HAL
open science

The Simple and Multiple Job Assignment Problems

Fabrice Chauvet, Jean-Marie Proth, A. Soumare

To cite this version:

Fabrice Chauvet, Jean-Marie Proth, A. Soumare. The Simple and Multiple Job Assignment Problems. [Research Report] RR-3744, INRIA. 1999, pp.15. inria-00072918

HAL Id: inria-00072918

https://inria.hal.science/inria-00072918v1

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

The Simple and Multiple Job Assignment Problems

F. Chauvet - J.-M. Proth - A. Soumare

N° 3744
Août 1999

THÈME 4

Rapport de recherche

Les rapports de recherche de l'INRIA
sont disponibles en format postscript sous
ftp.inria.fr (192.93.2.54)

si vous n'avez pas d'accès ftp
la forme papier peut être commandée par mail :
e-mail : dif.gesdif@inria.fr
(n'oubliez pas de mentionner votre adresse postale).

par courrier :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

INRIA research reports
are available in postscript format
ftp.inria.fr (192.93.2.54)

if you haven't access by ftp
we recommend ordering them by e-mail :
e-mail : dif.gesdif@inria.fr
(don't forget to mention your postal address).

by mail :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

LES PROBLEMES D'AFFECTATION A TACHE UNIQUE ET A TACHES MULTIPLES

F. Chauvet[1], J.-M. Proth[2] et A. Soumare[3]

Résumé

Ce papier s'intéresse à deux problèmes d'affectation que nous avons eu à résoudre pour une entreprise. Dans les deux cas, le nombre d'employés auxquels des tâches devaient être affectées était bien supérieur au nombre de tâches. Le problème d'affectation à tâche unique se caractérise par le fait qu'au plus une tâche est affectée à chaque employé. Cette contrainte disparaît dans le problème d'affectation à tâches multiples. Dans les deux cas, l'objectif est de minimiser l'instant où la dernière tâche se terminera. Ces problèmes sont connus sous le nom de problèmes d'affectation goulot.

Nous montrons que le problème d'affectation à tâche unique peut être résolu de manière optimale par une approche itérative basée sur la dichotomie. A chaque itération, un problème de programmation linéaire est résolu ; dans notre cas, la solution de ce problème est entière.

Nous proposons une heuristique simple et efficace, ainsi qu'une approche par séparation et évaluation, pour résoudre le problème d'affectation à tâches multiples.

Nous présentons des exemples numériques. Ces exemples montrent que l'heuristique est satisfaisante pour résoudre le problème d'affectation à tâches multiples.

[1] INRIA-Lorraine, 4 rue Marconi, 57070 METZ FRANCE

[2] INRIA-Lorraine, 4 rue Marconi, 57070 METZ FRANCE Phone: 00 333 87203501

e-mail: Jean-Marie.Proth@loria.fr

or 22, rue du Colombier, 57420 POUILLY FRANCE Phone: 00 333 87 52 62 02 Fax: 00 333 87 52 67 47

e-mail: jean-marie.proth@libertysurf.fr Auteur auquel doit être adressée la correspondance.

[3] Faculté des Sciences, Université de Nouakchott, MAURITANIE

THE SIMPLE AND MULTIPLE JOB ASSIGNMENT PROBLEMS

F. Chauvet[1], J.-M. Proth[2] and A. Soumare[3]

Abstract

This paper addresses two real-life assignment problems. In both cases, the number of employees to whom tasks should be assigned is significantly greater than the number of tasks. In the simple job assignment problem, at most one task (job) should be assigned to each employee; this constraint is relaxed in the multiple job assignment problem. In both cases, the goal is to minimize the time the last task is completed: these problems are known as Bottleneck Assignment Problems (BAPs for short).

We show that the simple job assignment problem can be solved optimally using an iterative approach based on dichotomy. At each iteration, a linear programming problem is solved: in this case the solution is integer.

We propose a fast heuristic to solve the multiple job assignment problem, as well as a Branch-and-Bound approach which leads to an optimal solution.

Numerical examples are presented. They show that the heuristic is satisfactory for the application at hand.

Key words: Assignment, Bottleneck assignment, Dichotomy, Branch-and-Bound

[1] INRIA-Lorraine, 4 rue Marconi, 57070 METZ FRANCE

[2] INRIA-Lorraine, 4 rue Marconi, 57070 METZ FRANCE Phone: 00 333 87203501

e-mail: Jean-Marie.Proth@loria.fr

or 22, rue du Colombier, 57420 POUILLY FRANCE Phone: 00 333 87 52 62 02 Fax: 00 333 87 52 67 47

e-mail: jean-marie.proth@libertysurf.fr Corresponding author

[3] Faculté des Sciences, Université de Nouakchott, MAURITANIE

1. INTRODUCTION

We consider a set of employees that are supposed to perform a set of tasks. We assume that each employee has a given capacity (for instance, the working period) and that the part of the capacity consumed by each employee performing each of the tasks is known. We also assume that the costs incurred by an employee performing a task is known; this cost may be different from one pair (employee, task) to another.

The goal of the well-known Generalized Assignment Problem (GAP for short) is to assign tasks to employees such that the workload of an employee does not exceed his capacity, while minimizing the total cost.

Kuhn [5] presented the Hungarian method which applies in the particular case when the number of employees is equal to the number of tasks, and each employee performs exactly one task.

More recently, numerous authors have developed algorithms to provide at least heuristic algorithms to solve the GAP that is an NP-hard problem. Amongst these authors are DeMaio and Roveda [1], Neebe and Rao [8], and Srinivasan and Thompson [9], to quote only a few. Also, many applications can be found in the literature as, for instance, [3], [8] and [9].

It appears that, in practice, a large number of situations require to minimize the maximal cost related to an employee instead of minimizing the sum of the costs. This problem is referred to as the Minimax Generalized Assignment Problem (MGAP for short) or, more usually, the Bottleneck Generalized Assignment Problem (BGAP). The problem presented in this paper is of the BGAP type.

The BGAP problems have been introduced by Francis and White [2] in 1974, and further developed by J. B. Mazzola and A. W. Neebe in [6] and [7]. These authors set the BGAP problem type as follows:

Minimize the maximum of the costs related to an employee subject to the following constraints:

Each task is performed by one employee.

The workload of an employee should not exceed his capacity.

Note that each employee is allowed to perform more than one task.

Mazzola and Neebe proposed an algorithm that uses a Branch-and-Bound approach [7]. At each step of the algorithm, a linear programming problem is solved to obtain a lower bound of the solution. The upper bound is obtained by applying any of the heuristic algorithms available on the market as, for instance, those presented in [2] or [3].

The problem presented in this paper has been studied for the National Company for Industry and Mining of Mauritania. Every morning, a lot of workers are at the disposal of the company to perform the tasks of the day. These workers are known from the company, and their efficiency with regard to the tasks that must be performed during the day is recorded in a computer. The goal is to assign the tasks to workers in order to minimize the instant the last task is completed. The capacity of the workers is not taken into account since the number of workers is huge compare to the number of tasks to be performed. Furthermore, the cost of an assignment is simply the time required by the worker to complete the task. These particular

aspects make the approaches proposed hereafter quite different from the one of Mazzola and Neebe.

In section 2, we consider the case when a worker is not allowed to perform more than one task. This situation arises when specific resources, available in one unit only, are required to perform the task. It is showed that, in this case, a linear programming problem always lead to an optimal integer solution.

Section 3 is devoted to the multiple job case, that is the case when each worker is allowed to perform more than one task. An efficient heuristic is proposed, as well as a Branch-and-Bound approach. Numerical examples show that the results provided by the heuristics are closed to the optimal one, and thus can be used on-line in the company.

2. MINIMIZING THE MAXIMUM WORKING TIME: THE SINGLE JOB CASE

2.1. Setting the problem

The National Company for Industry and Mining of Mauritania proposed the problem presented in this section. In this company, n employees are available to perform m maintenance tasks, and n is much greater than m. The employees are usually not specialized to perform particular tasks, and most of the tasks do not require a particular knowledge. Nevertheless, due to their background and their past positions in the company, employees are more or less efficient to perform a given task. In other words, the times that two employees require performing the same task might be different from each other. We denote by $t_{i,j}$ the time required by employee i to perform task j, and $t_{i,j}>0$.

The problem consists in assigning employees to at most one task each while minimizing the time to complete all the tasks.

We define E=[1, 2, ..., n]x[1, 2, ..., m] and we consider the (0-1) variables $x_{i,j}$, $(i, j)\in E$ defined as follows:

$$x_{i,j} = \begin{cases} 1 \text{ if employee i is assigned to task j} \\ \qquad 0 \text{ otherwise} \end{cases}$$

The problem to solve, denoted by Q_0, can be written as follows:

$$\text{Minimize } \max_{(i,j)\in E} x_{i,j} t_{i,j} \tag{1}$$

s.t.

$$\sum_{j=1}^{m} x_{i,j} \leq 1 \text{ for } i = 1, 2, ..., n \tag{2}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \text{ for } j = 1, 2, ..., m \tag{3}$$

$$x_{i,j} \in \{0,1\} \text{ for } (i, j)\in E \tag{4}$$

Criterion (1) means that the completion time of the task, which is completed last, should be as less as possible. In other words, all the tasks are completed as soon as possible. Constraints (2) are introduced to make sure that each employee is assigned to at most one task. According to constraint (3), exactly one employee is assigned to each task. Finally, constraints (4) guaranties that the $x_{i,j}$ variables are (0-1) variables.

Note that, taking into account constraints (3) and (4), the criterion (1) is equivalent to:

$$\text{Minimize} \max_{j\in\{1,\dots,m\}} \sum_{i=1}^{n} x_{i,j}t_{i,j} \qquad (5)$$

This problem can be rewritten at least in the two different manners presented hereafter. The first formulation is a mixed linear programming (MLP for short) problem. The second approach leads to a sequence of linear programming (LP for short) problems combined with a dichotomy process.

2.2. The MLP formulation

Problem (1) to (4) can be rewritten as:

Minimize z (6)

s.t.

$x_{i,j}t_{i,j} \le z$ for $(i, j) \in E$, (7)

and constraints(2), (3) and (4).

z is a real variable while the $x_{i,j}$ are (0-1) variables. Such a problem is solved using a branch-and-bound process, which is usually quite heavy. We prefer the dichotomy approach presented in the next subsection.

2.3. The dichotomy approach

2.3.1. Basic results

We define $t_{min} = \min_{(i,j)\in E} t_{i,j}$ and $t_{max} = \max_{(i,j)\in E} t_{i,j}$. We consider $\theta \in [t_{min}, t_{max}]$ and we set:

$$u^{\theta}_{i,j} = \begin{Bmatrix} t_{i,j} \text{ if } t_{i,j} > \theta \\ 0 \text{ otherwise} \end{Bmatrix} \text{ for } (i, j) \in E \qquad (8)$$

We then consider the assignment problem Q_θ defined as follows:

$$\text{Minimize} \sum_{(i,j)\in E} x_{i,j}u^{\theta}_{i,j} \qquad (9)$$

s.t.

constraints (2), (3) and (4) apply.

Result 1 holds.

Result 1

The optimal value of the criterion of problem Q_θ is zero if and only if the optimal value of the criterion of problem Q_0 is less than or equal to θ.

Proof:

a. The optimal solution $S^*_\theta = \{x^*_{i,j}\}_{(i,j)\in E}$ of problem Q_θ belongs to the domain D defined by constraints (2), (3) and (4). Thus S^*_θ is a feasible solution of Q_0. Furthermore, the optimal criterion of Q_θ being equal to zero, it turns out that $u^{\theta}_{i,j}=0$ for any $x^*_{i,j} = 1$. Thus, for any $(i, j)\in E$:

2. if $x^*_{i,j} = 1$, then $t_{i,j} \le \theta$ and $x^*_{i,j}t_{i,j} \le \theta$,

3. if $x^*_{i,j} = 0$, then $x^*_{i,j} t_{i,j} = 0$.

Finally, $\underset{(i,j)\in E}{\text{Max}}\, x^*_{i,j} t_{i,j} \le \theta$ and S^*_θ is feasible for Q_0.

As a consequence, $\underset{(i,j)\in E}{\text{Max}}\, x^{**}_{i,j} t_{i,j} \le \theta$ for the optimal solution $S^{**}_0 = \left\{x^{**}_{i,j}\right\}_{(i,j)\in E}$ of Q_0.

b. Conversely, if the optimal value of problem Q_0 is less than or equal to θ, then $t_{i,j} \le \theta$ for any $x^{**}_{i,j}=1$, where $\left\{x^{**}_{i,j}\right\}_{(i,j)\in E}$ is the optimal solution of Q_0. Thus the criterion of problem Q_θ is equal to zero for the solution $\left\{x^{**}_{i,j}\right\}_{(i,j)\in E}$. Since $\left\{x^{**}_{i,j}\right\}_{(i,j)\in E}$ is a feasible solution to problem Q_θ, then the optimal value of the criterion of problem Q_θ is zero.

Q.E.D.

Let $t_{i(1),j(1)} \le t_{i(2),j(2)} \le \ldots \le t_{i(mn),j(mn)}$ be the $t_{i,j}$ values ordered in the increasing order, and let r be the smallest integer such that, for any $j \in \{1, 2, \ldots, m\}$, there exists $i \in \{1, 2, \ldots, n\}$ and $s \le r$ which satisfy $t_{i,j} = t_{i(s),j(s)}$. In other words, r is the smallest integer which guaranties that at least one element of the sequence $t_{i(1),j(1)}, t_{i(2),j(2)}, \ldots, t_{i(r),j(r)}$ concerns each one of the tasks 1, 2, ..., m. As a consequence, if $\theta < t_{i(r),j(r)}$ the optimal value of the criterion of problem Q_θ cannot be equal to zero.

The following corollary of result 1 is straightforward.

Corollary

If it is possible to find m pairs $(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)$ in the set $\{(i(1), j(1)), (i(2), j(2)), \ldots, (i(r), j(r))\}$ such that $i_u \ne i_v$ and $j_u \ne j_v$ for $u,v \in \{1, 2, \ldots, m\}$, $u \ne v$, then:

1. the solution which consists in assigning i_k to j_k, k=1, 2, ..., m, is optimal to problem Q_0,
2. $t_{i(r),j(r)}$ is the corresponding optimal value of the criterion.

Proof:

Obvious if we observe that:

the proposed solution is feasible to problem Q_0 and the corresponding value of the criterion is $t_{i(r),j(r)}$,

the optimal solution cannot have a criterion whose value is less than $t_{i(r),j(r)}$.

Q.E.D.

Another result of importance is given hereafter.

Result 2

Let Q^1_θ be the problem derived from Q_θ by replacing constraint (4) by $x_{i,j} \ge 0$ for $(i, j) \in E$. Then the optimal solution of Q^1_θ is optimal for Q_θ.

Proof:

The matrix of constraints (2), (3) and $x_{i,j} \ge 0$ for $(i, j) \in E$ is totally unimodular. Thus the solution of Q^1_θ is integer. According to constraints (3) and (4), the only possible values of the optimal variables are 0 or 1.

Q.E.D.

2.3.2. The dichotomy algorithm

The dichotomy algorithm starts by setting $\theta_1=t_{i(r),j(r)}$, $\theta_2=\theta_{max}$ and w=mn. r is the parameter defined in the previous section.

If the optimal value of the criterion of problem Q_{θ_1} is equal to zero, then θ_1 is the optimal value of the criterion of problem Q_0, and the optimal solution to Q_{θ_1} is optimal to Q_0.

Otherwise, we compute $v=\lceil (r+v)/2 \rceil$, and we set $\theta_3=t_{i(v),j(v)}$. Remember that $\lceil a \rceil$ is the lowest integer greater than or equal to a.

If the optimal value of the criterion of problem Q_{θ_3} is equal to zero we set w=v, otherwise we set r=v. In both cases, we restart the computation with v. The algorithm stops when v=r+1, and the optimal solution to problem Q_0 is the optimal solution to problem Q_{θ_2} where $\theta_2=t_{i(w),j(w)}$.

The formal algorithm is presented hereafter.

Algorithm 1

1- Order the times $t_{i,j}$, $(i, j)\in E$, in their increasing order. Let $t_{i(1),j(1)}\leq t_{i(2),j(2)}\leq\ldots\leq t_{i(mn),j(mn)}$ be the sequence of ordered times.

2- Compute the lowest integer r such that, for any $j\in\{1, 2, \ldots, m\}$, there exists $i\in\{1, 2, \ldots, n\}$ and $s\leq r$ which satisfy $t_{i,j}=t_{i(s),j(s)}$.

3- Set $\theta_1=t_{i(r),j(r)}$ and $\theta_2=t_{i(w),j(w)}$, with w=mn.

4- Compute the optimal solution of Q_{θ_1} and the optimal value $C^*_{\theta_1}$ (see result 2).

5- Compute the optimal solution of Q_{θ_2} and the optimal value $C^*_{\theta_2}$ (see result 2).

6- If $C^*_{\theta_1}=0$, then θ_1 is the optimal value of the criterion of Q_0, and the optimal solution of Q_{θ_1} is the optimal solution of Q_0. End of the algorithm.

7- Compute $v=\lceil (r+v)/2 \rceil$.

8- Set $\theta_3=t_{i(v),j(v)}$.

9- Compute the optimal solution of Q_{θ_3} (see result 1).

10- If $C^*_{\theta_3}=0$, then set w=v, otherwise set r=v.

11- If w=r+1, then the optimal solution to Q_0 is the optimal solution to Q_{θ_2} where $\theta_2=t_{i(w),j(w)}$. End of the algorithm.

12- Go to 7.

This algorithm is applied to the example presented in the next subsection.

2.3.3 Application

The example presented in these subsection concerns five tasks and eight employees. The times needed by the employees to complete the tasks are given in table 1.

As we can see, a possible increasing order could be the one presented in table 2, where the integer located at the intersection of the i-th row and the j-th column is the rank of $t_{i,j}$.

$t_{6,4}$ is the first value such that the part of the ordered sequence which ends at $t_{6,4}$ contains at least one element belonging to each column. Thus, r=25 since $t_{6,4}$ is the 25-th element in the sequence, and $\theta_1=5$ and $\theta_2=12$.

Table 1: Completion times (Example 1).

TASKS⇒ EMPLOYEES ⇓	1	2	3	4	5
1	1	4	5	8	9
2	3	3	6	6	3
3	2	6	7	7	4
4	2	5	12	9	4
5	3	4	5	8	8
6	2	7	5	5	4
7	1	3	4	6	5
8	2	3	5	8	4

Table 2: Ranks

TASKS⇒ EMPLOYEES ⇓	1	2	3	4	5
1	1	13	21	34	39
2	7	9	28	29	12
3	3	27	32	33	16
4	4	20	40	38	17
5	8	14	22	35	37
6	5	31	23	**25**	18
7	2	10	15	30	26
8	6	11	24	36	19

We then solve problem Q_5^1. An optimal solution to this problem is summarized in Table 3 where the value 1 in a box means that the corresponding task is assigned to the corresponding employee.

This solution is not unique. Since the optimal value of the criterion of problem Q_5^1 is equal to zero, this solution is also the optimal solution to problem Q_0, and the optimal value of the criterion of Q_0 is equal to 5. In this example, algorithm 1 stopped at step 3.

Table 3: Optimal solution to Q_0.

TASKS⇒ EMPLOYEES ⇓	1	2	3	4	5
1	1				
2					
3					
4					
5			1		
6				1	
7		1			
8					1

3. MINIMIZING THE MAXIMUM WORKING TIME: THE MULTIPLE JOB CASE

3.1. Introductory remarks

It may happen that some of the employees are much efficient than others in performing the tasks. In this case, allowing an employee to perform several tasks in series may lead to a completion time of the tasks which is better than the one obtained applying Algorithm 1. It is the case for example 2 presented in Table 4.

Table 4: Completion times (Example 2).

TASKS⇒ EMPLOYEES ⇓	1	2	3	4	5
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5
6	6	6	6	6	6
7	7	7	7	7	7
8	8	8	8	8	8

If each employee is allowed to perform at most one task (problem Q_0), an optimal solution to Q_0 consists in assigning employee i to task i, for i=1 to 5, and the optimal value of the criterion is 5.

If each employee is allowed to perform more than one task in series (problem R_0), an optimal solution to problem R_0 consists of assigning:

employee 1 to tasks 1, 2 and 3,
employee 2 to task 4,
employee 3 to task 5.

The optimal value of the criterion is 3.

3.2. Setting the problem

The problem R_0 to be solved is derived from problem Q_0 by removing constraint (2) which restricts to 1 the maximal number of tasks that can be performed by an employee. Thus, problem R_0 can be written as follows:

Minimize $\max_{(i,j)\in E} x_{i,j}t_{i,j}$ (10)

s.t.

$\sum_{i=1}^{n} x_{i,j} = 1$ for $j = 1, 2, ..., m$ (11)

$x_{i,j} \in \{0,1\}$ for $(i, j)\in E$ (12)

Problem R_0 can also be rewritten as:

Minimize z (13)

s.t.

$x_{i,j}t_{i,j} \leq z$ for $(i, j)\in E$, (14)

(11) and (12) apply.

This problem is a Mixed Linear Programming (MLP) problem whose optimal solution can be obtained only by applying a Branch and Bound approach. We will propose such an approach in subsection 3.4. The next subsection is devoted to the presentation of a heuristic algorithm which lead to a near-optimal solution to problem R_0.

3.3. A near-optimal solution to the multi job case problem

This algorithm is an iterative approach, which consists of assigning, at each iteration, an unassigned task to an employee such that the maximal working time of the employees is minimal. This algorithm, referred to as Algorithm 2, is given hereafter.

Algorithm 2

1. Initialization
   - 1.1 For i=1 to n, set y_i=0. *Variables y_i represent the cumulative working times of the employees.*
   - 1.2. Set A={1, 2, ..., m}. *A is the set of tasks that have not been assigned to an employee yet.*
   - 1.3. Set B=∅. *B is the set of tasks that have already been assigned to an employee.*
2. If A=∅, stop: all the tasks have been assigned.
3. For i=1 to n, let $s(i)\in A$ such that $y_i+t_{i,s(i)}=\min_{j\in A}(y_i + t_{i,j})$.

4. Let $i^* \in \{1, 2, \ldots, n\}$ such that: $y_{i^*}+t_{i^*, s(i^*)} = \min_{i \in \{1, \ldots, n\}} (y_i + t_{i, s(i)})$.
5. Assign task $s(i^*)$ to employee i^*.
6. Set $A=A \backslash \{s(i^*)\}$.
7. Set $B=B \cup \{s(i^*)\}$.
8. Set $y_{i^*}=y_{i^*}+t_{i^*, s(i^*)}$.
9. Go to 2.

To illustrate this approach, we apply Algorithm 2 to example 1 (see Table 1). The results are given in table 6.

Table 6: Applying Algorithm 2 to Example 1.

It.	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	Assignments	Adjustments
0	0	0	0	0	0	0	0	0		A={1, ..., 5} B=∅
1	1	0	0	0	0	0	0	0	Task 1⇒ Emp. 1	A={2, ..., 5} B={1}
2	1	3	0	0	0	0	0	0	Task 2⇒ Emp. 2	A={3, 4, 5} B={1, 2}
3	1	3	0	0	0	0	4	0	Task 3⇒ Emp. 7	A={4, 5} B={1, 2, 3}
4	1	3	4	0	0	0	4	0	Task 5⇒ Emp. 3	A={4} B={1, 2, 3, 5}
5	1	3	4	0	0	5	4	0	Task 4⇒ Emp. 6	A=∅ B={1, ..., 5}

It appears that at most one task has been assigned to each employee, and that the optimal value of the criterion is 5.

3.4. The Branch and Bound approach

3.4.1 Presentation of the algorithm

Algorithms 1 and 2 provide an upper bound for the optimal value of the criterion of problem R_0 defined by (13), (14), (11), (12). The optimal value that algorithm 1 provides is an upper bound since problem Q_0 is more constrained than R_0. Algorithm 2 also provides an upper bound since it is a heuristic for R_0. Both can be used in the Branch-and-Bound approach presented hereafter.

Assume that both algorithms have been applied and have lead to a1 and a2, values of the criterion, respectively. Then $a^* = \min(a1, a2)$ is the upper bound of the optimal value of the criterion used in the algorithm.

Assume that tasks 1, 2, ..., j-1 have already been assigned to employees $i_1, i_2, \ldots, i_{j-1}$ respectively. We may have $i_k = i_l$ for k, l∈ {1, 2, ...j-1} and k≠l; in other words, it is possible that more than one task has been assigned to the same employee. The working load w_i of each employee i∈ {1, 2, ..., n} at this point of the assignment can be computed as follows:

$$\begin{cases} \text{Set } w_i = 0 \text{ for } i = 1, 2, \ldots, n \\ \text{For } k = 1, 2, \ldots, j-1, \text{ set } w_{i_k} = w_{i_k} + t_{i_k,k} x_{i_k,k} \end{cases}$$

$$\text{where } x_{i_k,k} = \begin{cases} 1 \text{ if task k has been assigned to employee } i_k \\ 0 \text{ otherwise} \end{cases}$$

The next task to assign is j. This task can be assigned to anyone of the n employees. Assume that we want to assign j to employee i_1∈ {1, 2, ..., n}. To find a lower bound of the criterion, we first solve the problem R_1 defined as follows:

Minimize z

s.t.

$v_i \le z$ for i∈ {1, 2, ..., n}

$v_i - \sum_{k=j+1}^{m} x_{i,k} t_{i,k} = w_i$ for i∈ {1, 2, ..., n} and $i \ne i_1$

$v_{i_1} - t_{i_1,j} - \sum_{k=j+1}^{m} x_{i_1,k} t_{i_1,k} = w_{i_1}$

$\sum_{i=1}^{n} x_{i,k} = 1$ for $k = j+1, \ldots, m$

$x_{i,j} \ge 0$ for (i, j)∈ E

v_i is the work load of employee i after assigning task j to employee i_1 and tasks j+1 to m to the others employees under the hypothesis that one task can be shared amongst several employees.

This problem is a linear programming problem, thus easy to solve.

z is a lower bound of the criterion, taking into account the assignments already done. Thus, if z>a*, the partial assignment of tasks 1, 2, ..., j-1 to $i_1, i_2, \ldots, i_{j-1}$ respectively cannot lead to an optimal solution of R_0 since any solution obtained by completing the partial assignment would lead to a value of the criterion greater than the upper bound of the optimal value. The formal algorithm presented hereafter is derived from the previous remarks.

Algorithm 3

1. Compute the optimal solution of problem Q_0 obtained by adding constraint (2) to problem R_0. The solution to this problem is obtained using Algorithm 1. Let a1 be the optimal value of the criterion.
2. Compute the solution of R_0 using Algorithm 2. Let a2 be the optimal value of the criterion.
3. Compute a*=max(a1, a2): a* is an upper bound of the optimal solution of problem R_0.
4. Set ND=1. *ND will contain the number of leaves of the Branch-and-Bound tree. At the beginning of the computation, ND=1 since only the root of the tree exists.*

5. Set $l_0^1 = \varnothing$. *l_j^p will contain the list of the employees to whom the j-th first tasks have been assigned. l_j^p correspond to the p-th leaf of the Branch-and-Bound tree. In this list, the employees are given in the order of the assignment of the tasks.*

6. For j=1 to m do

6.1 Set NS=0: *NS is the counter of the number of leaves at the next level of the Branch-and-Bound tree.*

6.2. For k=1 to ND do

6.2.1For i=1 to n do

6.2.1.1. Solve problem R_1

6.2.1.2. If $z \leq a^*$ then

6.2.1.2.1 Set NS=NS+1

6.2.1.2.2 Set $l_j^{NS} = l_{j-1}^k \bigcup \{i\}$

6.2.2 Next i

6.3 Next k

6.4. ND=NS

7. Next j

8. Selecting the optimal solution.

Each l_m^k, k=1, 2, ..., ND, leads to a solution for which the value of the riterion is less than a. The optimal solution is obtained as follows.*

8.1.Set z*=0

8.2.Set k*=1

8.3.Set j=0

8.4.For all $i \in l_m^1$ considered in the order of the list do:

8.4.1.Set j=j+1

8.4.2.Set $z^*=z^*+t_{i,j}$

8.5.If ND>1 then

8.5.1.For k=2 to ND do

8.5.1.1.Set z=0

8.5.1.2.Set j=0

8.5.1.3.For all $i \in l_m^k$ considered in the order of the list, do:

8.5.1.3.1. Set j=j+1

8.5.1.3.2. Set $z=z+t_{i,j}$

8.5.1.4.If z<z* then:

8.5.1.4.1. Set z*=z

8.5.1.4.2. Set k*=k

*8.6.*Optimal set of assignments: $l_m^{k^*}$. *The j-th element of the list is the employee to which task j is assigned*

Optimal value of the criterion: z*

3.4.2. Application

In Table 7, we present the results obtained by running 21 examples generated at random. In each example, the number n of employees and the number m of tasks are inputs of the program.

Other inputs are lb and ub, respectively the lower bound and the upper bound of the times required by the first employee to perform the tasks, as well as the so-called "deviation parameter" denoted by d. The times required by employee $i \in \{1, 2, \ldots, n\}$ to perform the tasks are generated at random in $\{lb+(i-1)d, ub+(i\ 1)d\}$.

As mentioned in Algorithm 3, a1 and a2 are the values of the criterion obtained by solving Q_0 using Algorithm 1 and R_0 using algorithm 2, respectively. The upper bound used in the branch-and-bound algorithm is $a^*=\min\{a1, a2\}$; this value is underlined in Table 7.

The optimal value of the criterion obtained by applying the branch-and-bound algorithm (Algorithm 3) is reported in column B&B; this value is underlined if it is different from a*.

Table 7: Comparative results

Example Number	n	m	lb	ub	d	a1	a2	B&B
1	10	6	5	15	4	25	18	18
2	10	8	5	15	0	9	10	9
3	14	8	5	20	1	16	19	16
4	14	5	5	25	4	23	22	20
5	15	7	5	20	5	35	25	24
6	14	6	5	20	0	9	9	9
7	14	6	5	20	2	15	15	15
8	14	6	5	20	4	25	19	18
9	14	6	5	20	6	35	25	23
10	12	8	5	20	4	37	29	28
11	14	6	5	20	7	40	28	26
12	15	6	2	15	0	4	4	4
13	15	6	2	15	2	12	9	9
14	15	6	2	15	4	22	15	14
15	15	6	2	15	6	32	17	17
16	15	6	2	15	8	42	25	23
17	12	9	5	20	0	7	10	7
18	12	9	5	20	4	37	28	27
19	12	9	5	20	6	53	36	35
20	15	5	5	20	0	6	6	6
21	15	5	5	20	5	27	23	22

We can draw two main conclusions from Table 7 and the numerous examples that have been solved:

(i) If the performances of the employees are very different from each other, that is if d is large, then Algorithm 2 provides a better result than Algorithm 1. This is easy to understand since Algorithm 1 leads to a solution in which each employee performs at most one task, and Algorithm 2 allows an efficient employee to perform more than one task.

(ii) The Branch-and-Bound algorithm (Algorithm 3) is better than the best of the two Algorithms 1 and 2 in about 50% of the examples, but the improvement obtained by applying Algorithm 3 is of less than 5% in about 80% of the cases.

4. CONCLUSION

In this paper, we consider the case when the number of employees who are supposed to perform a set of tasks is greater than the number of tasks. The goal is to minimize the time required to perform all the tasks, that is the makespan.

Two cases are considered:

(i) Each employee is allowed to perform at most one task (single job case). We propose an algorithm (Algorithm 1) which is based on a dichotomy approach. At each iteration, a linear programming problem is solved. This algorithm leads to an optimal solution.

(ii) Each employee is allowed to perform more than one operation (multiple job case). To solve this problem, we propose an heuristic (Algorithm 2) and a branch-and-bound approach (Algorithm 3) which takes advantage of the two previous algorithms to obtain an upper bound of the optimal solution.

The main conclusions to draw from this study is that the single job approach concerns employees whose efficiency with regard to all the tasks is similar in probability, while the multiple job approach concerns situations in which some of the employees are significantly more efficient than others. In this case, it was showed that the heuristic approach provides results that are close to the optimal ones.

The next step of the research is twofold: (i) improve the bounds in the branch-and-bound approach and, (ii) develop heuristics for the multiple job case.

References

[1] A. O. DeMaio and C. A. Roveda, An all zero-one algorithm for a certain class of transportation problems, *Op. Res., vol. 19, pp. 1406-1418, 1971*

[2] R. L. Francis and J. A. White, *Facility Layout and Location,* Prentice-Hall, Englewood Cliffs, New Jersey, 1974

[3] R. S. Garfinkel, A. W. Neebe and M. R. Rao, The m-center problem : minimax facility location, *Management Sci., vol. 23, pp. 1133-1142, 1977*

[4] R. S. Garfinkel and M. R. Rao, The Bottleneck Transportation Problem, *Naval Res. Logistics Quarterly, Vol. 18, PP. 465-472, 1971*

[5] Kuhn, H. W.,The Hungarian Method for the Assignment Problem, *Naval Research Logistics Quarterly* 2, pp.83-97, 1955.

[6] J. B. Mazzola and A. W. Neebe, Bottleneck Generalized Assignment Problem, *Eng. Costs Prod. Economics, vol. 14, pp. 61-65, 1988*

[7] J. B. Mazzola and A. W. Neebe, An Algorithm for the Bottleneck Generalized Assignment Problem, *Computers Ops. Res., vol. 20, n°4, pp. 355-362, 1993*

[8] A. W. Neebe and M. R. Rao, An algorithm for the fixed-charge assigning users to sources problems, *Journal Operational Research Soc., vol. 34, pp. 1107-1113, 1983*

[9] V. Srinivasan and G. L. Thompson, An algorithm for assigning users to sources in a special case of transportation problem, *Ops. Res., vol. 21, pp. 284-295, 1973*

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399

RR-3744