G^C
$t_1 \rightarrow p_3$ $t_2 \rightarrow p_2$ $t_3 \rightarrow p_2$ $t_4 \rightarrow p_1$
$G_1^C = 7$ $G_2^C = 15$ $G_3^C = 15$

7. Conclusions

We considered a generalized assignment problem where processing units can process more than one task at a time but processing of a task by some processing unit can exclude a simultaneous processing of some other tasks by this processing unit.

Three criteria were used for evaluation of feasible assignments. In the case where no feasible assignment exists, a blocking configuration was identified and all possible actions of unblocking were proposed. We demonstrated some interesting properties of the blocking configurations as well as exhaustiveness of the proposed actions of unblocking. In the case where there exist more than one feasible assignment that meets a constraint on the value of criterion G_1, we proposed an exact two-stage method for generating a set of non-dominated assignments with respect to criteria G_2 and G_3. In the first stage, a set of supported non-dominated assignments was searched by minimization of a parameterised linear combination of G_2 and G_3. For a given value of the parameter, the resulting single-objective minimization problem was solved as a problem of finding a minimum cost integer flow in a special network. In the second stage, non-supported non-dominated assignments were searched in the triangular area between all consecutive pairs of supported non-dominated assignments. This stage was organized in two steps; first, using the method by Ulungu and Teghem (1995) a non-supported assignment was found without considering the incompatibility constraints; if this assignment did not satisfy the incompatibility constraints, then it was used by a branch-and-bound procedure as a starting point to find a feasible non-supported assignment, or to conclude that it does not exist in the considered area.

As finding all supported and non-supported non-dominated solutions of the generalized assignment problem is computationally NP-hard, it might be useful in the future to apply a metaheuristic procedure for this problem.

The theoretical considerations were illustrated by two numerical examples: one with a blocking configuration and another with three feasible non-dominated assignments – two supported and one non-supported.

Acknowledgments

The second author wishes to acknowledge financial support from State Committee for Scientific Research (KBN), research grant no. 8T11F 006 19, and from the Foundation for Polish Science, subsidy no. 11/2001.

References

Figueira, J., 2002. "On the integer bi-criteria network flow problem: a branch-and-bound approach". *Cahier du LAMSADE*, University of Paris Dauphine, Paris (to appear).

Ford, L.R. and Fulkerson, D.R., 1962. "*Flows in Networks*". Princeton University Press, Princeton, NJ.

Orlin, J.B., 1997. "A polynomial time primal network simplex algorithm for minimum cost flows". *Mathematical Programming* 78, 109-129.

Roy, B., 1996. "*Multicriteria Methodology for Decision Aiding*". Kluwer Academic Publishers, Dordrecht.

Slowinski, R., 2001. "MASCOME : Multi-criteria Assignment Subject to Constraints Of Mutual Exclusion", *software system* developed in the Laboratory of Intelligent Decision Support Systems, Poznan University of Technology, Poznan.

Sedeno-Noda, A. and Gonzalez-Martin, C., 2001. "An algorithm for the biobjective integer minimum cost flow problem", *Computers & Operations Research* 28 (2), 139-156.

Ulungu, E.L. and Teghem, J., 1994. "Multi-objective combinatorial optimization problems: a survey". *Journal of Multiple-Criteria Decision Analysis* 3 (2), 83-104.

Ulungu, E.L. and Teghem, J., 1995. "The two phase method: an efficient procedure to solve bi-objective combinatorial optimization problems". *Foundations of Computing and Decision Sciences* 20 (2), 149-165.