Lower bounds for the Quadratic Semi-Assignment Problem

by

Federico Malucelli* and Daniele Pretolani*

* Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 PISA, Italy

Abstract: In this paper we will present class of new lower bounds for the Quadratic Semi-Assignment Problem (QSAP). These bounds are based on recent results about polynomially solvable cases, in particular we will consider the QSAP's whose quadratic cost coefficients define a *reducible graph*. Several lower bounds will be computationally compared, moreover we will present a method which improves these bounds by means of Lagrangean decomposition.

Keywords: Quadratic Semi-Assignment Problem, Lower Bounds, Lagrangean Decomposition.

Résumé: Dans ce papier on présente une classe de nouvelles limites inférieures pour le problème de semi-couplage quadratique (QSAP). Cettes limites se basent sur les récents résultats à propos de cases solvables en temps polynomial, en particulier on considère les QSAP dont les coefficients quadratiques definent un *graph réducible*. Plusieurs limites inférieures seront comparées computationellement; finalement on présente une nouvelle méthode pour améliorer les limites proposées en utilisant une décomposition de Lagrange.

Mots-clés: Semi-Couplage Quadratique Problème, Limites Inférieures, Décomposition de Lagrange.

1. Introduction

The Quadratic Semi-Assignment Problem (QSAP), has an important role in modelling many practical applications. For example clustering and partitioning problems [9], assignment of professors to departments [5], some scheduling problems [3]. Sometimes the model has been complicated to take into account "real life" factors as in [4, 15].

The QSAP is well known to be NP-hard [13]; some lower bounds for the problem have been devised in [6] and [5]. Polynomial classes are presented in [1, 2] studying distributed computing systems. In [10] a lower bound based on these ideas is applied to a branch and bound algorithm. In [11] the idea of new lower bounds is presented and the reported preliminary results show that the new bounds favourably compare with the one proposed in [10]. In this paper we intend to explore in depth the possibilities offered by this kind of approach.

In order to illustrate the problem we will use the following application example: consider a distributed computing system with p not necessarily identical processors, and n processes to be assigned to the processors. Let us call N the set of processes and M the set of processors. The following data are known:

- during the computation processes i and j exchange f_{ij} units of information;
- the time needed to move one unit of information from processor s to processor r is d_{rs};
- the computation time required by process i when it runs on processor s is e_{is}.

The *mapping problem* is that of assigning the processes to the processors so that the global time spent by the system (execution and communication time) is minimized. Let Π be the set of all the feasible assignment functions $\rho: N \rightarrow M$ which associate a processor $\rho(i) \in M$ to each process $i \in N$; the problem can be formulated as a QSAP as follows:

$$Z = \min \{ \sum_{i,j \in N} f_{ij} d_{\rho(i)\rho(j)} + \sum_{j \in N} e_{i\rho(i)}, \rho \in \Pi \}. \qquad (1.1)$$

The non zero f_{ij} coefficients define the communication pattern between processors that is usually represented by an undirected graph. As presented in [11] the optimal mapping can be found in polynomial time when this graph belongs to the class of *reducible graphs*.

The QSAP can be also formulated in a more general way: considering the matrix q_{ijhk} $i,j \in N$, $h,k \in M$, the problem is:

$$Z = \min \{ \sum_{i,j \in N} q_{ij\rho(i)\rho(j)}, \rho \in \Pi \}. \qquad (1.2)$$

The paper is organized as follows. In section 2 we will introduce the class of reducible graphs and devise a polynomial algorithm for solving instances of QSAP whose associated graph is reducible. In section 3 we will exploit these results to provide lower bounds. A generalization of this class of bounds which derive from a Lagrangean decomposition of the problem will be introduced in section 4. Finally in section 5, some preliminary computational results will be reported.

2. The class of reducible graphs

Consider the undirected graph $G(N,A)$ where $n=|N|$ and $m=|A|$; G is *reducible* if and only if it can be reduced to a single node by the following operations:

- *Tail reduction*

  let i be a node of degree 1 (i.e. there is only one arc incident with node i) and (j,i) be the arc connecting node i to the rest of the graph G. The graph G can be reduced to a new graph G' where node i and arc (j,i) have been deleted (see fig. 1). We will denote with Tail(i) the above reduction operation.

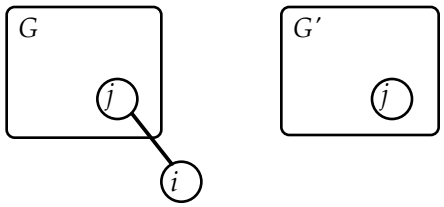

fig. 1

- *Series reduction*

  let i be a node of degree 2 and let (i,j) and (i,h), $j \neq h$, the two arcs incident with i; the graph G can be reduced to a new graph G' obtained from G by eliminating node i, arc(j,i), arc (i,h) and adding a new arc (j,h) (see fig. 2). This operation is denoted by Series(h, i, j).

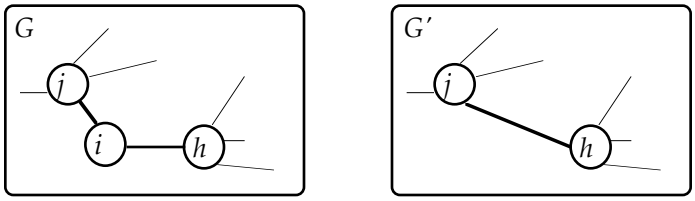

fig. 2

- *Parallel reduction*

  let $a=(i,j)$ and $a'=(i,j)$ be two "parallel" arcs of graph G. The graph can be reduced to a new graph G' with a single arc between nodes i and j (see fig. 3). This operation is denoted by Parallel(i, j).

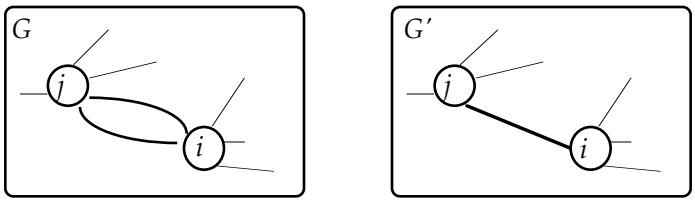

fig. 3

It is easy to see that the class of reducible graphs is a proper extension of the class of *series-parallel* graphs, which are reducible to a single arc by a sequence of parallel and series reductions ([16]). For example, trees are included in the class of reducible graphs, although they do not belong to the series-parallel class.

Let us state some properties of the reducible graphs which will be useful in the rest of the paper. For further detail we refer to [11].

Property 1

Suppose repeatedly applying reductions to a graph G as far as possible; the graph G' obtained is independent of the sequence of reductions.

This means that if different reductions are applicable to the same graph, we can break ties arbitrarily without affecting the final result. This establishes the *confluence* property of the reduction operations.

Property 2

A simple reducible graph G (i.e. without parallel arcs) contains at most $2n-3$ arcs.

An algorithm that recognizes reducible graphs can be easily obtained as follows: given an input graph G, repeatedly apply reductions to G as far as possible; G is reducible if and only if the resulting reduced graph contains a single node. An algorithm which runs in linear $O(m)$ time is described in [11].

2.1 A solution algorithm for QSAP on reducible graphs

Consider the undirected graph $G(N,A)$ where the set of nodes N is $\{1,\ldots,n\}$ (each node represents a process) and the set of arcs A is determined by the coefficients f_{ij}, that is $A=\{(i,j): f_{ij}>0 \text{ or } f_{ji}>0\}$; let $m=|A|$. Further on, we assume that G is connected; in fact, if G is not connected, one independent QSAP for each connected component of G can be identified.

When the graph G is reducible, the corresponding QSAP can be solved in polynomial time. In order to carry out the computation of the QSAP optimal solution, we introduce some labels associated to the nodes and the arcs of G. In particular we will associate the labels u_{ir} $\forall r\in M$ to each node $i\in N$, and the labels v_{irjs} $\forall r, s\in M$ to each arc $(i,j)\in A$. Initially these labels are set as follows:

$$u_{ir} = e_{ir},\ \forall i\in N,\ \forall r\in M,$$
$$v_{irjs} = f_{ij}d_{rs}+f_{ji}d_{sr},\ \forall\ i, j\in N,\ \forall r, s\in M.$$

Note that initially the labels associated to each node represent the set of all possible processor assignments for the related process, and their initial value is the execution time of the process on the different processors. The labels on an arc represent the set of all possible assignments for a pair of communicating processes; their initial value is the communication time.

Our solution method consists of updating the labels according to the reduction operations performed on G. At the end when G has been reduced to a single node, the minimum label u_{ir} $\forall r \in M$ t gives the optimal solution value.

The label updating can be described as follows:

- *Tail reduction*

  Let $i \in N$ be a node of degree 1 and $(j,i) \in A$ be the arc connecting i to the graph G. Labels u_{jr} are modified as follows, for each $r \in M$:

  $$u_{jr} = \min \{u_{is} + v_{isjr}, s \in M\}. \quad (2.1)$$

  In practice u_{jr} is modified in order to take into account the best possible assignment for i once j has been assigned to r. This operation can be carried out in $O(p^2)$ time.

- *Series reduction*

  Let $i \in N$ a node of degree two and let (i,j) and (i,l) be the two arcs incident with i. Labels v_{jslr} are set as follows, for each $r, s \in M$:

  $$v_{jslr} = \min \{v_{jrit} + v_{itls} + u_{it}, t \in M\} \quad (2.2)$$

  The new v_{jslr} takes into account the best possible assignment for i once j and l have been assigned to r and s, respectively. This operation can be carried out in $O(p^3)$ time.

- *Parallel reduction*

  Let $a'=(i,j)$ and $a''=(i,j)$ be two parallel arcs of graph G. Let v'_{irjs} and v''_{irjs} be the labels associated to a' and a'' respectively. The labels v_{jslr} of the new arc that will substitute a' and a'' in G' are obtained as follows:

  $$v_{irjs} = v'_{irjs} + v''_{irjs}, \forall r, s \in M.$$

  This operation can be carried out in $O(p^2)$ time.

Since at most $O(n)$ reduction operations are applied on a reducible graph G, the overall complexity of the transformations is $O(np^3)$.

The computation above gives the value of the optimal solution. In order to obtain an optimal assignment ρ associated to that value an extra computation is needed. To this end, we store in a stack the local choices we make in each Series or Tail reduction operation. Note that we do not need to store any information when executing a Parallel reduction, since this operation does not perform any assignment.

The information stored in the elements of the stack is the following:

- *Tail reduction*

  let (i,j) and i the arc and the node eliminated by the reduction, and for each v in M denote by $i(r)$ the index $s \in M$ giving the minimum in u_{jr} in (2.1). We put on the stack a label *Tail*, the nodes i and j, and the set $\{i(r): r=1,\ldots p\}$;

- *Series reduction*

  let (i,j) be the new arc introduced by the reduction and let h be the eliminated node; for each pair $r, s \in M$, denote by $h(r,s)$ the index $t \in M$ giving the minimum in (2.2). We put on the stack a label *Series*, the nodes i , j and h, and the set $\{h(r,s): r,s=1,\ldots,p\}$.

At the end of the reduction, let (i, r) be the minimum label of the remaining node; we set $\rho(i) = r$. Then, we repeatedly remove elements from the stack and, according to the label *Tail* or *Series*, we perform the following operations:

Tail let $r = \rho(j)$; set $\rho(i) = i(r)$;

Series let $r = \rho(i)$ and $s = \rho(j)$; set $\rho(h) = h(r,s)$.

It is easy to see that $\rho(j)$ ($\rho(i)$ and $\rho(j)$, respectively) has been already assigned when a Tail (Series, respectively) reduction is considered, hence the above method correctly finds an optimal assignment ρ.

3. Subgraph and Partition lower bounds

The existence of sharp and efficiently computable lower and upper bounds is a crucial part of enumerative algorithms. The problem has been widely studied in the literature; for example [6] and [5] present efficient methods for solving *relaxations* of particular formulations of QSAP.

The algorithm for polynomially solvable cases, presented in section 2, can be applied to obtain lower bounds. This approach has been partially exploited in [10], where in practice only tail reductions are performed (i.e. a spanning tree is extracted form G) and in [11], where some lower bound based on the extraction of reducible subgraphs from G are presented. In the following, we will review these methods, and we will present a new bound.

Let $G=(N,A)$ be a non reducible communication graph corresponding to a given QSAP problem; and let $G_r= (N, A_r)$ be a reducible subgraph of G. We can define a new problem *restricted* to graph G_r, in which only communication costs corresponding to arcs in A_r are considered; the objective function becomes:

$$\sum_{(i,j)\in A_r} f_{ij}d_{\rho(i)\rho(j)} + f_{ji}d_{\rho(j)\rho(i)} + \sum_{i\in N} e_{i\,\rho(i)}.$$

Assume that the quadratic costs corresponding to arcs in $A\backslash A_r$ are non-negative: it can be verified that the optimal solution Z_r of the problem restricted to G_r is a lower bound for the original problem. We call *subgraph bound* the value obtained in this way.

Consider the partition of the set of edges $A\backslash A_r$ in k subsets $A_1,\ldots,A_k$ such that each partial graph $G_l= (N, A_l)$, $1 \le l \le k$, is reducible, and the problems restricted to graphs G_l, in which linear costs are set to zero:

$$Z_l = \min \{ \sum_{(i,j)\in A_l} f_{ij}d_{\rho(i)\rho(j)} + f_{ji}d_{\rho(j)\rho(i)}: \rho\in\Pi\}.$$

In the light of the above decomposition, the optimal solution value of (1.1) can be written as:

$$Z = Z_r^* + \sum_{l}^{k} Z_l^* ,$$

where Z_r^* and Z_l^*, $1 \le l \le k$, are the costs of the optimal solution of (1.1) in the problems defined on graphs G_r and G_l, $1 \le l \le k$. It is easy to see that $Z_r \le Z_r^*$ and $Z_l \le Z_l^*$, hence the sum:

$$L = Z_r + \sum_{l}^{k} Z_l \le Z_r^* + \sum_{l}^{k} Z_l^* = Z$$

is a lower bound for the original problem. We call *partition bound* the value L obtained as above; it is easy to see that the restricted problems G_r and $G_l = (N, A_l)$, $1 \le l \le k$, can be solved with an overall $O(mp^3)$ complexity.

Note that the partition bound can be used also when the non negativity hypothesis of the quadratic costs is relaxed, while the subgraph bound can be used only if the quadratic costs are non-negative.

Consider the problem of determining the reducible subgraph G_r. In order to obtain a sharper bound, it is conceivable to search for a subgraph with a large set of arcs; moreover, arcs (i,j) corresponding to processes that exchange a large amount of information should be preferred. Thus one should find a reducible subgraph $G_r = (N, A_r)$ with maximum weight $W(G_r)$, where

$$W(G_r) = \sum_{(i,j)\in A_r} f_{ij} + f_{ji}.$$

The problem of finding the reducible subgraph with the maximum number of arcs is proved to be NP-hard in [11]. Thus the reducible subgraph of maximum cardinality or maximum weight cannot be easily identified. This is not true, however, if we require that G_r is a tree; in fact, many efficient algorithms for finding a maximum spanning forest in a graph are known [14]. Let us define *tree bound* the value obtained solving a restricted problem where G_r is a maximum spanning tree with respect to the weight $W(G_r)$; this bound can be determined in time $O(np^2)$. Moreover, in the partition bound one may think to consider only subgraphs of G which are trees; we will call *tree partition bound* the value obtained when G_r and G_l, $1 \le l \le k$, are trees. In this case the resulting complexity is $O(mp^2)$.

In practical applications, it is necessary to devise efficient heuristic algorithms to identify subgraphs with sufficiently large weight. However it must be observed that the maximum weight subgraph G_r does not always give the best lower bound. It is conceivable to require G_r to be *maximal*, i.e. that no arcs in $A \backslash A_r$ can be added to A_r obtaining a reducible graph. A trivial algorithm to find a maximal reducible subgraph has a $O(nm)$ complexity; an interesting problem is the one of finding a maximal reducible subgraph in less than $O(nm)$ time. Similar problems arise when a partition of the graph into k reducible subgraph is searched.

Note that in the partition bounds, all the linear costs are considered during the solution of the first subproblem. A possible variant of the bound could be that of considering the linear costs in subproblems other than the first one, or partition them among the various subproblems. The study of the best way of distributing the linear costs among the subproblems can be interpreted in the framework of the Lagrangean Decomposition techniques which will be discussed in the next section.

Now we propose a bound which has the same complexity of the partition bound, but it does not require to find an explicit partition of G. Consider a pair of arcs (i,j) and (i,l), suppose to replace them with a new arc (j,l). We call this operation *L-I reduction*. It easy to see that any graph can be reduced to a single node using tail, series, parallel and L-I reductions. We define the label updating corresponding to the L-I reduction as follows:

- *L-I reduction*

  Let $i \in N$ a node of degree at least two and let (i,j) and (i,l) be two arcs incident with i. Labels v_{jslr} are set as follows, for each $r, s \in M$:

  $$v_{jslr} = \min \{v_{jrit} + v_{itls} : t \in M\}$$

  The new v_{jslr} takes into account the best possible assignment for i once j and l have been assigned to r and s, respectively, without considering the linear costs (i.e. labels u_{ir}). This operation can be carried out in $O(p^3)$ time.

Consider the graph G' obtained from G performing a L-I reduction. The optimal solution of the QSAP associated to G' is not greater than the optimal solution of the QSAP associated to G. This follows from the fact that the contribution of the new arc (j,l) cannot be greater than the contribution of the pair (i,j) and (i,l). In order to obtain a lower bound, we repeatedly apply the reduction operations until G is reduced to a single node. Since at each step of the reduction process the optimal solution value of the resulting problem does not increase, the minimum label of the last

remaining node gives a lower bound. We will call *L-I bound* the value obtained using this method; the overall complexity of is $O(np^3)$. Obviously the value of the bound can be greatly affected by the selection of the reduction operation to perform at each step, and by the choice of the two arcs (i,j) and (i,l) a L-I reduction is applied to. In the following we will assume to perform a L-I reduction only when the others reductions cannot be applied; moreover for a L-I reduction we always select a node i of minimum degree.

4. Lagrangean decomposition

In this section we will propose a theoretical improvement of our lower bounds, introducing a *Lagrangean Decomposition* technique; in particular, our approach is a slight variant of the one introduced in [7]. Lagrangean decompositions have been often used in the literature ([8], [12]); this technique seems to be quite suitable when it allows to exploit the hidden structure of a problem, decomposing it into efficiently solvable subproblems.

In order to describe our approach, and the properties of the Lagrangean decomposition, it is useful to introduce the integer linear formulation of the QSAP:

$$\begin{aligned} Z = \min \quad & \sum_{i,j\in N} \sum_{r,s\in M} f_{ij} d_{rs} x_{ir} x_{js} + \sum_{j\in N, r\in M} e_{ir} x_{ir} \\ & \text{s.t.} \\ & x\in X=\{ \sum_{r\in M} x_{ir}=1,\ \forall I\in N,\ x_{ir}\in(0,1)\}\forall I\in N,\ \forall r\in M\} \end{aligned} \tag{4.1}$$

Variable x_{ir} is equal to one if and only if process i has been assigned to processor r.

Consider a generic decomposition of the matrix f such that $f=f^1+f^2$. Problem (4.1) can be rewritten as follows:

$$\min \quad \{ \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} + \sum_{j,r} e_{ir} x_{ir} + \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} x_{ir} x_{js} :\ x\in X\} . \tag{4.2}$$

Let us introduce a new set of variables y_{ir}, and the constraints $x_{ir}=y_{ir} \forall I\in N,\ \forall r\in M$. Then (4.2) becomes:

$$\min \quad \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} + \sum_{j,r} e_{ir} x_{ir} + \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} y_{ir} y_{js}$$
$$x, y \in X, \qquad (4.3)$$
$$x = y.$$

The Lagrangean relaxation of constraints $x=y$, introducing a multiplier λ_{ir} for each $i \in N$ and $r \in M$, defines the following Lagrangean function $L(\lambda)$ and the corresponding generalized dual problem *LD*:

$$L(\lambda) = \min \{ \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} + \sum_{i,r} (e_{ir} + \lambda_{ir}) x_{ir} : x \in X \} +$$
$$\min \{ \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} y_{ir} y_{js} - \sum_{i,r} \lambda_{ir} y_{ir} : y \in X \}$$

$$LD = \max_{\lambda} L(\lambda).$$

This approach can be generalized to any partition $f = f^0 + f^1 + \ldots + f^k$. In this case we must introduce $k+1$ sets of variables $\{x^0, x^1, \ldots, x^k\}$, and k set of constraints $x^{i-1} = x^i$, $i = 1, \ldots, k$; let $\{\lambda^1, \ldots, \lambda^k\}$ be the Lagrangean multipliers associated to these sets of constraints. The Lagrangean dual then becomes:

$$LD = \max_{\lambda^1, \ldots, \lambda^k} L(\lambda^1, \ldots, \lambda^k). \qquad (4.5)$$

This kind of decomposition allows to obtain efficiently solvable subproblems when the matrices f^0, f^1, …, f^k correspond to reducible subgraphs, in particular, when they define a partition of the communication graph into reducible subgraphs G^0, G^1, …, G^k. Now let *PB* be the value of the partition bound which uses the above decomposition; the following properties are straightforwad:

$$PB = L(\lambda^1, \ldots, \lambda^k) \;\; \lambda^i = 0, \;\; i = 1, \ldots, k;$$
$$LD \geq PB$$

Note that the linear costs appear only in the first subproblem associated to variables x^0. We might think to distribute these costs among the subproblems in a different way, as we suggested in the previous section. In the case of Lagrangean Decomposition this is equivalent to consider initializations of multipliers which are not all equal to zero.

The number of multipliers of $L(\lambda^1,\ldots,\lambda^k)$ can be extremely large (O(knp)). This could turn the solution of (4.5) into an intractable problem. In order to reduce the number of multipliers we can introduce a different kind of relaxation.

Consider the following constraints:

$$\sum_{r=1}^{p} rx_{ir} - \sum_{r=1}^{p} ry_{ir} = 0, \qquad i=1,\ldots,n \tag{4.6}$$

The following relation holds trivially for each x and $y \in X$:

$$x = y \;\Leftrightarrow \sum_{r=1}^{p} rx_{ir} - \sum_{r=1}^{p} ry_{ir} = 0, i=1,\ldots,n.$$

This suggest to study the following decomposition which is still equivalent to the original QSAP:

$$\begin{aligned} \min \quad & \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} + \sum_{j,r} e_{ir} x_{ir} + \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} y_{ir} y_{js} \\ & x, y \in X, \\ & \sum_{r=1}^{p} rx_{ir} - \sum_{r=1}^{p} ry_{ir} = 0, \qquad i=1,\ldots,n \end{aligned} \tag{4.7}$$

The Lagrangean function obtained relaxing constraints (4.6), and the coresponding generalized dual are:

$$\begin{aligned} L'(\mu) = \; & \min \{ \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} + \sum_{i} \mu_i \sum_{r} (e_{ir} + r) x_{ir} : x \in X \} \quad + \\ & \min \{ \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} y_{ir} y_{js} - \sum_{i} \mu_i \sum_{r} rx_{ir} : y \in X \} \\ LD' = \; & \max_{\lambda} L'(\mu). \end{aligned} \tag{4.8}$$

LD' is a lower bound for the QSAP, since (4.7) is equivalent to (4.1). Also in this case the result can be extended to a decomposition in more than two components. This kind of decomposition introduces a number of multipliers bounded by O(kn).

Theorem 4.1

$$LD \geq LD'$$

Proof:

Suppose that μ^* is an optimal solution of (4.8) and (x^*, y^*) is the corresponding solution of $L'(\mu^*)$. Let us define λ^* as follows:

$$\lambda_{ir}^* = \begin{cases} r\,\mu_i^* & \text{if } x_{ir}^* = 1 \text{ or } y_{ir}^* = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Note that $L'(\mu^*)$ and $L(\lambda^*)$ are equivalent hence (x^*, y^*) is an optimal solution also for $L(\lambda^*)$ and it gives $L'(\mu^*) = L(\lambda^*)$. $\Diamond$

The above theorem shows that *LD*' cannot give a lower bound better than *LD*, however this technique can be used to obtain a good initial solution of the stronger relaxation.

5. Numerical comparison of lower bounds

In the present section we will compare the lower bound presented in [10] with the lower bounds described in section 3; in particular we will focus on the effectiveness of the partition technique and the use of reducible graphs.

In the following tables **Tree** and **Red** are the subgraph bounds which use a spanning tree and a maximal reducible subgraph, respectively; **Tree_P**, and **Red_P** are the partition bounds which use a decomposition into trees and reducible subgraphs, respectively; **LI_Red** is the bound obtained using also the L-I reductions. Problems with n=50, and m={200, 300, 625} (tables 1, 3, 5) and n=100 and m={400, 1000, 2500} (tables 2, 4, 6) have been considered. Each table entry contains the average value of the bound over a sample of 10 instances.

Tables 1, 2, 3 and 4 report the results for QSAP of type (1.1), where d_{rs} ($r \neq s$) represent the distances on a mesh of size 2×4 (p=8), 4×4 (p=16) and 4×8 (p=32), f_{ij} are integer and uniformly distributed in [1..10]. In tables 1 and 2 the distance d_{rr} r=1,…,p, is uniformly distributed in [0..1]; the linear costs e_{ir} are equal to $\varepsilon_{ir}\delta$, where $\delta = mp/4n$ and ε_{ir} are integer and uniformly distributed in [1..10]. This implies that, on average, the total linear cost is of the same order of the total quadratic cost. In tables 3 and 4,d_{rr} r=1,…,p, are equal to the maximum distance; the linear costs e_{ir} are uniformly distributed in [1..10]. The choice of having nonzero distances d_{rr}, r=1,…,p, is suggested by the fact that if d_{rr}=0, r=1,…,p, the partition bound is equal to the subgraph bound. In fact for the

problems on graphs G_l, $1 \le l \le k$ an optimal solution $Z_l = 0$ can be obtained by assigning all processes to the same processor.

m	*p*	Tree	Red	Tree_P	Red_P	LI_Red
200	8	617.8	705.3	937.6	999.9	990.8
200	16	609.7	703.5	929.5	998.1	1008.6
200	32	797.8	985.0	1384.7	1517.1	1539.8
350	8	951.8	1071.0	1529.3	1615.7	1639.4
350	16	967.8	1102.3	1545.3	1647.0	1668.4
350	32	1237.9	1464.9	2306.9	2475.4	2563.6
625	8	1629.6	1766.1	2639.9	2743.6	2797.5
625	16	1577.4	1733.0	2587.7	2710.5	2779.2
625	32	2001.8	2284.1	3877.8	4096.2	4269.2

table 1: n**=50,** $d_{rr} \in$ **[0..1]**r**=1,…,**p**,** $e_{ir} = \varepsilon_{ir}\delta$**, with** $\delta = mp/4n$**,** $\varepsilon_{ir} \in$ **{1,…,10}**

m	*p*	Tree	Red	Tree_P	Red_P	LI_Red
400	8	1233.3	1360.4	1878.9	1962.2	1956.2
400	16	1227.8	1378.7	1873.4	1980.5	1972.6
400	32	1600.5	1859.1	2790.3	2973.0	3021.2
1000	8	2723.8	2924.5	4352.3	4495.4	4544.1
1000	16	2672.4	2911.7	4300.9	4482.6	4529.1
1000	32	3378.8	3753.5	6406.9	6682.6	6934.4
2500	8	6416.6	6698.1	10443.4	10653.8	10810.3
2500	16	6344.2	6668.0	10371.0	10623.7	10824.4
2500	32	8065.1	8600.8	15552.9	15958.2	16581.5

table 2: n**=100,**$d_{rr} \in$ **[0..1]**r**=1,…,**p**,** $e_{ir} = \varepsilon_{ir}\delta$**, with** $\delta = mp/4n$**,** $\varepsilon_{ir} \in$ **{1..10}**

m	*p*	Tree	Red	Tree_P	Red_P	LI_Red
200	8	216.3	310.6	809.4	852.9	848.3
200	16	194.8	291	787.9	833.3	823.6
200	32	220.1	366	1186.6	1263.8	1242.4
350	8	208.5	326.1	1343.4	1443.9	1439.2
350	16	186.7	309.4	1321.6	1427.2	1415.2
350	32	193	374.1	2027.5	2201.9	2179.5
625	8	205.8	349.5	2294.8	2505.9	2528.3
625	16	184.5	332.4	2273.5	2488.8	2504.6
625	32	180.6	392.2	3518.4	3894.8	3902.2

table 3: n**=50,** d_{rr} **max distance,** $e_{ir} \in$ **[1..10]**

m	*p*	Tree	Red	Tree_P	Red_P	LI_Red
400	8	435.8	573.8	1625.3	1692.5	1682.7
400	16	391.5	531.7	1581.0	1650.4	1624.7
400	32	444.4	646.7	2384.2	2495.6	2449.7
1000	8	411.6	613.9	3721.0	3951.6	3946.1
1000	16	370.3	578.7	3679.7	3916.4	3891.0
1000	32	368.9	669.7	5684.2	6097.5	6030.8
2500	8	411.2	703.7	8952.3	9808.0	9869.3
2500	16	370.0	673.1	8911.1	9777.4	9821.5
2500	32	358.5	777.3	13900.0	15356.2	15430.3

table 4:n=100, d_{rr} max distance $e_{ir} \in$ [1..10]

The above tables show that **Red** dominates **Tree**, and the difference between the two bounds is larger when linear costs are small (see tables 3 and 4).

Moreover the use of the partition technique is worthy, in particular when quadratic costs dominate the linear ones (see tables 3 and 4). Note that the partition is effective also when spanning tree are used; in fact the relative difference between **Red_P** and **Tree_P** is usually smaller than that of **Red** and **Tree**.

Tables 5 and 6 report the results for problems of type (1.2), where both the quadratic and the linear costs are uniformly distributed in [0..10].

m	*p*	Tree	Red	Tree_P	Red_P	LI_Red
200	8	99.6	126.3	103.7	141.0	128.8
200	16	53.9	77.2	53.9	77.7	55.0
200	32	40.9	77.1	40.9	77.6	48.9
350	8	97.4	139.3	104.0	182.1	177.1
350	16	52.6	86.7	52.6	91.2	73.8
350	32	41.5	89.6	41.5	96.9	78.6
625	8	95.4	148.9	110.5	264.4	291.5
625	16	51.7	97.5	51.7	118.7	124.2
625	32	40.8	107.2	40.8	144.0	166.6

table 5: n=50, $q_{ijrs} \in$ [0..10]

m	p	Tree	Red	Tree_P	Red_P	LI_Red
400	8	188.9	229.8	196.7	252.2	224.8
400	16	105.1	138.8	105.1	138.9	91.9
400	32	86.1	133.4	86.1	133.4	77.0
1000	8	190.9	263.7	218.2	394.2	363.1
1000	16	108.0	170.4	108.0	183.9	128.9
1000	32	85.7	172.1	85.7	191.6	131.3
2500	8	192.9	304.7	271.3	849.9	939.0
2500	16	106.1	199.2	106.1	326.4	352.5
2500	32	84.9	218.6	84.9	416.2	511.9

table 6: n=100, $q_{ijrs} \in$ [0..10]

For these problems, the partition technique is sometimes less effective than in the previous cases, in particular when p is large. Nevertheless **Red_P** gives good improvements when m is large. On the contrary the improvement of **Tree_P** on **Tree** is often negligible. This can be explained by the fact that, if $q_{ijrs} = 0$ occurs with high probability, the tail operation is likely to leave node labels unchanged.

Note that **LI_Red** is almost always equivalent to the **Red_P**, except for problems with a large number of arcs, where **LI_Red** tends to give better bounds.

In order to have better insights on how the bounds behave when the size of the problem changes, we made other computational experiments where we let n range between 20 and 50. For each group of problems we considered two values of p (i.e. p=2*3 and p=4*3) and three different classes of graphs: the sparse class where $m=n$, the medium class where $m=n^2/10$, and the dense class where $m=n^2/4$. The costs have been generated as for problems of tables 3 and 4. We set up a rudimental simulated annealing to produce feasible solutions; further on SA will denote the value of the feasible solution generated by the simuated annealing. We compared the differences SA-**Tree** and SA-**Red_P**. In tables 7 and 8 we reported the percent of gap **SA-Tree** closed by **Red_P**, that is (**Red_P**-**Tree**)/(SA-**Tree**)%. Each entry of the tables is the average over ten instances.

	sparse	medium	dense
20	45.9%	72.1%	78.1%
30	34.5%	70.5%	76.5%
40	31.7%	70.6%	75.6%
50	30.2%	71.3%	75.8%

table 7: gap closed by Red_P, *p*=2*3.

	sparse	medium	dense
20	30.3%	58.0%	66.0%
30	23.9%	58.4%	63.8%
40	21.7%	58.9%	63.7%
50	21.4%	59.5%	63.6%

table 8: gap closed by Red_P, *p*=4*3.

It should be noted that the relative behaviour of the bounds does not seem to be affected when n increases. On the contrary when p increases the gap between SA and **Tree** closed by Red_P decreases slightly. On the other hand when the number of arcs in the graph increases, **Red_P** becomes more effective with respect to **Tree**. In particular, for dense graphs **Red_P** closes between 63% and 78% of the gap. Finally we have to point out that the used heuristic is not very effective. In fact for problems with $m=n$, the bound provided by **Red_P** is very often equal the value of the optimal solution (i.e. the graph is reducible), while the value given by SA remains very large. This means that the effectiveness of **Red_P** with respect to **Tree** is underestimated, for example when $n=m$ the estimate of the closed gap is about 30% instead of being close to 100%.

References

1. Bokhari, S.H., "Assignment problems in parallel and distributed computing". Boston: Kluwer Academic Publishers.
2. Bokhari, S.H., "A shortest tree algorithm for optimal assignments across space and time in a distributed processor system". IEEE Transactions on Software Engeneering, 1981. SE-7: pp. 583-589.
3. Chretienne, P., "A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints". European Journal of Operational Research, 1989. 43: pp. 225-230.
4. Dixit, V.V. and D.I. Moldovan, "The allocation problem in parallel production systems". Journal of Parallel and Distributed Computing, 1990. 8: pp. 20-29.
5. Gallo, G. and B. Simeone, "Optimal grouping of researchers into departments". Ricerca Operativa, 1993. 57: pp. 45-69.
6. Gallo, G., E.M. Tomasin, and A.M. Sorato, "Lower bounds for the quadratic semi-assignment problem. 1986, Rutgers University, New Brunswick, NJ 08903
7. Guignard, M. and S. Kim, "Lagrangean decomposition: a model yielding stronger Lagrangean bounds". Mathematical Programming, 1986. 32: pp. 215-218.
8. Hansen, P., Methods of nonlinear 0-1 programming, in Annals of Discrete Mathematics. 1979, North-Holland: Amsterdam. pp. 55-70.
9. Hansen, P. and K. Lih, "Improved algorithms for partitioning problems in parallel, pipelined and distributed computing. 1989, RUTCOR.
10. Magirou, V.F. and J.Z. Milis, "An algorithm for the multiprocessor assignment problem". Operations Research Letters, 1989. 8: pp. 351-356.
11. Malucelli, F. and D. Pretolani, "Quadratic semi-assignment probems on structured graphs". Ricerca Operativa, 1994. to appear.
12. Michelon, P. and N. Maculan, "Lagrangean methods for 0-1 quadratic problems". Discrete Applied Mathematics, 1993. 42: pp. 257-269.
13. Sahni, S. and T. Gonzalez, "P-complete Approximation Problems". ACM Journal, 1976. (23): pp. 555-565.
14. Tarjan, R.E., "Data Structures and Network Algorithms". 1983, SIAM Publications.
15. Towsley, D., "Allocating programs containing branches and loops within a multiple processor system". IEEE Transactions on Software Engeneering, 1986. SE-12(10): pp. 1018-1024.
16. Valdes, J., E.L. Lawler, and R.E. Tarjan, "The recognition of Series Parallel digraphs". SIAM Journal on Computing, 1982. 11(2): pp. 299-313.