RAIRO Operations Research
Will be set by the publisher

COMPARISON OF DIFFERENT LOWER BOUNDS FOR THE CONSTRAINED MODULE ALLOCATION PROBLEM

Sourour Elloumi[1], Frédéric Roupin[2] and Eric Soutif[3]

Abstract. We consider the Constrained Module Allocation Problem (CMAP), where a set of program modules must be assigned to a set of processors having a limited capacity. The optimal assignment minimizes the sum of execution costs and communication costs between modules. This problem is naturally formulated as a quadratic 0-1 problem with linear constraints. In this paper, we propose seven lower bounds for the CMAP, coming from three families of techniques: linearization, semidefinite programming and lagrangian decomposition. We explain in details how to use these techniques. We make several comparisons from a theoretical point of view. Furthermore, we carry out an extensive experimental comparison, based on many generated instances of different types. Our objective is to give an empirical qualitative measure of the advantages and drawbacks of each lower bound.

Keywords: Quadratic 0-1 Programming, Linearization Techniques, Semidefinite Positive Relaxation, Lagrangian Decomposition, Experiments

.

[1] CEDRIC-CNAM, 292 Rue Saint-Martin I418, F-75141 Paris cedex 03, France elloumi@cnam.fr
[2] CEDRIC-IIE, 18, Allée Jean Rostand, 91025 Evry cedex, France roupin@iie.cnam.fr
[3] CEDRIC-CNAM, 292 Rue Saint-Martin I418, F-75141 Paris cedex 03, France soutif@cnam.fr

Introduction

Due to the advances in VLSI technology, there has been a proliferation of distributed computing systems in the past few years. The *Module Allocation Problem* (MAP) in these systems consists of finding a suitable assignment of program modules (or tasks) to processors so that the sum of execution and communication costs is minimized. Several variants of this problem have already been considered, with different assumptions on the architecture of the distributed system or on the structure of costs and feasible solutions. See [2], [3], [4], [5], [6], [11], [28], [14], [23].

In this paper, we consider the *Constrained Module Allocation Problem* (CMAP). We suppose each processor has a memory limit. That is, the sum of memory requirements of the modules assigned to a given processor must not exceed its memory limit. Moreover, communication links are assumed to be identical: communication costs are independent from the processors the modules are assigned to, they are said *uniform*.

This model was considered in [4], and formulated as a quadratic program with 0-1 variables. It has then be considered in [28] where it is shown that, unless P=NP, no polynomial-time algorithm can guarantee to find a feasible solution within c percent of the optimal value, where c is any fixed positive constant. In [30] and [23], generalizations of the CMAP are considered (non-uniform communication costs, memory constraint for one processor only, memory and processing constraints), and heuristic methods are proposed.

In this paper, our objective is to compare several different methods for computing a lower bound of the optimal solution value for the CMAP. This experimental comparison will have two criteria. The first criterion is the quality of the lower bound, i.e., the relative gap between the bound and the best known solution. The second criterion is its computation time. Whenever possible, we will also compare the quality of the lower bounds from a theoretical point of view.

The following of this paper is organized as follows. In Section 1, we state the problem and recall its formulation as a quadratic 0-1 problem (Q01). Then, the different lower bounds we consider will be grouped into three classes. In Section 2, we present three lower bounds obtained by linearization, i.e., *(i)* transformation of program (Q01) into a linear MIP, and *(ii)* computation of its LP-relaxation. These three lower bounds will be called L1, L2, and L3. In Section 3, three bounds obtained by semidefinite programming are described. They are denoted by S0, S1, and S2. In Section 4, we present a lower bound, called D0 and obtained by lagrangian decomposition. In Section 5, we present an extensive computational comparison of all these seven lower bounds. Finally, we make some concluding remarks in Section 6.

1. Problem Statement and Formulation by a Quadratic 0-1 Program

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_P\}$ be a set of P processors, and let $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$ be a set of T program modules to be assigned to the processors. We denote by q_{tp} $(t = 1, \ldots, T,\ p = 1, \ldots, P)$ the execution cost of module t on processor p, and by $c_{tt'}$ $(t,\ t' = 1, \ldots, T,\ t \neq t')$ the communication cost occuring when modules t and t' are assigned to different processors. Communication costs are said *uniform* since they are independent from the processors and they are symmetric, i.e., $c_{tt'} = c_{t't}$. They can be represented by a *communication graph.* In the CMAP, each processor p $(p = 1, \ldots, P)$ has a limited memory n_p, and the memory requirement of module t is denoted by s_t. In any feasible solution, the sum of memory requirements of the modules assigned to a processor p must not exceed n_p. This kind of constraints are called *memory constraints* in [28] and *storage constraints* in [30].

A natural mathematical programming formulation of CMAP considers the variables vector $x = (x_{tp})$ $(t = 1, \ldots, T; p = 1, \ldots, P)$ where x_{tp} is equal to 1 if module t is allocated to processor p and is equal to 0 otherwise.

Let $c_0 = \sum_{t=1}^{T-1} \sum_{t'=t+1}^{T} c_{tt'}$, the CMAP can be formulated by the following quadratic 0-1 problem:

$$\text{(Q01)} : \min F(x) = c_0 + \sum_{t=1}^{T} \sum_{p=1}^{P} q_{tp} x_{tp} - \sum_{t=1}^{T-1} \sum_{t'=t+1}^{T} \sum_{p=1}^{P} c_{tt'} x_{tp} x_{t'p} \tag{1}$$

Subject to:

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1, \ldots, T \tag{2}$$

$$\sum_{t=1}^{T} s_t x_{tp} \leq n_p \qquad p = 1, \ldots, P \tag{3}$$

$$x \in \{0,1\}^{T \times P} \tag{4}$$

Constraints (2) force every module t to be allocated to exactly one processor, and constraints (3) are the memory constraints. To obtain the objective function in the form (1), we first write it as the sum of execution costs and communication costs between pairs of modules allocated to different processors, i.e.,

$$F(x) = \sum_{t=1}^{T} \sum_{p=1}^{P} q_{tp} x_{tp} + \sum_{t=1}^{T-1} \sum_{t'=t+1}^{T} \sum_{p=1}^{P} \sum_{p'=1,\ p' \neq p}^{P} c_{tt'} x_{tp} x_{t'p'}$$

Then we use constraints (2), together with the fact that communication costs are uniform and symmetric, to get (1).

We will further use a third formulation of the objective function (1), that is valid since the communication costs are symmetric:

$$F(x) = c_0 + \sum_{t=1}^{T} \sum_{p=1}^{P} q_{tp} x_{tp} - \sum_{t=1}^{T} \sum_{t'=1,\ t' \neq t}^{T} \sum_{p=1}^{P} \frac{1}{2} c_{tt'} x_{tp} x_{t'p} \tag{5}$$

In the following, we shall use either the form (1) or (5) of the objective function, and we assume that (Q01) is feasible.

2. Three bounds obtained by linearization

In this section, we describe three lower bounds which first linearize (Q01), i.e. find an equivalent formulation by a mixed integer linear program (MILP). Then, by dropping the integrality constraints, one can compute the LP-relaxation in polynomial time. Another advantage of the linearization process is that the MILP can also be handled by a MIP solver in order to compute optimal solutions for medium-sized instances.

2.1. The lower bound L1

Here, we use the well known linearization technique introduced first by Dantzig [13], and then extensively used by many authors. It consists of replacing every product ab of two binary variables by a new real variable c, and adding the constraints $c \leq a$, $c \leq b$, $c \geq a + b - 1$, and $c \geq 0$. This linearization is very general for 0-1 quadratic problems. By considering the sign of the coefficients of the quadratic terms in the objective function, one can easily drop some of those new constraints.

More precisely, we define the lower bound L1 as L1= $v(LP_1)$ where (LP_1) is the following linear program and $v(LP_1)$ its optimal value:

$$(LP_1)\ :\ \min F_1(x,z) = c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}z_{tpt'} \tag{6}$$

Subject to:

(2), (3)

$$z_{tpt'} \le x_{tp} \qquad 1 \le t < t' \le T; p = 1,\dots,P \tag{7}$$
$$z_{tpt'} \le x_{t'p} \qquad 1 \le t < t' \le T; p = 1,\dots,P \tag{8}$$
$$x_{tp} \ge 0 \qquad t = 1,\dots,T; p = 1,\dots,P \tag{9}$$
$$z_{tpt'} \ge 0 \qquad 1 \le t < t' \le T; p = 1,\dots,P \tag{10}$$

Observe that, in any optimal solution of problem (LP_1), variable $z_{tpt'}$ is equal to the minimum value of x_{tp} and $x_{t'p}$, which is also the product $x_{tp}x_{t'p}$ if those variables are in $\{0,1\}$.

Billionnet and Elloumi [4] show that the lower bound L1 is also equal to the optimal value of the lagrangian dual problem obtained by dualizing all the constraints of problem (Q01), i.e., constraints (2) and (3). These authors show that, moreover, this lower bound can be very weak. For example, L1=0 if all the execution costs q_{tp} are equal to 0. However, L1 can be very tight when there are no memory constraints (3). A computational study reported in [3] shows that the integrality gap is either equal to 0 or is very small.

2.2. The lower bound L2

This lower bound is an improvement on L1. The main idea here is to add a set of new quadratic constraints to problem (Q01), and then apply the same linearization trick as above. The new quadratic constraints are obtained by multiplying each constraint by each variable. Using these new constraints has first been suggested by Adams and Sherali [1] and has then been used by many authors, for a variety of quadratic 0-1 problems with linear constraints.

Here, L2 is precisely the optimal value of the following linear problem:

$$(LP_2)\ :\ \min F_2(x,r) = c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}r_{tpt'p} \tag{11}$$

Subject to:

(2), (3)

$$\sum_{p=1}^{P} r_{tpt'p'} = x_{t'p'} \qquad 1 \le t < t' \le T; p' = 1,\dots,P \tag{12}$$
$$\sum_{p=1}^{P} r_{t'p'tp} = x_{t'p'} \qquad 1 \le t' < t \le T; p' = 1,\dots,P \tag{13}$$
$$\sum_{t=1}^{t'-1} s_t r_{tpt'p} + \sum_{t=t'+1}^{T} s_t r_{t'ptp} \le (n_p - s_{t'})x_{t'p} \qquad t' = 1,\dots,T; p = 1,\dots,P \tag{14}$$
$$\sum_{t=1}^{t'-1} s_t r_{tpt'p'} + \sum_{t=t'+1}^{T} s_t r_{t'p'tp} \le n_p x_{t'p'} \qquad t' = 1,\dots,T; p,p' = 1,\dots,P, p' \ne p \tag{15}$$
$$x_{tp} \ge 0 \qquad t = 1,\dots,T; p = 1,\dots,P \tag{16}$$
$$r_{tpt'p'} \ge 0 \qquad 1 \le t < t' \le T; p,p' = 1,\dots,P \tag{17}$$

For every module t, we multiply the corresponding assignment constraint (2) by each variable $x_{t'p'}$ such that $t' \ne t$. This yields constraints (12) for $t < t'$ and constraints (13) for $t' < t$. Similarly, for each processor p, we multiply the corresponding memory constraint (3) by each variable $x_{t'p}$, this yields constraints (14). The same constraints (14), multiplied by $x_{t'p'}$ with $p' \ne p$, lead to constraints (15).

We take into account the fact that $x_{tp}x_{tp} = x_{tp}$ and $x_{tp}x_{tp'} = 0$ for $p' \neq p$. One can easily show that constraints (2), (3) and (12)-(17), all together, imply the linearization constraints, i.e., whenever x_{tp} and $x_{t'p'}$ are 0-1, then $r_{tpt'p'}$ is equal to $x_{tp}x_{t'p'}$. Obviously, L2 $\geq$ L1. Moreover, when there is no memory constraints (3), Billionnet and Elloumi [7] show that computing L2 also leads to an optimal reduction for the initial problem, i.e. the best way of writing the objective function as a constant plus a non-negative quadratic function.

Last, observe that, following the general method proposed in [1], one would build another family of valid inequalities by multiplying each linear constraint by $(1 - x_{tp})$. Because of constraints (2), it is easy to show that this new family of constraints would be redundant in (LP_2).

2.3. The lower bound L3

Here, we use a different linearization technique, initially proposed by Glover [22]. This method has then been used for example in [9] for the quadratic knapsack problem.

The main idea is to replace the expression $x_{tp}\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2}c_{tt'}x_{t'p}\right)$ by a unique variable h_{tp}, for $t = 1, \ldots, T; p = 1, \ldots, P$. Let α be a given $T \times P$ vector and let (L_α) be the following mixed integer linear problem:

$$(L_\alpha)\ :\ \min \delta(x,h) = c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T}\sum_{p=1}^{P} h_{tp} \tag{18}$$

Subject to:
(2), (3)

$$h_{tp} \leq \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2}c_{tt'}x_{t'p} \qquad t = 1, \ldots, T; p = 1, \ldots, P \tag{19}$$

$$h_{tp} \leq \alpha_{tp}x_{tp} \qquad t = 1, \ldots, T; p = 1, \ldots, P \tag{20}$$

$$h_{tp} \geq 0 \qquad t = 1, \ldots, T; p = 1, \ldots, P \tag{21}$$

$$x_{tp} \in \{0,1\} \qquad t = 1, \ldots, T; p = 1, \ldots, P \tag{22}$$

In the following proposition, we prove sufficient conditions for on vector α so that problems (L_α) and (Q01) have the same optimal values.

Proposition 1. *If vector α satisfies:*

(i) for any (t,p) such that there exists no feasible solution to (Q01) with $x_{tp} = 1$, α_{tp} is any arbitrary number, and

(ii) for any (t,p) such that there exists a feasible solution to (Q01) with $x_{tp} = 1$,

$$\alpha_{tp} \geq max\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2}c_{tt'}x_{t'p}\ : x \text{ solution to (Q01) with } x_{tp} = 1\right) \tag{23}$$

then, problems (L_α) and (Q01) have the same optimal values.

Proof: First, observe that, for every feasible solution $\tilde{x}$ to (Q01), it is possible to build a feasible solution $(\tilde{x}, \tilde{h})$ to (L_α), where, for any (t,p),

$$\tilde{h}_{tp} = \min\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2}c_{tt'}\tilde{x}_{t'p},\ \alpha_{tp}\tilde{x}_{tp}\right) \tag{24}$$

and that identity (24) constitutes a necessary optimality condition for problem (L_α).

Now we will show that, if conditions *(i)* and *(ii)* are satisfied, then, for any (t,p), $\tilde{h}_{tp}$ is equal to $\tilde{x}_{tp}\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2}c_{tt'}\tilde{x}_{t'p}\right)$, i.e., $\delta(\tilde{x},\tilde{h}) = F(\tilde{x})$. For this, let us discuss the two cases : $\tilde{x}_{tp} = 0$ and $\tilde{x}_{tp} = 1$.

- if (t,p) is such that $\tilde{x}_{tp} = 0$ (either *(i)* or *(ii)* is satisfied), then identity (24) implies $h_{tp} = 0$.
- if (t,p) is such that $\tilde{x}_{tp} = 1$. In this case, (t,p) satisfies *(ii)*, and it follows from identity (24) and inequality (23) that $h_{tp} = \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} \tilde{x}_{t'p}$.

■

Hence, for any choice of α satisfying Proposition 1, solving the LP-relaxation $(\underline{L}_\alpha)$ of (L_α) gives a lower bound for problem (Q01). The advantage of this linearization method is that it uses a small number of additional variables $O(TP)$, while (LP_1) (resp. (LP_2)) uses up to $O(T^2P)$ (resp. $O(T^2P^2)$) additional variables.

A trivial choice of α which satisfies Proposition 1 would be to set all its components to an arbitrary large number M. Another trivial choice of α will be described in the following proposition. A drawback of this choice is that it gives a lower bound which will be proved in the following proposition to be even worst than L1.

Proposition 2. *Let the vector β be defined by $\beta_{tp} = \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'}$, then the optimal value of $(\underline{L}_\beta)$ is less than or equal to L1.*

Proof: Obviously, β satisfies Proposition 1. Now, let $(\tilde{x}, \tilde{z})$ be an optimal solution to problem (LP_1). Then $\tilde{z}_{tpt'} = \min(\tilde{x}_{tp}, \tilde{x}_{t'p})$, and
$F_1(\tilde{x}, \tilde{z}) = c_0 + \sum_{t=1}^{T} \sum_{p=1}^{P} q_{tp} \tilde{x}_{tp} - \sum_{t=1}^{T} \sum_{p=1}^{P} \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} \min(\tilde{x}_{tp}, \tilde{x}_{t'p})$.

As already observed in the proof of Proposition 1, it is possible to build a feasible solution $(\tilde{x}, \tilde{h})$ to (L_β) with

$$\tilde{h}_{tp} = \min\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} \tilde{x}_{t'p}, \beta_{tp} \tilde{x}_{tp} \right) \tag{25}$$

the objective function is

$$\delta(\tilde{x}, \tilde{h}) = c_0 + \sum_{t=1}^{T} \sum_{p=1}^{P} q_{tp} \tilde{x}_{tp} - \sum_{t=1}^{T} \sum_{p=1}^{P} \min\left(\sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} \tilde{x}_{t'p}, \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} \tilde{x}_{tp} \right) \tag{26}$$

and one can observe that $\delta(\tilde{x}, \tilde{h}) \leq F_1(\tilde{x}, \tilde{z})$. ■

Since the obtained lower bound increases as the values α_{tp} decrease, we will choose coefficients as small as possible, among those which satisfy Proposition 1. Hence, we compute our values α^*_{tp} by solving the following *generalized assignment problems*:

$$(G_{tp}) \ : \ \max \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} x_{t'p} \tag{27}$$

Subject to: :

$$x_{tp} = 1 \tag{28}$$

$$\sum_{p'=1}^{P} x_{t'p'} = 1 \qquad t' = 1, \ldots, T \tag{29}$$

$$\sum_{t'=1}^{T} s_{t'} x_{t'p'} \leq n_{p'} \qquad p' = 1, \ldots, P \tag{30}$$

$$x \in \{0,1\}^{T\times P} \tag{31}$$

If (G_{tp}) is infeasible, we set α^*_{tp} to an arbitrary negative number, for example -1, in such a way that x_{tp} will be fixed to 0 in problem (L_{α^*}). Otherwise, we set α^*_{tp} to the optimal value of problem (G_{tp}). Vector

α^* satisfies the sufficient conditions of Proposition 1 and hence, problem (L_{α^*}) has the same optimal value as problem (Q01).

We denote by L3 the optimal value of the LP-relaxation of problem (L_{α^*}). It is our third lower bound obtained by linearization. Our computational results reported in Section 5 will show that, in general, L3 is much better than L1.

Observe that computing coefficients α^*_{tp} is NP-hard since it amounts to solve generalized assignment problems. However, for the instances that we consider in Section 5, the largest instances have $T = 20$ and $P = 5$. For these instances, one has to compute 100 values α^*_{tp}, each by solving a generalized assignment problem with 100 variables. As proved by our computational results, this preprocessing can be done within a few seconds.

3. Semidefinite Programming Approach

In this section, we discuss three different SDP relaxations for problem $(Q01)$. The first one, (SDP_0) is the basic SDP relaxation. The two other relaxations, (SDP_1) and (SDP_2) can be viewed as successive improvements on (SDP_0), obtained respectively by applying the algorithm proposed in [29] to the linear problems (LP_1) and (LP_2) built above in Section 2. This algorithm provides a set of rules (see Table 1) that allows to build semidefinite relaxations for any quadratic or linear program with bivalent variables, starting from an existing linear relaxation of the considered problem. Moreover, it guarantees to obtain a SDP relaxation which is always better than the linear relaxation used to build it. In fact, (SDP_2) was originally proposed in [29], where some numerical tests are presented for this relaxation. For our numerical evaluation, we have used the Spectral Bundle algorithm of Helmberg ([24]) and SDP_S, a implementation of the Roupin's algorithm [15].

(P_L)	$(SDP\{0,1\})$
$x \in [0,1]^n$	$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$ $\mathbf{d}(X) = x$
$A \bullet X + c^T x = (\leq) d$	Rule **Q** $A \bullet X + c^T x = (\leq) d$
$c^T x = d$	Rule **LE1** $\begin{cases} cc^T \bullet X = d^2 \\ c^T x = d \end{cases}$ $\Leftrightarrow cc^T \bullet X - 2dc^T x + d^2 = 0$ Rule **LE2** $\sum_{j=1}^n c_j X_{ij} = dx_i \, \forall i \in \{1,...,n\}$ $c^T x = d$
$d' \leq c^T x \leq d$	Rule **LI1** $cc^T \bullet X - (d + d') \, c^T x + dd' \leq 0$ Rule **LI2** $\sum_{j=1}^n c_j X_{ij} \leq dx_i \quad \forall i \in \{1,...,n\}$ $d' x_i \leq \sum_{j=1}^n c_j X_{ij} \quad \forall i \in \{1,...,n\}$ $d' \, (1 - x_i) \leq \sum_{j=1}^n c_j \, (x_j - X_{ij}) \quad \forall i \in \{1,...,n\}$ $\sum_{j=1}^n c_j \, (x_j - X_{ij}) \leq d \, (1 - x_i) \quad \forall i \in \{1,...,n\}$

Table 1. *Rules to build a SDP relaxation from a linear relaxation*

The rules described in Table 1 are very general. x is a real vector of $\mathbb{R}^n$. Matrix X is such that X_{ij} is the linearization variable used in (P_L) which represents $x_i x_j$ (more precisely variables $z_{tpt'}$ in (LP_1) and $r_{tpt'p'}$ in (LP_2)). $\mathbf{d}(X)$ denotes the diagonal of the matrix X, and $A \bullet B = \sum_{i=1}^n \sum_{j=1}^n A_{ij} B_{ij}$, where A and B are two symmetric real matrices. It is well known that $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$ and $\mathbf{d}(X) = x$ imply $0 \leq x_i \leq 1$ for all i in $\{1, \ldots, n\}$ (see e.g. [27]).

In Table 1, $(SDP\{0,1\})$ is the semidefinite program built from a given linear program (P_L) (typically, a relaxation of the initial combinatorial problem). Rule **Q** applies for constraints made from the linearization of quadratic constraints (for instance any valid inequality included in (P_L)). Moreover, it allows us to keep in the SDP the linearization constraints (see Subsection 2.1), which are non-redundant as proved in [25]. Results presented in [20,26,29] imply that it is equivalent to use the set of constraints of Rule **LE2** or the set of constraints of Rule **LE1** (which obviously imply $c^T x = d$). But as noticed in [20], using **LE1** or **LE2** may have a serious impact on the convergence of the semidefinite solver used. In particular, for the Spectral Bundle algorithm [24], numerical tests presented in [29] show that despite of the larger number of constraints when one uses **LE2** (instead of **LE1**), the corresponding SDP is solved faster. The equivalence of the constraints in Rule **LE1** is true when $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$, and $\mathbf{d}(X) = x$.

Moreover, one can prove [29] that applying Rule **LI2** leads to stronger SDP relaxations than using **LI1** under some additional hypothesis. In particular, if the coefficients c_j are non-negative (see [29] for details). In all cases, the original linear constraints of (P_L) are satisfied by x if (X, x) is a feasible solution of $(SDP\{0,1\})$. Hence, the semidefinite relaxations presented in the next sections will be always tighter than the linear relaxations used to build them.

3.1. A Basic Semidefinite relaxation

First, we consider the following standard basic semidefinite relaxation made directly from (Q01), the $0-1$ formulation of our problem:

$$(SDP_0) \;:\; \text{Min } c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}X_{tpt'p} \tag{32}$$

Subject to:

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\dots,T \tag{33}$$

$$\sum_{t=1}^{T} s_t x_{tp} \le n_p \qquad p = 1,\dots,P \tag{34}$$

$$\mathbf{d}(X) = x \tag{35}$$

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \tag{36}$$

$$X_{tpt'p'} \ge 0 \quad p,p' = 1,\dots,P;\; t,t' = 1,\dots,T \tag{37}$$

In this relaxation, the only links between the linearization variables $X_{tpt'p'}$ and the original ones x_{tp} are constraints (35), and the semidefinite constraint (36), which is equivalent to $X \succeq xx^T$. Recall that this last constraint can be seen as a relaxation of $X = xx^T$. We denote by $S0$, the bound obtained by (SDP_0). In the next sections, by using the rules presented in [29], we present two better SDP relaxations starting respectively from the linear relaxations (LP_1) and (LP_2) of Section 2.

3.2. A tighter Semidefinite relaxation

Here, we consider the linear relaxation (LP_1), and build from it the semidefinite relaxation (SDP_1). We choose to apply Rule **Q** to (8), (9), and (11) (this provides constraints (44) and (45)), Rule **LE1** to (2) (this provides constraints (39) and (40)), and Rule **LI1** to (3) (this provides constraints (41)). As recalled in the previous Section, constraints (42) and (43) imply (10). We obtain the following semidefinite relaxation:

$$(SDP_1) \ : \ \text{Min } c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}X_{tpt'p} \tag{38}$$

Subject to:

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\dots,T \tag{39}$$

$$\sum_{p=1}^{P}\sum_{p'=1}^{P} X_{tptp'} = 1 \qquad t = 1,\dots,T \tag{40}$$

$$\sum_{t=1}^{T}\sum_{t'=1}^{T} s_t s_{t'} X_{tpt'p} - n_p \sum_{t=1}^{T} s_t x_{tp} \leq 0 \qquad p = 1,\dots,P \tag{41}$$

$$\mathbf{d}(X) = x \tag{42}$$

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \tag{43}$$

$$X_{tpt'p'} \leq x_{tp}\,;\ X_{tpt'p'} \leq x_{t'p'} \quad p,p' = 1,\dots,P;\ t,t' = 1,\dots,T \tag{44}$$

$$X_{tpt'p'} \geq 0 \quad p,p' = 1,\dots,P;\ t,t' = 1,\dots,T \tag{45}$$

Observe that constraints (41) and $X \succeq xx^T$ imply $\sum_{t=1}^{T} s_t x_{tp} \leq n_p \ \forall p \in \{1,\dots,P\}$ (for the proof see [29]). We denote by $S1$ the bound obtained by (SDP_1).

3.3. Our tightest Semidefinite relaxation

Here, we build a third semidefinite relaxation by applying again the algorithm presented in [29]. This time we start from the linear relaxation (LP_2). To obtain the SDP, we apply Rule **Q** to (12)-(15) and (17) (this provides constraints (49), (50) and (53) (considering the fact that $\mathbf{d}(X) = x$), and Rule **LE1** to (2) (this provides constraints (47) and (48)).

$$(SDP_2) \ : \ \text{Min } c_0 + \sum_{t=1}^{T}\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t=1}^{T}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}X_{tpt'p} \tag{46}$$

Subject to:

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\dots,T \tag{47}$$

$$\sum_{p=1}^{P}\sum_{p'=1}^{P} X_{tptp'} = 1 \qquad t = 1,\dots,T \tag{48}$$

$$\sum_{p=1}^{P} X_{tpt'p'} = x_{t'p'} \qquad t,t' = 1,\dots,T, p' = 1,\dots,P \tag{49}$$

$$\sum_{t=1}^{T} s_t X_{tpt'p'} \leq n_p x_{t'p'} \qquad p,p' = 1,\dots,P, t' = 1,\dots,T \tag{50}$$

$$\mathbf{d}(X) = x \tag{51}$$

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \tag{52}$$

$$X_{tpt'p'} \geq 0 \quad p,p' = 1,\dots,P;\ t,t' = 1,\dots,T \tag{53}$$

Observe that it is useless to apply Rule **LI1** to (2) to add the constraints $\sum_{t=1}^{T}\sum_{t'=1}^{T} s_t s_{t'} X_{tpt'p} - n_p \sum_{t=1}^{T} s_t x_{tp} \leq 0$ for $p \in \{1,\ldots,P\}$. Indeed, as remarked is Subsection 2.2, constraints (47), (49), and (50), imply $\sum_{t=1}^{T} s_t\left(x_{tp} - X_{tpt'p'}\right) \leq n_p\left(1 - x_{t'p'}\right)$ (for $p, p' = 1,\ldots,P, t' = 1,\ldots,T$). Since we have $s_t \geq 0$ and $\sum_{t=1}^{T} s_t X_{tpt'p'} \geq 0$, and constraints (50), we get $\sum_{t=1}^{T}\sum_{t'=1}^{T} s_t s_{t'} X_{tpt'p} - n_p \sum_{t=1}^{T} s_t x_{tp} \leq 0$ for $p \in \{1,\ldots,P\}$ (this result is proved in [29]).

We have chosen to keep constraints (48) although they are implied by constraints (47) and (49). Indeed, our numerical tests have shown this speeds up the convergence of SB, the SDP solver we have used [24]. We illustrate this point in Figure 1: the convergence is faster when one uses this redundant constraints (48). We denote by $S2$ the bound obtained by (SDP_2).

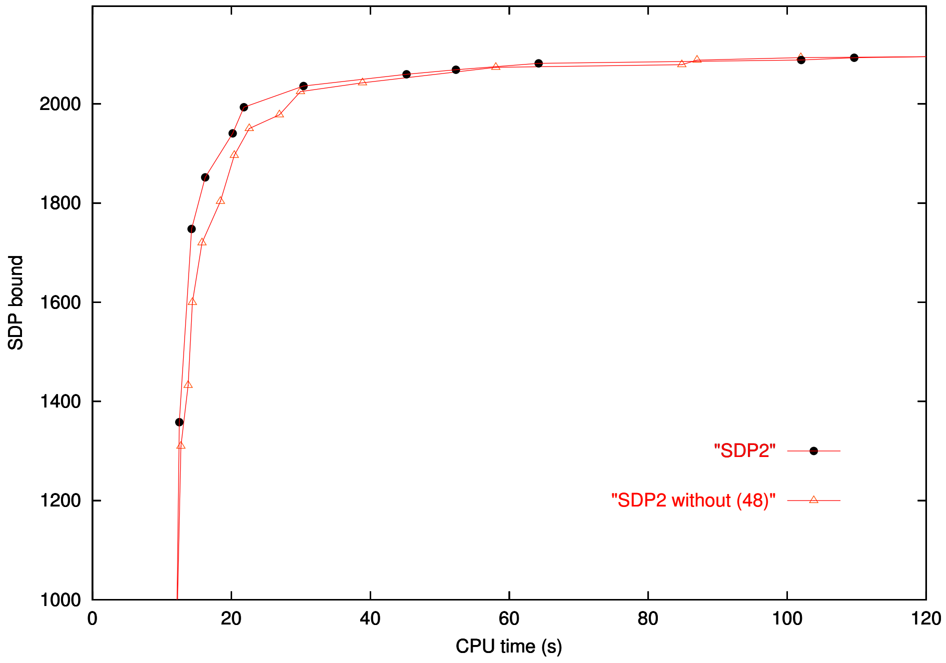

FIGURE 1. *Convergence of the SDP solver when using or not the redundant constraints (48)*

Proposition 3. *Let $S0$, $S1$ and $S2$ be the respective optimal values of (SDP_0), (SDP_1) and (SDP_2). One has $S0 \leq S1 \leq S2$.*

Proof: First, constraints (33), (35), (36) and (37) are in the three SDP. Second, constraint (40) in (SDP_1) is not in (SDP_0) but in (SDP_2) (48). Finally, constraints (34) (in (SDP_0)) are implied by (41) and (43) (in (SDP_1)), which are implied by (47), (50) and (52) (in (SDP_2)). Indeed, as recalled before, here the set of constraints produced by Rule **LI2** implies the one produced by Rule **LI1**. ■

Proposition 4. *One has $L1 \leq S1$ and $L2 \leq S2$.*

Proof: This is a consequence of the general result proved in [29]: the semidefinite relaxations obtained by following the algorithm are tighter than the linear program used to build them. ■

4. Lagrangian decomposition method for bound D0

This method was introduced by Chardaire and Sutter [10], then it was used by Faye [19], Billionnet et al. [8], and Elloumi et al. [18].

To each variable x_{tp} $(t = 1,\dots,T; p = 1,\dots,P)$ we will associate $(T-1)$ copies $y_{tp}^{t'}$ $(t' = 1,\dots,T, t' \neq t)$. We build the following quadratic 0-1 program, equivalent to (Q01):

$$\min\ c_0 + \sum_{t=1}^{T}\left(\sum_{p=1}^{P} q_{tp}x_{tp} - \sum_{t'=1,t'\neq t}^{T}\frac{1}{2}c_{tt'}\sum_{p=1}^{P} x_{tp}y_{t'p}^{t}\right) \tag{54}$$

Subject to:

$$x_{tp} = y_{tp}^{t'} \qquad 1 \le t \neq t' \le T;\ p = 1,\dots,P \tag{55}$$

$$x_{tp}y_{t'p'}^{t} = x_{t'p'}y_{tp}^{t'} \qquad 1 \le t < t' \le T,\ 1 \le p \neq p' \le P \tag{56}$$

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\dots,T \tag{57}$$

$$\sum_{p=1}^{P} y_{t'p}^{t} = 1 \qquad 1 \le t \neq t' \le T \tag{58}$$

$$s_t x_{tp} + \sum_{t'=1,t'\neq t}^{T} s_{t'}y_{t'p}^{t} \le n_p \qquad t = 1,\dots,T; p = 1,\dots,P \tag{59}$$

$$x_{tp} \in \{0,1\} \qquad t = 1,\dots,T; p = 1,\dots,P \tag{60}$$

$$y_{tp}^{t'} \in \{0,1\} \qquad 1 \le t \neq t' \le T;\ p = 1,\dots,P \tag{61}$$

Constraints (55) are the usual copy constraints. Constraints (56) are quadratic copy constraints. They are redundant. In its new formulation (54), the objective function can be decomposed into T sub-functions. For a given t, the corresponding sub-function uses only variables x_{tp}, and the copy variables $y_{t'p'}^{t}$, $(t' \neq t)$. This decomposition, for each module t, also applies for constraints (57)- (59). Hence, a natural idea is to apply a lagrangian relaxation scheme, by dualizing constraints (55) and (56). Let:

- $\lambda_{tpt'} \in \mathbb{R}$ be the Lagrange multipliers associated to constraints (55), and let
- $\mu_{tpt'p'} \in \mathbb{R}$ be the Lagrange multipliers associated to constraints (56).

The dual function $\omega(\lambda,\mu)$ can be decomposed into:

$$\omega(\lambda,\mu) = c_0 + \sum_{t=1}^{T} v\left(\mathcal{P}_t(\lambda,\mu)\right) \tag{62}$$

where $(\mathcal{P}_t(\lambda,\mu))$ is the following 0-1 quadratic program:

$$\begin{aligned}\min\ &\sum_{p=1}^{P}\left(q_{tp} + \sum_{t'=1,t'\neq t}^{T}\lambda_{tpt'}\right)x_{tp} - \sum_{p'=1}^{P}\sum_{t'=1,t'\neq t}^{T}\lambda_{t'p't}y_{t'p'}^{t} - \sum_{t'=1,t'\neq t}^{T}\tfrac{1}{2}c_{tt'}\sum_{p=1}^{P}x_{tp}y_{t'p}^{t}\\ &+ \sum_{t'=1,t'>t}^{T}\sum_{p=1}^{P}\sum_{p'=1}^{P}\mu_{tpt'p'}x_{tp}y_{t'p'}^{t} - \sum_{t'=1,t'<t}^{T}\sum_{p=1}^{P}\sum_{p'=1}^{P}\mu_{t'p'tp}x_{tp}y_{t'p'}^{t}\end{aligned} \tag{63}$$

Subject to: (57)-(61)

Now, we can observe that, for a given module t, if the values of the P variables x_{tp} are fixed, program $(\mathcal{P}_t(\lambda,\mu))$ becomes linear. Moreover, it is particularly easy to enumerate the different values for variables x_{tp}, since t is assigned to exactly one processor among P. There are P solution subsets, one for each p, where $x_{tp} = 1$ and $x_{tp'} = 0$ for $p' \neq p$.
Hence, the optimal value of $(\mathcal{P}_t(\lambda,\mu))$ can be computed as:

$$v\left(\mathcal{P}_t(\lambda,\mu)\right) = \min_{p=1,\dots,P} v\left(\mathcal{P}_t^{p}(\lambda,\mu)\right) \tag{64}$$

where $\mathcal{P}_t^p(\lambda,\mu)$ is obtained from $(\mathcal{P}_t(\lambda,\mu))$ by fixing x_{tp} to 1. It is precisely the integer linear program:

$$\begin{aligned}\mathcal{P}_t^p(\lambda,\mu)\ :\ \min\ & q_{tp} + \sum_{t'=1,t'\neq t}^{T} \lambda_{tpt'} - \sum_{p'=1}^{P}\sum_{t'=1,t'\neq t}^{T} \lambda_{t'p't} y_{t'p'}^t - \sum_{t'=1,t'\neq t}^{T} \tfrac{1}{2} c_{tt'} y_{t'p}^t \\ &+ \sum_{t'=1,t'>t}^{T}\sum_{p'=1}^{P} \mu_{tpt'p'} y_{t'p'}^t - \sum_{t'=1,t'<t}^{T}\sum_{p'=1}^{P} \mu_{t'p'tp} y_{t'p'}^t\end{aligned} \tag{65}$$

Subject to:

$$\sum_{p'=1}^{P} y_{t'p'}^t = 1 \qquad t' = 1,\dots,T,\ t'\neq t \tag{66}$$

$$\sum_{t'=1,t'\neq t}^{T} s_{t'} y_{t'p}^t \leq n_p - s_t \tag{67}$$

$$\sum_{t'=1,t'\neq t}^{T} s_{t'} y_{t'p'}^t \leq n_{p'} \qquad p' = 1,\dots,P, p'\neq p \tag{68}$$

$$y_{t'p'}^t \in \{0,1\} \qquad t' = 1,\dots,T,\ t'\neq t; p' = 1,\dots,P \tag{69}$$

Problem $\mathcal{P}_t^p(\lambda,\mu)$ represents a generalized assignment problem, which is NP-hard. We will compute only its LP-relaxation, in order to obtain a lower bound of the dual function.

Now, we can summarize our method of computing a lower bound of the dual function, for given values of the Lagrange multipliers: :

Computation of $\underline{\omega}(\lambda,\mu)$

```
- For each $t$:
    * For each $p$, solve the LP-relaxation $\underline{v}(\mathcal{P}_t^p(\lambda,\mu))$ of $\mathcal{P}_t^p(\lambda,\mu)$,
    * Deduce the lower bound of the optimal value of $\mathcal{P}_t(\lambda,\mu)$ as
      $\underline{v}(\mathcal{P}_t(\lambda,\mu)) = \min_{p=1,\dots,P} \underline{v}(\mathcal{P}_t^p(\lambda,\mu))$
- Compute a lower bound of $\omega(\lambda,\mu)$ as $\underline{\omega}(\lambda,\mu) = c_0 + \sum_{t=1}^{T} \underline{v}(\mathcal{P}_t(\lambda,\mu))$
```

Finally, we apply a classical sub-gradient algorithm in order to compute the lower bound D0 as:

$$D0 \quad = \quad \max_{\lambda,\mu\in\mathbb{R}^{TP}} \underline{\omega}(\lambda,\mu).$$

5. Computational results

Instances

We use instances from [16] that have been introduced in [17] and also used in [29]. In these instances, 4 configurations are considered. For each configuration, two classes of instances are generated: a class with a complete communication graph, and a second class where the density of the communication graph is of 50%. This gives a total of 8 types of instances. For each type, 5 instances of size 10 modules and 3 processors and 5 instances of 20 modules and 5 processors are generated. As we will see, the quality of the bounds will different for these different configurations.

	Config 1	Config 2	Config 3	Config 4
int_q	[0,100]	[0,10]	[0,100]	[0,0]
int_c	[0,100]	[0,100]	[0,10]	[0,100]

Table 2. Different configurations of the instances

Table 2 describes the different configurations. The execution costs q_{tp} are generated in the interval int_q, and the communication costs $c_{tt'}$ are generated in the interval int_c. For all the configurations, the sizes of the modules s_t are generated in the interval [1, 10], and the capacities of the processors n_p in the interval $[S/P,\ 2 * S/P]$ where $S = \sum_{t=1}^{T} s_t$ is the sum of all the module sizes. In this way, we are sure that problem (Q01) has at least one fractional solution $\tilde{x}$ where $\forall(t,p),\ \tilde{x}_{tp} = \frac{1}{P}$. The optimal or best known solution values of these instances are given in [16].

All the experiments have been performed on a Pentium II / 400 MHz computer with 384 MB memory. To implement bounds L1, L2, L3, and D0, we use the modeler AMPL [21] and the linear programs solver Cplex 7.0 [12]. For bounds S0, S1, and S2, we use the semidefinite programs modeler SDP_S [15] based on the semidefinite programs solver SB [24].

Results
In Table 3, we present the average relative gap between the optimal or best known solution value *opt* and each of the seven bounds we presented in Sections 2-4. The instances we consider in this table have a complete communication graph.

In Table 4 we report the computation times corresponding to the results presented in Table 3. Tables 5 and 6 concern instances with a half-complete communication graph.

First, we can observe that the quality of all the bounds is dependent on the type of instances. This gives an indication on the difficulty of the instances. The more difficult ones are those where there is no execution costs (Config 4), and the less difficult ones are those where execution costs are 10 times larger than communication costs (Config 3).

The three bounds obtained by linearization and described in Section 2 have a different computational behavior. As expected, bound L1 is very weak, except for instances of Config 3. It is equal to 0 (error=100%) for instances of Config 4. Computing L1 is however very quick, probably because of the structure of the linear program (LP_1). Bound L2 is the tightest one among those obtained by linearization, but it is the most expensive from the computational time point of view. Bound L3 appears to be a good compromise between bounds L1 and L2 since it is generally much better than than L1 and its CPU time is not much larger. Observe also that there exists instances where L3<L1, showing that these two bounds are not comparable.

Columns 7, 8, and 9 of Tables 3-6 concern the results of the three bounds obtained by SDP, described in Section 3. The SDP solver SB [24] is based on a spectral bundle method, each iteration of which computes a lower bound of the optimal value of the SDP (in the minimization case). Hence, whenever the solution procedure is stopped, it provides a lower bound for problem (Q01). We parametrise the solver to halt either because the optimal value is reached (with the SB-precision 10^{-5}), or after 1800 seconds of CPU time. By this choice, we emphasize on having an accurate idea on the quality of the lower bounds rather than minimizing their CPU time. This point is illustrated in Figure 2 where an example of convergence plot is presented, for the same instance as Figure 1. In this figure, it appears clearly that, when using SDP_2, convergence to an almost-optimal value is very rapid. This means that the resolution process could be stopped much earlier. Observe also that, for the considered instance, bound L2 is equal to 1144 and requires 85 seconds of CPU time. While solving SDP_2 (resp. SDP_1), the value 1144 is reached after 12 seconds (resp. 106 seconds). All this illustrates the power of bound S2, compared to the other bounds.

Another important remark about Tables 3-6 is the weakness of bound S0. It is even weaker than bound L1, while it requires much more CPU time. Also, bound D0 seems to be not very interesting to solve CMAP since it is not precise enough to compensate for its large computation time. However, D0 could be improved, for example by decomposing the objective function into les than T sub-functions.

	T	P	$\frac{(opt-L1)}{opt}$	$\frac{(opt-L2)}{opt}$	$\frac{(opt-L3)}{opt}$	$\frac{(opt-S0)}{opt}$	$\frac{(opt-S1)}{opt}$	$\frac{(opt-S2)}{opt}$	$\frac{(opt-D0)}{opt}$
Config 1	10	3	68%	12%	23%	69%	9%	3%	15%
	20	5	85%	12%	27%	85%	10%	2%	19%
Config 2	10	3	94%	29%	37%	94%	26%	12%	33%
	20	5	98%	17%	32%	98%	16%	3%	26%
Config 3	10	3	4%	1%	4%	11%	1%	1%	1%
	20	5	19%	1%	10%	31%	1%	0%	1%
Config 4	10	3	100%	25%	39%	100%	22%	8%	32%
	20	5	100%	18%	30%	100%	14%	3%	24%

TABLE 3. Average error associated to the seven bounds for the 40 instances with a complete communication graph

	T	P	$L1$	$L2$	$L3$	$S0$	$S1$	$S2$	$D0$
Config 1	10	3	1	1	1	6	948	884	30
	20	5	1	82	4	59	1800	1800	238
Config 2	10	3	1	1	1	144	998	890	25
	20	5	1	75	4	600	1800	1800	223
Config 3	10	3	1	1	1	3	96	274	19
	20	5	1	78	4	71	1800	1800	239.8
Config 4	10	3	1	1	1	2	798	568	27
	20	5	1	66	4	34	1800	1800	206

Table 4. Average CPU times of the seven bounds for the 40 instances with a complete communication graph (seconds rounded to the next integer)

	T	P	$\frac{(opt-L1)}{opt}$	$\frac{(opt-L2)}{opt}$	$\frac{(opt-L3)}{opt}$	$\frac{(opt-S0)}{opt}$	$\frac{(opt-S1)}{opt}$	$\frac{(opt-S2)}{opt}$	$\frac{(opt-D0)}{opt}$
Config 1	10	3	37%	21%	27%	41%	12%	5%	25%
	20	5	69%	29%	40%	70%	12%	3%	35%
Config 2	10	3	85%	48%	57%	85%	29%	11%	57%
	20	5	98%	17%	32%	98%	16%	3%	26%
Config 3	10	3	2%	1%	3%	4%	2%	1%	1%
	20	5	2%	0%	7%	12%	1%	0%	1%
Config 4	10	3	100%	43%	52%	100%	27%	13%	55%
	20	5	100%	56%	63%	100%	20%	9%	66%

Table 5. Average error associated to the seven bounds for the 40 instances with a half-complete communication graph

	T	P	$L1$	$L2$	$L3$	$S0$	$S1$	$S2$	$D0$
Config 1	10	3	1	1	1	4	575	458	26
	20	5	1	72	4	85	1800	1800	214
Config 2	10	3	1	1	1	12	860	898	25
	20	5	1	81	4	199	1800	1800	156
Config 3	10	3	1	1	1	4	189	156	19
	20	5	1	69	4	58	1800	1800	228
Config 4	10	3	1	1	1	3	1631	855	25
	20	5	1	69	4	34	1800	1800	127

Table 6. Average CPU times of the seven bounds for the 40 instances with a half-complete communication graph (seconds rounded to the next integer)

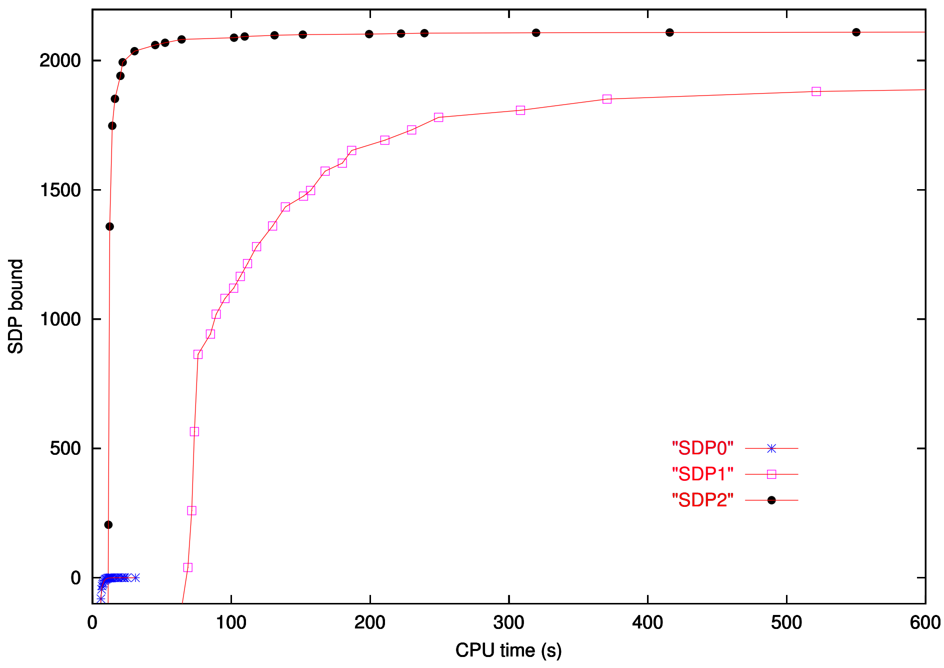

Figure 2. *Convergence of the SDP solver when computing S0, S1, and S2 for an instance with T=20, P=5 and a half-complete communication graph (optimal value=2316)*

6. Conclusion

In this paper, we compare seven lower bounds for the CMAP, a difficult combinatorial problem. The numerical experiments first give indications about the structural difficulty of the problem: the instances are much more difficult to solve when there are no execution costs and easier to solve when these execution costs are higher.

The seven bounds we consider come from three families of optimization techniques. In several cases, we could compare the bounds from a theoretical point of view. These theoretical comparisons are summed up with the graph in Figure 3. In this figure, arrows come from the different propositions we prove in this paper, and dashed lines are based on inspection of the computational results.

We complete the comparison between the seven bounds by an empirical study where we point out both the tightness and the CPU time of the bounds. Figure 4 illustrates our results. For each lower bound X, a CPU time is computed as the average of all the CPU times associated to X in Tables 4 and 6. The quality of bound X is measured as the "opposite" of the error, i.e., the complement to 100 of the average of all the errors associated to X in Tables 3 and 5. Having a look on Figure 4, one can immediately conclude which bounds one would choose if the dominant criterion in the CPU time, the quality, or a compromise.

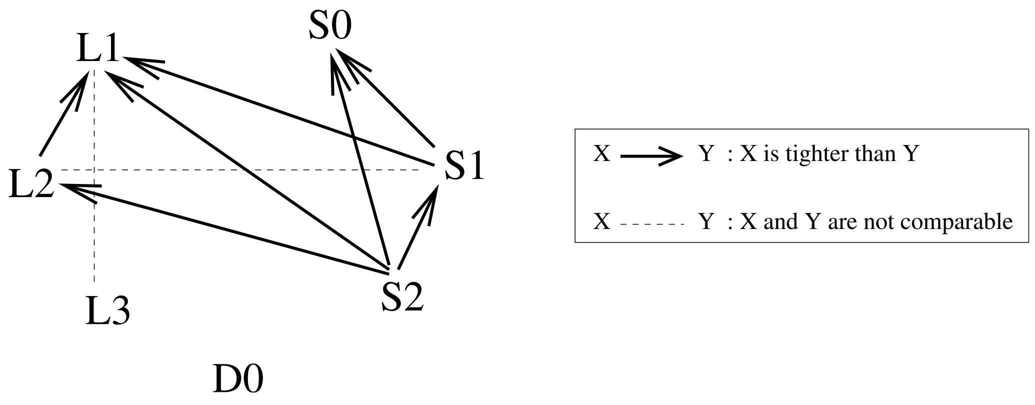
L1
S0
L2
S1
L3
S2
D0
X → Y : X is tighter than Y
X - - - - - - Y : X and Y are not comparable

Figure 3. Summary of the theoretical comparison of the seven bounds

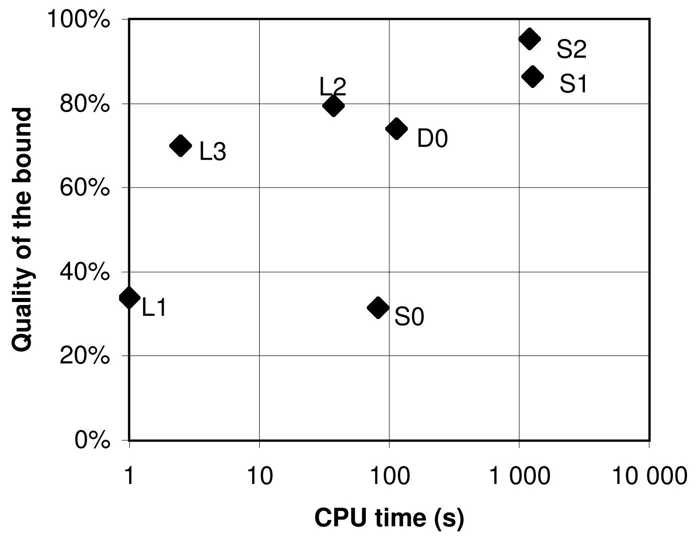

Figure 4. Empirical comparison of the seven bounds

References

[1] W. P. Adams and H. D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.

[2] R. K. Arora and S. P. Rana. Analysis of the module assignment problem in distributed computing systems with limited storage. *Information Processing Letters*, 10(3):111–115, April 1980.

[3] A. Billionnet, M. C. Costa, and A. Sutter. An efficient algorithm for a task allocation problem. *Journal of the ACM*, 39(3):502–518, July 1992.

[4] A. Billionnet and S. Elloumi. Placement de tâches dans un système distribué et dualité lagrangienne. *Revue d'Automatique, d'Informatique et de Recherche Opérationnelle (R.A.I.R.O.), série verte*, 26(1):83–97, 1992.

[5] A. Billionnet and S. Elloumi. Placement des tâches d'un programme à structure arborescente sur un réseau de processeurs : synthèse de résultats récents. *Information Systems and Operational Research (INFOR)*, 32(2):65–86, 1994.

[6] A. Billionnet and S. Elloumi. An algorithm for finding the K-best allocations of a tree structured program. *Journal of Parallel and Distributed Computing*, 26(2):225–232, 1995.

[7] A. Billionnet and S. Elloumi. Best reduction of the quadratic semi-assignment problem. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 109:197–213, 2001.

[8] A. Billionnet, A. Faye, and E. Soutif. A new upper bound for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 112:664–672, 1999.

[9] A. Billionnet and E. Soutif. Using a mixed integer programming tool for solving the 0-1 quadratic knapsack problem. Forthcoming in INFORMS Journal on Computing.

[10] P. Chardaire and A. Sutter. A decomposition method for quadratic zero-one programming. *Management Science*, 41(4):704–712, 1995.

[11] Maw-Sheng S. Chern, G. H. Chen, and Pangfeng Liu. An LC branch-and-bound algorithm for the module assignment problem. *Information Processing Letters*, 32(2):61–71, July 1989.

[12] Cplex. *ILOG CPLEX Division.* ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451.

[13] G.B. Dantzig. On the significance of solving linear programming problems with some integer variables, 1958. The Rand Corporation, document p. 1486.

[14] W. Fernandez de la Vega and M.Lamari. The task allocation problem with constant communication. *Discrete Applied Mathematics*, 131(1):169 – 177, 2003.

[15] G. Delaporte, S. Jouteau, and F. Roupin. Sdp_s: a tool to formulate and solve semidefinite relaxations for bivalent quadratic problems. In Proceedings ROADEF 2003, Avignon 26-28 Février 2003. *http://semidef.free.fr.*

[16] S. Elloumi. The task assignment problem, a library of instances. *http://cedric.cnam.fr/oc/TAP/TAP.html.*

[17] S. Elloumi. *Contribution à la résolution des programmes non linéaires en variables 0-1, application aux problèmes de placement de tâches dans les systèmes distribués.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 1991.

[18] S. Elloumi, A. Faye, and E. Soutif. Decomposition and linearization for 0-1 quadratic programming. *Annals of Operations Research*, 99:79–93, 2000.

[19] A. Faye. *Programmation quadratique en variables bivalentes sous contraintes linéaires. Application au placement de tâches dans les systèmes distribués et à la partition de graphes.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 1994.

[20] A. Faye and F. Roupin. Partial lagrangian and semidefinite relaxations of quadratic problems. In proceedings ROADEF'2005, Tours, 14-16 february 2005. Research report RC673, available at *http://cedric.cnam.fr.*

[21] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA, 1993.

[22] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22:455–460, 1975.

[23] Y. Hamam and K. Hindi. Assignment of program modules to processors: A simulated annealing approach. *European Journal of Operational Research*, 122:509–513, 2000.

[24] C. Helmberg. *A C++ implementation of the Spectral Bundle Method.* http://www-user.tu-chemnitz.de/ helmberg/SBmethod/. Version 1.1.1.

[25] M. Laurent, S. Poljak, and F. Rendl. Connections between semidefinite relaxations of the max-cut and stable set problems. *Mathematical Programming*, 77:225–246, 1997.

[26] C. Lemarechal and F. Oustry. Semidefinite relaxations and lagrangian duality with application to combinatorial optimization. RR-3710, INRIA Rhone-Alpes, ZIRST - 655 avenue de l'Europe, F-38330 Montbonnot Saint-Martin, June 1999.

[27] S. Poljak, F.Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0,1)-quadratic programming. *Journal of Global Optimization*, 7:51–73, 1995.

[28] F. Roupin. On approximating the memory-Constrained Module Allocation Problem. *Information Processing Letters*, 61(4):205–208, March 1997.

[29] F. Roupin. From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *Journal of Combinatorial Optimization*, 8(4):469–493, 2004.

[30] P. Ajith Tom and C. Siva Ram Murthy. An improved algorithm for module allocation in distributed computing systems. *Journal of Parallel and Distributed Computing*, 42(1):82–90, 10 April 1997.