Lower bounds for the Quadratic Semi-Assignment Problem

by

Federico Malucelli* and Daniele Pretolani*

Abstract: This paper present a class of lower bounds for the Quadratic Semi-Assignment Problem (QSAP). These bounds are based on recent results on polynomially solvable cases, in particular we will consider the QSAPs whose quadratic cost coefficients define a *reducible graph*. The idea is to decompose the problem into several subproblems, each defined on a reducible subgraph. A Lagrangean decomposition technique is used to improve the results. Several lower bounds are computationally compared on several types of test problems.

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

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

1. Introduction

The Quadratic Semi-Assignment Problem (QSAP) plays an important role in modeling many practical applications, e.g. clustering and partitioning problems (Hansen and Lih 1989), assigning professors to departments (Gallo and Simeone 1991), and some scheduling problems (Chretienne 1989). Sometimes the model has been extended to take into account "real life" factors as in (Towsley 1986; Dixit and Moldovan 1990).

The QSAP is well known to be NP-hard (Sahni and Gonzalez 1976); tractable cases are presented in (Bokhari 1981; Bokhari 1987) which study distributed computing systems, and in (Malucelli 1994). In (Magirou and Milis 1989) a lower bound based on the solution of polynomially solvable subproblems is applied within a branch and bound algorithm. In (Malucelli and Pretolani 1994) this idea is further developed; the preliminary results presented in (Malucelli and Pretolani 1993) show that the new bounds favorably compare with the one proposed in (Magirou and Milis 1989). Other lower bounds based on Lagrangean relaxations have been devised in (Gallo, Tomasin et al. 1986), (Gallo and Simeone 1991) and in (Billionet, Costa et al. 1992).

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 N be 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 r to processor s is d_{rs};
- the computation time required by process i, when it runs on processor s, is e_{is}.

The *mapping problem* entails 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 $\rho(i) \in M$ with each process $i \in N$; the problem can be formulated as a QSAP as follows:

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

The non zero f_{ij} coefficients define the communication pattern between processes, usually represented by an undirected graph. As presented in (Malucelli and Pretolani 1994) the optimal mapping can be found in polynomial time when this graph belongs to the class of *reducible graphs*.

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

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

This paper is organized as follows. Section 2 introduces the class of reducible graphs and we devise a polynomial algorithm to solve instances of QSAP whose associated graph is reducible. In Section 3 we exploit these results to provide lower bounds. A generalization of this class of bounds which derives from a Lagrangean decomposition of the problem is introduced in Section 4. Computational results are reported in Section 5, while Section 6 concludes the paper along with some suggestions for future work.

2 QSAP on reducible graphs

2.1 Reducible graphs

Consider the connected 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 a sequence of 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). Tail(i) will denote the above reduction operation.

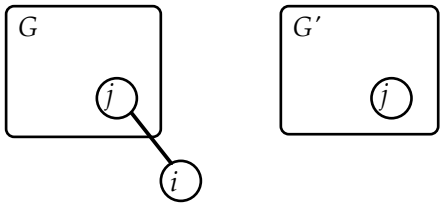

Fig. 1

- *Series reduction*

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

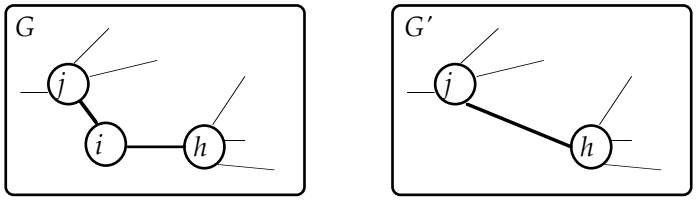

Fig. 2

- *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).

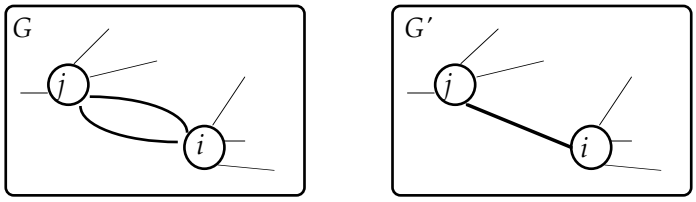

Fig. 3

The class of reducible graphs is clearly 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 (Valdes, Lawler et al. 1982). 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 details see (Malucelli and Pretolani 1994).

Property 1

Suppose that we can repeatedly apply reductions to a graph G as far as possible; graph G' thus obtained is independent of the sequence of reductions.

Property 1 means that if several 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 easily be 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 time is described in (Malucelli and Pretolani 1994).

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 $A=\{(i,j): f_{ij}>0 \text{ or } f_{ji}>0\}$. Later, 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 graph G is reducible, the corresponding QSAP can be solved in polynomial time. In order to carry out the computation of the optimal solution, we introduce some labels associated with the nodes and the arcs of G. Specifically, we will associate the labels u_{ir} $\forall r \in M$ with 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 with 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 various processors. The arc labels represent the set of all possible assignments for a pair of communicating processes; their initial value is the communication time.

Our solution consists of updating the labels according to the reduction operations performed on G. In the end when G has been reduced to a single node, the minimum label of that node 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 $(i,j) \in A$ be the arc connecting i to the graph G. Labels u_{jr} are modified as follows, for each $r \in M$:

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

  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_{jrls} are set as follows, for each $r, s \in M$:

  $$v_{jrls} := \min \{v_{itjr} + v_{itls} + u_{it}, t \in M\}. \quad (2.2)$$

  Here v_{jrls} gives 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 with a' and a'' respectively. The labels v_{irjs} of the new arc that will substitute a' and a'' 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 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 stacked information is:

- *Tail reduction*

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

- *Series reduction*

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

At the end of the reduction, let u_{ir} be the minimum label of the remaining node i; 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 already been assigned when a Tail (Series, respectively) reduction is considered, hence the above method finds an optimal assignment ρ.

3. Lower bounds

The problem of finding sharp and efficiently computable lower bounds for QSAP has been widely studied in the literature; for example (Gallo, Tomasin et al. 1986), (Gallo and Simeone 1991) and (Billionet, Costa et al. 1992) 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 was partially exploited in (Magirou and Milis 1989), and further developed in (Malucelli and Pretolani 1994). We review these methods below, and present a new bound based on a different kind of reduction.

3.1 Subgraph and Partition lower bounds

Let $G=(N,A)$ be a non reducible connected 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: we can verify that the optimal solution Z_r of the problem restricted to G_r is a lower bound for the original problem. We call the value thus obtained *subgraph bound.*

Consider a partition of the set of edges $A \backslash A_r$ in k subsets $A_1,\ldots,A_k$ such that each partial graph $G_l=(N, A_l)$, $l=1,\ldots,k$, is reducible; define the problems restricted to graphs $G_1,\ldots,G_k$, in which linear costs are set to zero:

$$Z_l = \min \{ \sum_{(i,j)\in A_l} f_{ij} d_{\rho(i)\rho(j)} + f_{ji} d_{\rho(j)\rho(i)} : \rho\in\Pi\}, \qquad l=1,\ldots,k.$$

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

$$Z = Z_r^* + \sum_{l=1}^{k} Z_l^*,$$

where Z_r^* and Z_l^*, $l=1,\ldots,k$, are the costs of the optimal solution of (1.1) in the problems defined on graphs G_r and G_l, $l=1,\ldots,k$. Clearly $Z_r \le Z_r^*$ and $Z_l \le Z_l^*$, hence the sum

$$L = Z_r + \sum_{l=1}^{k} Z_l \le Z_r^* + \sum_{l=1}^{k} Z_l^* = Z$$

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

Note that the partition bound can also be used when the non negativity hypothesis of the quadratic costs is relaxed, while the subgraph bound can only be used if the quadratic costs are non-negative. Furthermore, all the linear costs are considered during the solution of the first subproblem. A possible variant of the bound is to consider the linear costs in other subproblems besides the first one, or partition them among the various subproblems. The best way to distribute the linear costs among the subproblems can be studied within the framework of the Lagrangean Decomposition techniques which will be discussed in the next section.

3.2 Decomposition into reducible subgraphs

Consider the problem of determining the reducible subgraph G_r. In order to obtain a sharper bound, we could search for a subgraph with a large set of arcs; arcs (i,j) corresponding to processes that exchange a large amount of information are preferable. We should thus 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 has been shown to be NP-hard in (Malucelli and Pretolani 1994). Thus the reducible subgraph of maximum cardinality or maximum weight cannot easily be 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 tree in a graph are known (Tarjan 1983). Let us define as *tree bound* the value obtained by solving a restricted problem, where G_r is a maximum spanning tree with respect to the weight $W(G_r)$; this bound, proposed in (Magirou and Milis 1989), can be determined in time $O(np^2)$. Moreover, in the partition bound we could consider only subgraphs of G which are trees; we will call *tree partition bound* the value obtained when G_r and G_l, $l = 1,\ldots,k$, are trees. In this case the resulting complexity is $O(mp^2)$.

In practical applications, efficient heuristic algorithms are needed to identify subgraphs with sufficiently large weights. However, the maximum weight subgraph G_r does not always give the best lower bound. We may require G_r to be *maximal*, i.e. that no arcs in $A \backslash A_r$ can be added to A_r so that the reducibility will be maintained. A trivial algorithm to find a maximal reducible subgraph has an $O(nm)$ complexity; an interesting problem is to find a maximal reducible subgraph in less than $O(nm)$ time. Similar problems arise when we search for a partition of the graph into k reducible subgraphs.

3.3 L-I reduction

Next we propose a bound which has the same complexity as the partition bound, but does not entail finding an explicit partition of G. To this end, we introduce a new operation called *L-I reduction* which replaces a pair of arcs (i,j) and (i,l) with a new arc (j,l). This reduction does not guarantee that the connectivity of the graph will be maintained. In fact, any graph can be reduced

to a set of isolated nodes using tail, series, parallel and L-I reductions. Note that Property 1 does no longer hold if L-I reductions are used. We define a label updating that corresponds to the L-I reduction as follows:

- *L-I reduction*

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

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

  The new v_{jrls} take 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 by performing an L-I reduction. The optimal solution of the QSAP associated with G' is not greater than the optimal solution of the QSAP associated with G. Indeed, for any assignment the cost due to the new arc (j,l) cannot be greater than the cost of the pair (i,j) and (i,l). Suppose that we apply the reduction operations until G has been reduced to a set of isolated nodes. Since at each step of the reduction process the optimal solution value of the resulting problem does not increase, the sum of the minimum labels of the remaining nodes gives a lower bound. We call *L-I bound* the value obtained using this method; the overall complexity 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) an L-I reduction is applied to. Thereafter we will assume that an L-I reduction will only be performed when the other reductions cannot be applied; moreover, we always select for an L-I reduction a node i of minimum degree.

4. Lagrangean decomposition

This section outlines a theoretical improvement to our lower bounds by introducing a *Lagrangean Decomposition* technique; in particular, our approach is a slight variant of the one introduced in (Guignard and Kim 1986). Lagrangean decompositions have often been used in the literature (Hansen 1979; Hansen 1979; Michelon and Maculan 1993); this technique seems to be

quite suitable when it allows the hidden structure of a problem to be exploited by decomposing it into efficiently solvable subproblems.

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

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

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:

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

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:

$$\begin{aligned} \min \quad & \sum_{i,j} \sum_{r,s} f^1_{ij} d_{rs} x_{ir} x_{js} \; + \sum_{i,r} e_{ir} x_{ir} \; + \sum_{i,j} \sum_{r,s} f^2_{ij} d_{rs} y_{ir} y_{js} \\ & \text{s.t.} \\ & x, y\in X, \\ & x=y. \end{aligned} \tag{4.3}$$

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

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

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 with these sets of constraints. The Lagrangean dual then becomes:

$$LD = \max_{\lambda^1,\ldots,\lambda^k} L(\lambda^1,\ldots,\lambda^k). \quad (4.4)$$

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

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

Note that the linear costs only appear in the first subproblem, associated with variables x^0. These costs could be distributed among the subproblems in a different way, as we suggested in the previous section. In the case of Lagrangean Decomposition this is equivalent to considering initializations of multipliers which are not all equal to zero.

The number of multipliers of $L(\lambda^1,\ldots,\lambda^k)$ may be extremely large, namely O(*knp*). This could turn the solution of (4.4) into an intractable problem. Some ideas to reduce the number of multipliers are presented in (Malucelli and Pretolani 1993).

5. Numerical comparison of lower bounds

This section compares the lower bounds described in Sections 3 and 4. We investigate the effectiveness of the partition technique and the use of reducible graphs instead of trees; we then focus on the behavior of the partition bound and of the Lagrangean decomposition.

In the following tables **Tree** and **Red** denote the subgraph bounds which use a spanning tree and a maximal reducible subgraph; **Tree_P**, and **Red_P** are the partition bounds which use a decomposition into trees and reducible subgraphs; **LI_Red** is the bound obtained using the L-I reductions as well; **LD** is the value of the Lagrangean decomposition (4.4). In order to compute the optimal solution of the Lagrangean relaxation we use the *bundle trust region* algorithm developed in (Carraresi, Frangioni et al. 1994).

In the first group of experiments (Tables 1,…,6) we investigate the effects of various factors, such as the size of the problem, the density of the graph *G*, and the structure of the costs. To this aim, we consider quite large problems, namely *n*=50, *m*={200, 300, 625} (Tables 1, 3, 5) and *n*=100,

m={400, 1000, 2500} (tables 2, 4, 6). 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 a 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 $\delta\varepsilon_{ir}$, where $\delta = mp/4n$ and ε_{ir} are integer and uniformly distributed in [1..10]. This implies that the total linear cost is expected to be of the same order as the total quadratic cost. In Tables 3 and 4 d_{rr}, r=1,…,p, are equal to the maximum mesh distance; the linear costs e_{ir} are uniformly distributed in [1..10]. In this case, the contribution of linear costs is not expected to be very significant.

The decision to have non zero 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, $1 \leq l \leq k$ an optimal solution $Z_l = 0$ can be obtained by assigning all processes to the same processor. In the problems reported in Tables 3 and 4 this situation is avoided by setting a high value d_{rr}, r=1,…,p.

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

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

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

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

The above tables show that **Red** dominates **Tree**; as expected, the relative difference between the two bounds greater when quadratic costs dominate the linear ones (Tables 3 and 4). Moreover, the partition technique is worth applying, even in the least favorable case where linear costs are higher (Tables 1 and 2). Note that the partition is also effective when spanning trees are used; indeed, the

relative difference between **Red_P** and **Tree_P** is usually smaller than between **Red** and **Tree**. Finally, note that **LI_Red** is almost always equivalent to **Red_P**, but behaves better for problems with a large number of arcs, while **Red_P** tends to give better results for sparse graphs.

Tables 5 and 6 report the results for QSAP 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

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

For these problems, the partition technique is sometimes less effective than in the previous cases, in particular when p is large. In fact, **Red_P** gives significant improvements, especially when m is large; whereas the improvement of **Tree_P** on **Tree** is often negligible. In fact, for larger values of p, $q_{ijrs} = 0$ tends to occur with high probability, hence the tail operation is likely to leave node labels unchanged. The behavior of **LI_Red** is similar to the one shown in the previous tables.

In the following experiments, we tried to get a better insight into the behavior of lower bounds; we considered smaller problems, comparing the relative errors of lower bounds with respect to upper bounds. To obtain upper bounds (i.e. feasible solutions), we devised a heuristic procedure, based on a simple simulated annealing method. Hereafter UB will denote the value of the generated feasible solution. Metaheuristic methods for the QSAP are studied in (Domschke, Forst et al. 1992).

Tables 7 and 8 report the relative errors ε**(T)**=(UB-**Tree**)/**Tree** and ε**(R)=**(UB-**Red_P**)/**Red_P**, along with the percentage of gap UB**-Tree** closed by **Red_P**, that is **gap**=100(**Red_P**-**Tree**)/(UB-**Tree**).

We solved problems with n ranging between 10 and 50. For each value of n, we considered three different classes of graphs: **sparse**, where $m=n$, **medium**, where $m=n^2/10$, and **dense**, where $m=n^2/4$. We set p=2*3 in Table 7, and p=4*3 in Table 8. Costs were generated as for the problems in Tables 3 and 4.

p=2*3	**sparse**			**medium**			**dense**		
n	ε**(T)**	ε**(R)**	**gap**	ε**(T)**	ε**(R)**	**gap**	ε**(T)**	ε**(R)**	**gap**
10	0.238	0	100%	0.562	0.018	95.3%	1.561	0.073	89.0%
20	0.148	0.002	99.1%	0.960	0.039	92.5%	3.903	0.135	85.1%
30	0.144	0.007	95.9%	1.945	0.101	86.1%	5.921	0.130	86.5%
40	0.154	0.009	93.8%	2.882	0.119	85.7%	8.094	0.137	86.5%
50	0.159	0.012	92.4%	3.748	0.184	85.1%	9.958	0.136	86.9%

Table 7

p=4*3	**sparse**			**medium**			**dense**		
n	ε**(T)**	ε**(R)**	**gap**	ε**(T)**	ε**(R)**	**gap**	ε**(T)**	ε**(R)**	**gap**
10	0.312	0.004	98.8%	0.735	0.043	90.3%	1.833	0.095	86.7%
20	0.174	0.010	95.4%	1.204	0.098	83.7%	4.712	0.205	79.4%
30	0.163	0.006	96.9%	2.372	0.169	79.5%	7.398	0.250	77.3%
40	0.177	0.018	89.8%	3.530	0.207	78.1%	10.940	0.268	76.8%
50	0.197	0.026	85.6%	4.489	0.201	79.6%	12.582	0.281	76.3%

Table 8

As expected, **Red_P** outperforms **Tree**: in particular, in the dense case, the approximation given by **Tree** is meaningless, while the relative error provided by **Red_P** is reasonable; moreover, the approximation of **Red_P** is quite stable when n grows, which is not true in the case of **Tree**. In

the sparse case, the upper bound is very often equal to **Red_P**, and in the average UB and **Red_P** differs by less than 3% ; whereas the difference between UB and **Tree** is much more significant. Tables 9 and 10 report the relative errors ε(**R**) and ε(**LD**)=(UB-**LD**)/**LD**; here **gap**=100(**LD**-**Red_P**)/(UB-**Red_P**). We solved the same problems as in Tables 7 and 8, with *n* ranging between 10 and 30.

p=2*3	**sparse**			**medium**			**dense**		
n	ε(**R**)	ε(**LD**)	**gap**	ε(**R**)	ε(**LD**)	**gap**	ε(**R**)	ε(**LD**)	**gap**
10	0	0	–	0.018	0	100%	0.073	0.005	95.4%
20	0.002	0	100%	0.039	0.004	89.0%	0.135	0.046	63.7%
30	0.007	0.005	22.2%	0.101	0.020	78.8%	0.130	0.055	54.8%

Table 9

p=4*3	**sparse**			**medium**			**dense**		
n	ε(**R**)	ε(**LD**)	**gap**	ε(**R**)	ε(**LD**)	**gap**	ε(**R**)	ε(**LD**)	**gap**
10	0.004	0.002	50%	0.043	0.012	69.2%	0.095	0.026	73.5%
20	0.010	0.009	11.1%	0.098	0.037	61.0%	0.205	0.100	47.1%
30	0.006	0.003	58.3%	0.169	0.075	52.9%	0.250	0.153	33.7%

Table 10

In the sparse case the results are not very significant; in fact, as we noted before, **Red_P** is very close to the optimum, so there is little room for improvement. In the dense case the Lagrangean decomposition allows significant reductions to be obtained in the relative error, although this error increases rapidly when *n* grows. It would be interesting to investigate whether the error increase has to be related to the actual value of the duality gap, or if it may be due to a slow convergence of the dual method utilized, or to the poor quality of the upper bound.

All the algorithms were implemented in C and C++, and the experiments were made on an HP/9000-710 workstation. Since the execution time was not our main concern, we did not spend much time developing efficient codes. The parameters of the bundle algorithm had the same values for all the experiments: in particular, we fixed the maximum number of iterations to 500 and the bundle size to 150. With these settings the execution time for the largest problems did not exceed five minutes. Clearly a more careful analysis of the parameter setting could lead to more effective computations.

6. Conclusions and further work

In this paper we have presented two methods for improving lower bounds for the QSAP: a graph partition technique which allows the original problem to be decomposed into polynomially solvable subproblems, and the use of Lagrangean decomposition to combine the results of the individual subproblems. Our computational experiments show that both ideas are worth applying and lead to promising results. In addition, the introduction of the L-I reduction allows us to define a new lower bounding technique, whose results are comparable to those given by the partition bound.

A crucial point appears to be the efficient computation of the Lagrangean relaxation; in fact it might be interesting to investigate ad hoc procedures, as for example, multiplier adjustment techniques. Another interesting topic for further research is the study of heuristic methods to obtain feasible solutions for the QSAP, using the optimal solution of the Lagrangean decomposition as a starting point.

References

Billionet A., M. C. Costa and A. Sutter (1992). "An efficient algorithm for task allocation problem" **Journal of ACM** 39(3): 502-518.

Bokhari S. H. (1981). "A shortest tree algorithm for optimal assignments across space and time in a distributed processor system" **IEEE Transactions on Software Engineering** SE-7: 583-589.

Bokhari S. H. (1987). Assignment problems in parallel and distributed computing. Boston, Kluwer Academic Publishers.

Carraresi P., A. Frangioni and M. Nonato (1994). "A package based on the bundle idea for maximising a piecewise linear concave function", Dipartimento di Informatica - Università di Pisa.

Chretienne P. (1989). "A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints" **European Journal of Operational Research** 43: 225-230.

Dixit V. V. and D. I. Moldovan (1990). "The allocation problem in parallel production systems" **Journal of Parallel and Distributed Computing** 8: 20-29.

Domschke W., P. Forst and S. Voss (1992). *Tabu search techniques for the Quadratic Semi-Assignment Problem* in New Directions For Operations Research In Manufacturing Berlin, Springer. 389 - 405.

Gallo G. and B. Simeone (1991). "Optimal grouping of researchers into departments" **Ricerca Operativa** 57: 45-69.

Gallo G., E. M. Tomasin and A. M. Sorato (1986). "Lower bounds for the quadratic semi-assignment problem", Rutgers University, New Brunswick, NJ 08903, Technical Report DCS-TR-195.

Guignard M. and S. Kim (1986). "Lagrangean decomposition: a model yielding stronger Lagrangean bounds" **Mathematical Programming** 32: 215-218.

Hansen P. (1979). *Methods of nonlinear 0-1 programming* in Annals of Discrete Mathematics Amsterdam, North-Holland. 55-70.

Hansen P. and K. Lih (1989). "Improved algorithms for partitioning problems in parallel, pipelined and distributed computing", Rutcor, RRR #38-89.

Magirou V. F. and J. Z. Milis (1989). "An algorithm for the multiprocessor assignment problem" **Operations Research Letters** 8: 351-356.

Malucelli F. (1994). "A polynomially solvable class of quadratic semi-assignment problems", Dipartimento di Informatica - Università di Pisa.

Malucelli F. and D. Pretolani (1993). "Lower bounds for the quadratic semi-assignment problem", CRT - Université de Montreal, CRT-955.

Malucelli F. and D. Pretolani (1994). "Quadratic semi-assignment problems on structured graphs" **Ricerca Operativa** 69: 57-78.

Sahni S. and T. Gonzalez (1976). "P-complete Approximation Problems" **ACM Journal** (23): 555-565.

Tarjan R. E. (1983). Data Structures and Network Algorithms. SIAM Publications.

Towsley D. (1986). "Allocating programs containing branches and loops within a multiple processor system" **IEEE Transactions on Software Engeneering** SE-12(10): 1018-1024.

Valdes J., E. L. Lawler and R. E. Tarjan (1982). "The recognition of Series Parallel digraphs" **SIAM Journal on Computing** 11(2): 299-313.