A branch and bound algorithm for the minimum storage-time sequencing problem

Paolo Detti, Dario Pacciarelli

RT-DIA-36-1998

September 1998
ABSTRACT

The minimum storage-time sequencing problem generalizes many well known problems in Combinatorial Optimization, such as the directed linear arrangement and the problem of minimizing the weighted sum of completion times, subject to precedence constraints on a single processor. In this paper we propose a new lower bound, based on a Lagrangian relaxation, which can be computed very efficiently. To improve upon this lower bound, we employ a bundle optimization algorithm. We also show that the best bound obtainable by this approach equals the one obtainable from the linear relaxation computed on a formulation whose first Chvátal closure equals the convex hull of all the integer solutions of the problem.
1 Introduction

Let \( D = (N, A) \) be a digraph with node set \( N \) and edge set \( A \). Let \( n \) be the number of nodes and \( m \) be the number of arcs of the digraph. Let \( p_i \) denote the weight on node \( i \) and \( c_{ij} \) denote the weight on arc \( (i, j) \). In this paper nodes represent tasks to be scheduled on a single processor, and \( p_i \) indicates the time required to execute task \( i \). Arc \( (i, j) \) represents the fact that task \( j \) needs the result of task \( i \) in order to be performed, and \( c_{ij} \) represents the storage cost for time unit of the result generated by task \( i \) until it is consumed by task \( j \). We assume all the node weights to be nonnegative integers while the arc weights (i.e. the storage costs for time unit) are also integers but unrestricted in sign.

Let \( \mathbb{Z}_k \) denote the integer set \( \{1, 2, \ldots, k \} \). A feasible sequencing of the nodes of \( D \) is a \( 1-1 \) map \( \sigma : N \rightarrow \mathbb{Z}_n \) such that \( \sigma(j) > \sigma(i) \) for each arc \( (i, j) \in A \), where \( \sigma(i) \) indicates the position of task \( i \) in the sequencing \( \sigma \). Let \( \sigma^{-1}(k) \) indicate the task in position \( k \) in the sequencing \( \sigma \), then the length of an arc \( (i, j) \) with respect to sequencing \( \sigma \) is defined by \( c_{ij} \sum_{k=\sigma(i)+1}^{\sigma(j)} p_{\sigma^{-1}(k)} \). The minimum storage time sequencing problem (MSTS) consists of finding a sequencing with minimum total arc length. When \( p_i = 1 \), for all \( i = 1, \ldots, n \), and \( c_{ij} = 1 \) for every \( (i, j) \in A \), the problem is called directed linear arrangement problem (DLA) [6]. Even and Shiloach showed in [4] the NP-hardness of DLA. In the following we say that "\( j \) is sequenced after \( i \)" or "\( i \) is sequenced before than \( j \)" to mean \( \sigma(j) > \sigma(i) \).

For any sequencing \( \sigma \) let

\[
x_{ij} = \begin{cases} 
1 & \sigma(j) > \sigma(i) \\
0 & \text{otherwise}
\end{cases}
\]  

(1)

Then MSTS can be formulated as an integer program as follows:

\[
\min \sum_{(i,j) \in A} c_{ij} \left( \sum_{h=1}^{n} p_{hi} x_{ih} - \sum_{h=1}^{n} p_{jk} x_{jk} \right) 
\]  

(2)

subject to

\[
x_{ij} = 1 \quad (i, j) \in A
\]  

(3)

\[
x_{ij} + x_{ji} = 1 \quad i, j = 1, \ldots, n, i \neq j
\]  

(4)

\[
x_{ij} + x_{jk} + x_{ki} \leq 2 \quad i, j, k = 1, \ldots, n \quad i \neq j, j \neq k, i \neq k
\]  

(5)

\[
x_{ij} \in \{0, 1\} \quad i, j = 1, \ldots, n
\]  

(6)

Let us denote formulation (2)-(6) as \( IP_1 \). By removing constraints (3), the remaining set of constraints represents the well known linear ordering problem (LO) [19]. Here the objective function has the particular structure shown in (2), whereas in general LO may have any linear objective function. Let us denote by \( Q \) the MSTS polytope, i.e. the convex hull of all the incidence vectors of feasible solutions of MSTS, and by \( P \) the linear ordering polytope, i.e. the convex hull of all the incidence vectors of feasible solutions of LO. The following result shows that \( Q \) is a face of \( P \), i.e. it can be obtained simply by intersecting \( P \) with the hyperplanes \( x_{ij} = 1 \) for every \( (i, j) \in A \).

**Theorem 1.1**

\[
Q = P \cap \{ x \in \mathbb{R}^{|A|} : x_{ij} = 1, \text{ for all } (i, j) \in A \}
\]  

(7)
Proof. We will show that, all vertices of polyhedron \( Q \) are also vertices of \( P \). Consider a vertex \( \bar{x} \) of \( Q \). Clearly, \( \bar{x} \in P \), since it results from the convex combination of integer solutions of MSTS, that are also solutions of LO.

Let us suppose, by contradiction, that \( \bar{x} \) is not a vertex of \( P \). Then, there must exist two points \( x^1 \in P, x^2 \in P \) such that \( \bar{x} = \lambda x^1 + (1 - \lambda)x^2 \), for some real \( \lambda \in (0, 1) \).

Consider any component \( \bar{x}_{ij} = 1 \) of \( \bar{x} \) associated to some arc \((i, j) \in A \). Then, only two cases are possible: (i) \( x^1_{ij} = 1 \) and \( x^2_{ij} = 1 \), or (ii) \( x^1_{ij} > 1 \) and \( x^2_{ij} < 1 \) (or, equivalently, \( x^1_{ij} < 1 \) and \( x^2_{ij} > 1 \)). The second case is trivially impossible, since all points in \( P \) have components in \([0, 1]\), and therefore the case (i) must hold. By repeating the same argument to all components of \( \bar{x} \) correspondig to the arcs in \( A \), we obtain that both \( x^1 \) and \( x^2 \) belong to \( Q \), but this is a contradiction since it would imply that \( \bar{x} \) is not a vertex of \( Q \), and the thesis follows.

The linear ordering polytope has been studied by many authors in order to obtain good lower bounds based on the linear relaxation for some formulation of the problem. Among the others, Grötschel, Jünger, Reinelt [7, 19] and, more recently, Leung, Lee [11], Nutter and Penn [15].

MSTS has been studied in different particular cases and with different names, reflecting the area from which this optimization problem is derived. Adolphson and Hu [2] call optimal linear ordering the problem we refer to as DLA and proved it is polynomially solvable when the digraph \( D = (N, A) \) is a tree. Even and Shiloach proved the NP-completeness of DLA if \( D \) is a general acyclic digraph [4] while Ravi, Agrawal and Klein give an approximation algorithm for MSTS [18].

As shown in [2, 18], DLA (MSTS) is equivalent to the single-processor scheduling problem, to minimize the sum [the weighted sum] of completion times, subject to precedence constraints. For the \( 1/\text{pre}c/ \sum_{i=1}^n C_i \) problem Horn shows in [8] an algorithm equivalent to the one of Adolphson and Hu, and Lawler proves [10] that the problem \( 1/\text{pre}c/ \sum_{i=1}^n w_i C_i \) can be solved in polynomial time when the digraph \( D = (N, A) \) is “series-parallel”. Queyranne and Wang [17] give a complete description of the minimal linear system defining \( Q \) when the precedence constraints are series-parallel and Wolsey [23] compared different formulations for general acyclic precedence constraints.

In this paper we deal with the problem of finding exact algorithms for MSTS for a general digraph \( D = (N, A) \). While various heuristic methods are reported to work well for different applications, the lower bounding techniques are not usually satisfactory for processor scheduling. (The major difficulty is obtaining tight lower bounds in a reasonable computing time, which are needed to prove optimality and to design good algorithms based on the branch and bound approach.) A lower bound can be obtained by solving the linear relaxation for some formulations of the problem, nonetheless polyhedral methods have not been nearly so successful, at least for large size problems.

Different bounding techniques for MSTS are based on the Lagrangian relaxation of some constraints of the problem.

In [16] Potts presents a branch and bound algorithm for the \( 1/\text{pre}c/ \sum_{i=1}^n w_i C_i \) problem. A lower bound is obtained from formulation (3)-(6) by relaxing constraints (4) and (5). In [5] Hoogeveen and Vandevelde analyze the \( 1/\text{pre}c/ \sum_{i=1}^n w_i C_i \) problem by employing a different formulation in which variables are the job completion times \( C_i, i = 1, \ldots, n \). A branch and bound is presented where the lower bound is computed by relaxing all the
precedence constraints among the jobs. Slack variables are used in order to improve the lower bound.

In the next section we show a different formulation of the problem MSTS, similar to formulation (3)-(6), leading to a Lagrangian dual problem which can be solved in polynomial time. In Section 3 we make use of a subgradient and a bundle approach in order to improve the lower bound and compare the best bound obtainable by our relaxation with a linear relaxation approach. Section 4 illustrates some preprocessing rules, useful in order to reduce the size of the problem, and in Section 5 computational results are reported. In Section 6 we present the conclusions and possible enhancements of the proposed algorithm.

2 Lower bounds on the optimal solution

It is well known that the effectiveness of a branch and bound algorithm greatly depends upon the ability to fastly compute tight lower bounds at each step of the branch and bound. Let us formally define the tightness of an LP relaxation. Let us consider two different mixed integer formulations A and B of the same problem $P$. Let $P_A$ and $P_B$ denote the set of feasible solution of the LP relaxation of A and B respectively. We say that formulation A is tighter than B if $P_A \subset P_B$. In this section we consider a Lagrangian Lower Bound for MSTS.

A rooted spanning tree of digraph $D = (N,A)$ is a subgraph $T = (N,A_1)$ (if it exists) which is a rooted tree. Without loss of generality we suppose that a spanning rooted tree in the digraph $D$ always exists, at least by adding a dummy task $(n+1)$, with $p_{n+1} = 1$, and $n$ arcs $(i,n+1)$, with $c_{(i,n+1)} = 0$, for $i = 1, \ldots, n$.

Let $T = (N,A_1)$ be a rooted spanning tree for the digraph D and consider the formulation of MSTS given in (2)-(6).

A different formulation of MSTS is given by replacing constraints (3) with the following:

$$ \sum_{h=1}^{n} p_h x_{ih} - \sum_{k=1, k \neq j}^{n} p_k x_{jk} \geq p_j \quad (i,j) \in (A - A_1) \quad (8) $$

$$ x_{ij} = 1 \quad (i,j) \in A_1 \quad (9) $$

Let us denote formulation (2),(8),(9),(4)-(6) as $IP_2$. $IP_2$ is weaker than $IP_1$, since constraints (8) are implied from constraints (3)-(6). In fact, from (3), (5) we have

$$ x_{jk} + x_{ki} \leq 1 \quad (i,j) \in A, k \neq i, j $$

From (4), $x_{ki} = 1 - x_{ik}$, we have

$$ x_{ik} - x_{jk} \geq 0 \quad (i,j) \in A, k \neq i, j $$

$$ \sum_{k=1, k \neq i,j}^{n} p_k (x_{ik} - x_{jk}) \geq 0 \quad (i,j) \in A $$

5
Since \( x_{ij} = 1, x_{ji} = 0 \), we can write:

\[
\sum_{k=1}^{n} p_k x_{ik} - \sum_{k=1}^{n} p_k x_{jk} \geq p_j \quad (i, j) \in A
\]

The vice versa is not true in general. In fact, it is straightforward to find examples in which there are points feasible for the second formulation and unfeasible for the first one.

The smallest counterexample has four nodes and four arcs and it is shown in Figure 1. Here all \( p_i = 1, i = 1, \ldots, 4 \). If we replace the constraint \( x_{14} = 1 \) with the constraint \( x_{14} + x_{24} + x_{34} - x_{21} - x_{31} - x_{41} \geq 1 \), it is straightforward to verify that the solution \( x_{14} = x_{24} = 2/3; x_{41} = x_{42} = 1/3; x_{12} = x_{23} = x_{43} = x_{13} = 1 \), with the other variables equal to zero, is feasible for the linear relaxation of \( IP_2 \) while it is not feasible for the linear relaxation of \( IP_1 \). Actually it is not difficult to verify that, in this case, \( IP_1 \) is the convex hull of all the integer feasible solutions.

Nonetheless the second formulation is useful in order to find nice lower bounds on the optimal solution of the problem MSTS. Let consider the following Lagrangian relaxation, denoted as \( LR(\lambda) \), obtained by relaxing constraints (8):

\[
\min \sum_{(i,j) \in A} c_{ij}(\sum_{h=1}^{n} p_h x_{ih} - \sum_{h=1}^{n} p_h x_{jk}) + \sum_{(i,j) \in (A-A_1)} \lambda_{ij}(p_j - \sum_{h=1}^{n} p_h x_{ih} + \sum_{h=1}^{n} p_h x_{jk})
\]

subject to

\[
x_{ij} = 1 \quad (i, j) \in A_1
\]

\[
x_{ij} + x_{ji} \geq 1 \quad i, j = 1, \ldots, n, i \neq j
\]

\[
x_{ij} + x_{jk} + x_{ki} \leq 2 \quad i, j, k = 1, \ldots, n
\]

\[
x_{ij} \in \{0, 1\} \quad i, j = 1, \ldots, n
\]

\[
\lambda_{ij} \geq 0 \quad (i, j) \in (A - A_1)
\]

Formulation (10)–(15) can be solved in polynomial time for any fixed value of \( \lambda \). In fact the objective function (10) can be easily reduced to (2). Consider an arc \((i, j) \in A - A_1\)

Since \( A_1 \) is the arc set of a spanning rooted tree, the arc \((i, j)\) forms a cycle with the arcs of \( A_1 \) (called fundamental cycle), consisting of two paths starting respectively from nodes \( i \) and \( j \) and ending in a node \( q \), as shown in Figure 2. Indicate the nodes in the two chains as \( i = i_0, i_1, \ldots, i_{u-1}, i_u = q \) and \( j = j_0, j_1, \ldots, j_{v-1}, j_v = q \). The term \((\sum_{h=1}^{n} p_h x_{ih} + \sum_{h=1}^{n} p_h x_{jk})\) in (10) can be modified as follows:
\[
\sum_{h=1}^{\frac{n}{6}} p_h x_{ih} - \sum_{h=1}^{\frac{n}{6}} p_k x_{jk} = \sum_{h=1}^{\frac{n}{6}} p_h x_{ih} - \sum_{h=1}^{\frac{n}{6}} p_k x_{ik} + \\
+ \sum_{h=1}^{\frac{n}{6}} p_h x_{ih} - \sum_{h=1}^{\frac{n}{6}} p_k x_{j_{k-1}k} + \ldots + \sum_{h=1}^{\frac{n}{6}} p_h x_{i_{w-1}h} - \sum_{h=1}^{\frac{n}{6}} p_k x_{q_k} + \\
+ \sum_{h=1}^{\frac{n}{6}} p_h x_{qh} - \sum_{h=1}^{\frac{n}{6}} p_k x_{j_{w-1}k} + \sum_{h=1}^{\frac{n}{6}} p_h x_{j_{w-1}h} + \\
- \sum_{h=1}^{\frac{n}{6}} p_k x_{j_{w-2}k} + \ldots + \sum_{h=1}^{\frac{n}{6}} p_h x_{j_1h} - \sum_{h=1}^{\frac{n}{6}} p_k x_{jk}
\] (16)

By substituting all terms \((c_{ij} - \lambda_{ij})\left(\sum_{h=1}^{\frac{n}{6}} p_h x_{ih} - \sum_{h=1}^{\frac{n}{6}} p_k x_{jk}\right)\), for \((i, j) \in (A - A_1)\), in the objective function (10) with expression at second term in equation (16) we obtain an objective function only depending by the arcset \(A_1\), plus the term \(\sum_{(i, j) \in (A - A_1)} \lambda_{ij} p_j\) which is a constant for any fixed value of \(\lambda_{ij}\).

This fact implies that the solution of \(LR(\lambda)\) can be found in polynomial time for any fixed value of the Lagrangian multipliers \(\lambda_{ij}\). Hence, we are interested in finding the Lagrangian multipliers such that the lower bound on the optimal solution is maximized, i.e. we want to maximise \(LR(\lambda)\).

## 3 Lower bound improvement

We employ a subgradient algorithm and a bundle algorithm to improve the quality of the Lagrangian lower bound.

For a detailed description of the subgradient algorithm we refer, e.g., to [14]. In Figure 3 we roughly describe it. Here we simply recall that, for a given Lagrangian multiplier \(\lambda^*\), the problem of finding the optimal solution of the Lagrangian dual, by using, for example, the algorithm of Adolphson and Hu [2] can be solved in polynomial time. Moreover, \(LR(\lambda)\) is a convex function and, given a feasible point of \(LR(\lambda)\), a subgradient \(\sigma\) is given by equation (17).

![Figure 2: A fundamental cycle](image-url)
The Subgradient Algorithm

Input: formulation (10)–(15) of MSTS, starting point $\lambda^0$.

1. Initialize $t = 1$.
2. For a given $\lambda^{t-1}$, compute $x^t$ with the algorithm in [2].
3. For a given $x^t$, compute a subgradient $\sigma^t$ with equation (17) If $\sigma^t = 0$ then stop, otherwise go to Step 4.
4. Let $\lambda^t := \lambda^{t-1} + \delta_t \sigma^t$, for some step $\delta_t$ such that $\lim_{t \to \infty} \delta_t = 0$, $\sum_{t=1}^{\infty} \delta_t = \infty$
5. Let $t := t + 1$ and return to Step 2.

Figure 3: The Subgradient Algorithm

$$\sigma_{ij} = p_j - \sum_{h=1, h \neq i}^{n} p_h x_{ih} + \sum_{h=1, h \neq j}^{n} p_k x_{jk}, \quad \forall (i, j) \in (A - A_1) \quad (17)$$

A well known problem when applying subgradient methods to discrete optimization is its slow convergence to the optimal solution. In order to overcome this problem we employed the bundle optimization algorithm developed by Schramm and Zowe [20].

The Bundle Algorithm

Input: formulation (10)–(15) of MSTS, starting point $\lambda^0$.

1. Initialize $t = 1$.
2. Compute a descent direction $d_t$ by solving a quadratic problem
3. Perform a linesearch for $f$ along $\lambda_t + \theta d_t$, $\theta \geq 0$.
4. If the linesearch leads to a "sufficient decrease" in $f$, then put $\lambda_{t+1} = \lambda_t + \theta d_t$, $\theta \in \arg \min_{\theta > 0} f(\lambda_t + \theta d_t)$, and compute a subgradient $\sigma_{t+1} \in \partial f(\lambda_t + 1)$.
5. Otherwise, i.e. if the linesearch yields only an "insufficient decrease", put $\lambda_{t+1} = \lambda_t$, compute a subgradient $\sigma_{t+1} \in \partial f(\lambda_t + d_t)$.
6. Let $t := t + 1$ and return to Step 2.

Figure 4: The Bundle Algorithm

In the bundle methods (in Figure 4 we roughly describe it), at the generic iteration $t$, a quadratic problem is solved to compute a descent direction, say $d_t$, for the functional. In our computational experience, this method turned out to be particularly efficient for
a reduced size of lagrangian vector. In the optimization of a functional \( f(\lambda) \), bundle methods make use of the ”bundle” of information of the subgradient vector \( \sigma_t \) and of the functional values \( f(\lambda_t) \). At iteration \( t \), all values \((\sigma_t, f(\lambda_t)), (\sigma_{t-1}, f(\lambda_{t-1}))\)... collected so far are employed to reconstruct the ”shape” of the objective function \( f \) in a neighborhood of \( \lambda_t \). Unlike the subgradient approach the algorithm of Schramm and Zowe guarantees a decrease of \( f(\lambda) \) at each step, it has a stopping criterion and in general it ensures a faster convergence speed. For a more detailed description of the bundle methods, we refer to [20].

In the remainder of this section we give a result showing how the solution of the Lagrangian dual relates with the original problem MSTS. In order to do this, let us introduce a new relaxation of \( IP_2 \), denoted as \( LP_1 \):

\[
\begin{align*}
\min & \quad \sum_{(i,j) \in A} c_{ij} \left( \sum_{h=1 \atop h \neq i}^n p_h x_{ih} - \sum_{h=1 \atop h \neq j}^n p_h x_{jk} \right) \\
\text{subject to} & \\
\sum_{h=1 \atop h \neq i}^n p_h x_{ih} - \sum_{h=1 \atop h \neq j}^n p_h x_{jk} & \geq p_j \quad (i, j) \in (A - A_1) \\
x_{ij} & = 1 \quad (i, j) \in A_1 \\
x & \in P
\end{align*}
\]

We first show that the first Chvátal closure of \( LP_1 \) is \( Q \), i.e. is the convex hull of all the incidence vectors of feasible solutions of MSTS. Then we will show that the optimal solution of the Lagrangian dual equals that of \( LP_1 \).

We recall from section 1 that the convex hull \( Q \) of all the feasible solutions of MSTS is given by equation (7). Hence, here we only have to show that equalities \( x_{ij} = 1 \), for all \((i, j) \in (A - A_1)\) belong to the first Chvátal closure of \( LP_1 \).

From (4), (5), (19) (which are all valid inequalities for \( LP_1 \)) we have

\[
x_{ik} + x_{kj} + x_{ji} \leq 2 \quad i, j, k = 1, \ldots, n \quad i \neq j, j \neq k, i \neq k
\]
and therefore, since \( x_{kj} + x_{jk} = 1 \), \( x_{ij} + x_{ji} = 1 \), we have

\[
x_{ik} - x_{jk} \leq x_{ij} \quad i, j, k = 1, \ldots, n \quad i \neq j, j \neq k, i \neq k
\]
Equation (19) can be therefore rewritten as

\[
p_j x_{ij} - p_i x_{ji} + \sum_{h=1 \atop h \neq i,j}^n p_h (x_{ih} - x_{jh}) \geq p_j \quad (i, j) \in (A - A_1)
\]
and, from (23) and (4), we have

\[
p_j x_{ij} - p_i + p_i x_{ij} + \sum_{h=1 \atop h \neq i,j}^n p_h (x_{ij}) \geq p_j \quad (i, j) \in (A - A_1)
\]
Hence
\[ \frac{p_i + p_j}{\sum_{h=1}^{n} p_h} > 0 \quad (i, j) \in (A - A_1) \] (26)

Inequalities (26) are valid for \( LP_1 \), and therefore, by rounding up the right hand side in (26), inequalities \( x_{ij} \geq 1 \), for all \((i, j) \in (A - A_1)\), must belong to the first Chvátal closure of \( LP_1 \) (see [3]). Hence, the following lemma is proved.

**Lemma 3.1** The first Chvátal closure of \( LP_1 \) coincides with the convex hull \( Q \) of all the incidence vectors of feasible solutions of MSTS.

A polyhedron is said to have the *integrality property* if all its vertices are integers. Let us now recall a well known result in polyhedral theory [14]:

**Theorem 3.2** Let \( P \) be an integer linear programming problem, \( LR(\lambda) \) be a Lagrangian relaxation of \( P \) and \( LP(x) \) be a linear relaxation of \( P \), then \( \max \{LR(\lambda)\} \geq \min \{LP(x)\} \). The equality holds if the linear relaxation of the Lagrangian formulation possesses the integrality property.

We observe that \( LP_1 \) is a linear relaxation of \( IP_2 \), and its optimal solution is smaller or equal than the optimal solution of the Lagrangian relaxation of \( IP_2 \) obtained by relaxing constraints (19). Note that the remaining polyhedron, \( \{x \in P : x_{ij} = 1 \text{ for all } (i, j) \in A_1\} \), has integral vertices (it follows immediately from Theorem (1.1)), i.e. it possesses the integrality property.

Hence, the following result is proved.

**Theorem 3.3** The best bound obtainable from \( LR(\lambda) \) equals the one obtainable from \( LP_1 \), whose first Chvátal closure is the polytope \( Q \), convex hull of all the incidence vectors of feasible solutions of MSTS.

### 4 Preprocessing rules

In this section we consider some preprocessing transformations in order to reduce the number of nodes and arcs of the precedence graph. A transformation for a precedence graph \( G_1 = (N_1, A_1) \) is a procedure for constructing a precedence graph \( G_2 = (N_2, A_2) \) so that any optimal permutation for \( G_2 \) yields an optimal permutation for \( G_1 \). We first compute the transitive reduction of the precedence graph, to cut all redundant arcs. An arc \((i, j)\) is redundant if there is already a path from \( i \) to \( j \) which does not use \((i, j)\). In the MSTS problem, the transitive reduction of the digraph \( D = (N, A) \) is computed by cutting each redundant arc \((i, j)\) and adding the cost \( c_{ij} \) to all the arcs of a path from \( i \) to \( j \). An important class of preprocessing rules are the transformations derived by Adolphson [1] and by Monma [12] for the sequencing problem \( 1/prec/ \sum w_i C_i \). These transformations are based both on the set of precedence relationships and on the tasks weights.

Adolphson and Hu [2] showed, first, the equivalence between DLA problem and \( 1/prec/ \sum C_i \). Lawler [9] established the equivalence between the Minimum Storage Time Sequencing problem and \( 1/prec/ \sum w_i C_i \) for arbitrary acyclic digraph. In fact, a MSTS instance can be reduced to a sequencing problem considering the same digraph \( D = (N, A) \), and, for each node \( i \), the same processing times \( p_i \) and the weight \( w_i = 1/2[\sum_{(h,i) \in A} c_{hi} - \sum_{(i,h) \in A} c_{ih}] \).
To describe Adolphson results we first need to introduce some terminology. Let \( G = (N, A) \) be the digraph obtained from \( D \) by removing the weights \( c_{ij} \) on the arcs and by adding the weights \( w_i \) on the nodes. Let \( P(j) \) be the set of all predecessors of node \( j \) in \( G \), and \( S(j) \) be the set of all successors of node \( j \). Let \( IP(j) \) and \( IS(j) \) be the sets of immediate predecessors and immediate successors of node \( j \). For each node \( i \) let \( r_i \) be the ratio \( w_i / p_i \). The notation for \( w_i, p_i \) and \( r_i \) can also be extended to subsets \( J \) of nodes in the following manner: \( w_J = \sum_{j \in J} w_j, p_J = \sum_{j \in J} p_j, \) and \( r_J = \sum_{j \in J} w_j / p_j \).

We say that a node \( j \) is maximal if \( r_j = \max \{ r_k : k \in S(IP(j)) \cup IP(j), k \notin S(j) \} \). The following theorem allows to reduce the number of nodes in the precedence graph.

**Theorem 4.1** If \( j \) is maximal, then an optimal sequence exists containing the subsequence \((i, j)\) for some \( i \in IP(j) \).

The proof is reported in Adolphson [1]. From the theorem 4.1 two transformation can be easily derived. We call condensation of two nodes \( i \) and \( j \) the transformation obtained by substituting the two nodes with a unique node \( k \), where \( IP(k) = IP(i) \cup IP(j) \), and \( IS(k) = IS(i) \cup IS(j) \).

\( T1. \) If node \( j \) is maximal and \( IP(j) = \{ i \} \), then we may condense nodes \( i \) and \( j \).

If \( j \) is maximal and \( |IP(j)| > 1 \) it is not possible to apply \( T1 \). Suppose however that all immediate predecessors of \( j \) have the same set of successors. Then the previous theorem tell us that there must be an optimal ordering in which \( j \) precedes all other successors of \( IP(j) \). The second transformation may be summarized as follows:

\( T2. \) If node \( j \) is maximal, \( |IP(j)| > 1 \), and \( S(i) = S(IP(j)) \) for all \( i \in IP(j) \), then let \( j \) precede all other nodes of \( S(IP(j)) \) in the precedence graph.

By taking advantage of the natural symmetry of the problem, a simmetric result can be easily derived. we say that a node \( j \) is minimal if \( r_j = \min \{ r_k : k \in P(IS(j)) \cup IS(j), k \notin P(j) \} \).

**Theorem 4.2** If \( j \) is minimal, then an optimal sequence exists containing the subsequence \((i, j)\) for some \( i \in IS(j) \).

From the previous theorem we obtain two transformations \( T'1 \) and \( T'2 \):

\( T'1. \) If node \( j \) is minimal and \( IS(j) = \{ i \} \), then we may condense nodes \( i \) and \( j \).

\( T'2. \) If node \( j \) is minimal and \( |IS(j)| > 1 \), and \( P(i) = P(IS(j)) \) for all \( i \in IS(j) \), then let \( j \) succeed all other nodes of \( P(IS(j)) \) in the precedence graph.

Adolphson in [1] reported an \( O(n^3) \) algorithm scheme based on \( T \) and \( T' \) transformations. His algorithm terminates when either an optimal sequence is found or when none of the transformations can be applied. In particular, it guarantees to find a optimal sequence on series-parallel digraphs.

Transformations \( T1 \) and \( T'1 \) are employed at root and in each node of the branch and bound algorithm. In fact, after each branching, it could be possible to condense new nodes of the partially ordered digraph. In our computational experience \( T2 \) and \( T'2 \) resulted to be less efficient than \( T1 \) and \( T'1 \) both for computational time and quality. Hence, these transformation are only applied at root and are not implemented in the nodes of the branch and bound.
Table 1: Performance of Algo 1 for random graphs

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P$</th>
<th>Algo 1</th>
<th>MDE</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.001</td>
<td>(5)</td>
<td>(5)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>(5)</td>
<td>(4)</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>(5)</td>
<td>(1)</td>
<td>12.0</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>(4)</td>
<td>(1)</td>
<td>13.26</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>(0)</td>
<td>(0)</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>(5)</td>
<td>(0)</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>(5)</td>
<td>(0)</td>
<td>1.06</td>
</tr>
<tr>
<td>40</td>
<td>0.001</td>
<td>(5)</td>
<td>(5)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>(5)</td>
<td>(2)</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>(4)</td>
<td>(1)</td>
<td>10.6</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>(0)</td>
<td>(0)</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>(0)</td>
<td>(0)</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>(1)</td>
<td>(0)</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>(5)</td>
<td>(2)</td>
<td>0.2</td>
</tr>
<tr>
<td>50</td>
<td>0.001</td>
<td>(5)</td>
<td>(5)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>(5)</td>
<td>(1)</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>(1)</td>
<td>(0)</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>(0)</td>
<td>(0)</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>(0)</td>
<td>(0)</td>
<td>1.73</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>(1)</td>
<td>(0)</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>(5)</td>
<td>(0)</td>
<td>0.2</td>
</tr>
<tr>
<td>60</td>
<td>0.001</td>
<td>(5)</td>
<td>(5)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>(5)</td>
<td>(0)</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>(0)</td>
<td>(0)</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>(0)</td>
<td>(0)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>(0)</td>
<td>(0)</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>(0)</td>
<td>(0)</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>(5)</td>
<td>(1)</td>
<td>0.14</td>
</tr>
</tbody>
</table>

5 Computational experience

We tested the performance of our algorithm (referred to in this section as Algo 1) on two sets of problem instances. The first set is generated as in [16, 5]. Processing times and arc weights were generated from a discrete uniform distribution, while the precedence graph was induced by probability $P$: a random number $r_{ij}$ was generated from the uniform distribution $[0, 1]$ and arc $(i, j)$ was included in $D$ if and only if $r_{ij} \leq P$. The algorithm was tested on problems with $n = 30, 40, 50, 60$ nodes and $P = 0.001, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5$. For each value of $n$ and $P$ five instances were generated. Table 1 shows in the third column the number of times our algorithm (Algo 1) is able to find a proven optimal solution within a limit of 2000 nodes. Column 4 reports the number of times the tree optimal heuristic of Morton and Dharan (MDE) [13] finds an optimal solution. Finally, column 5 reports the improvement of Algo 1 against MDE.

Since Algo 1 takes advantage from the particular structure of the precedence graph $D$, it is expected to give good performance whenever the graph $D$ has a structure similar to a
tree. Hence, we built a second set of instances where the precedence graph was obtained by adding to a tree a limited number of arcs. In Table 2 the results are reported for this class of graphs. Column 1 reports the number of nodes \( n \) and the number of arcs \( m \) of the graph \( D \). The number of nodes generated by the algorithm and the optimal value of the solution are reported in column 2 and 3, respectively.

A measure of the density of the precedence graph \( D \), similar to the probability \( P \), is the order strength, which is defined as the ratio of the number of arcs in the transitive closure of \( D \) to the maximum number \( n(n-1)/2 \) of possible arcs. Order strength has been frequently used in the literature as a predictor of problem difficulty, due to its correlation with the total number of feasible schedules. From the computational experiences of various authors \([16, 5]\) for \( 1/\text{prec} \sum u_iC_i \), it results that the most difficult problems are those whose order strength is between 0.25 and 0.5.

Nonetheless, we observe that it is difficult to evaluate both exact and heuristic techniques with no standard test sets. In fact, as usual in most combinatorial problems, the same algorithm can perform very differently, even on similar instances. Let us consider, for example, the set of problems generated as follows: the number of nodes and the arc set of the precedence graph \( D \) are the same for all instances. Node weights are all equal to 1. Arc weights are also the same for all instances but they are associated with different arcs, more specifically they are rotated, i.e. rotation \( i \) is obtained by assigning weight of arc \( j \) in Table 5 to arc \( (i + j) \mod n \). Notice that the order strength for this graph is approximately 0.5.

In Table 3 computational results are reported for the 29 task - 37 arcs problem shown in Figure 5. \( LB \) is the initial lower bound on the problem, and \( Opt \) is the best value found by the algorithm within a limit of 5000 nodes (values which are not proven optimum are labeled with a star). Here we report the results on 10 different rotations. As it can be seen, there is no consistent pattern. The problem is solved at root in rotation 0 while in rotation 7 computation was abandoned after 5000 nodes had been generated. Even the gap between the initial lower bound and the optimum (computed as \( \frac{Opt - LB}{Opt} \)) does not give a good prediction of problem difficulty: the largest gap in rotation 28 is closed after 111 nodes, while smaller gaps (for example in rotation 3, 7 or 9) are closed only after a larger number of nodes.

Similar results can be obtained for the 29 nodes - 35 arcs obtained from the previous graph by removing, for example, arcs 8 and 18 (in Table 4 the computational results for the first 15 rotations are reported). The order strength of the resulting graph is now approximately 0.38, and therefore all these problems could be classified as difficult.

As a concluding remark for our computational experiences, we point out the need for a

<table>
<thead>
<tr>
<th>( n - m )</th>
<th>Nodes</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>42-55</td>
<td>67</td>
<td>2271</td>
</tr>
<tr>
<td>52-67</td>
<td>82</td>
<td>2545</td>
</tr>
<tr>
<td>62-78</td>
<td>147</td>
<td>2951</td>
</tr>
<tr>
<td>72-90</td>
<td>327</td>
<td>3350</td>
</tr>
<tr>
<td>82-101</td>
<td>344</td>
<td>3753</td>
</tr>
<tr>
<td>92-113</td>
<td>674</td>
<td>4152</td>
</tr>
<tr>
<td>102-125</td>
<td>1329</td>
<td>4551</td>
</tr>
</tbody>
</table>

Table 2: Performance of Algo 1 for graphs obtained from trees
standard set of challenging test problems for single processor scheduling with precedence constraints.

<table>
<thead>
<tr>
<th>Rotation</th>
<th>gap (%)</th>
<th>Nodes</th>
<th>LB</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1495</td>
<td>1495</td>
</tr>
<tr>
<td>3</td>
<td>9.39</td>
<td>2616</td>
<td>1457.95</td>
<td>1609</td>
</tr>
<tr>
<td>7</td>
<td>14.91</td>
<td>5000</td>
<td>2112.62</td>
<td>2483*</td>
</tr>
<tr>
<td>9</td>
<td>7.6</td>
<td>356</td>
<td>1576.94</td>
<td>1707</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>64</td>
<td>1267.4</td>
<td>1424</td>
</tr>
<tr>
<td>16</td>
<td>10.23</td>
<td>34</td>
<td>1545.8</td>
<td>1722</td>
</tr>
<tr>
<td>24</td>
<td>11.44</td>
<td>249</td>
<td>1490.5</td>
<td>1639</td>
</tr>
<tr>
<td>28</td>
<td>24.37</td>
<td>111</td>
<td>1133.7</td>
<td>1499</td>
</tr>
<tr>
<td>33</td>
<td>5.3</td>
<td>8</td>
<td>1538.83</td>
<td>1625</td>
</tr>
<tr>
<td>35</td>
<td>19.88</td>
<td>100</td>
<td>1497.3</td>
<td>1869</td>
</tr>
</tbody>
</table>

Table 3: Performance of Algo 1 for the 29 nodes-37 arcs graph

6 Conclusions and future research

In this section we suggest some possible enhancements of the proposed algorithm. In this paper we illustrate a Lagrangian relaxation of MSTS, obtained by searching a spanning rooted tree on the graph $G$. We do not make use of the results by Lawler [10] which could allow us to relax a smaller number of arcs simply by searching a spanning series-parallel digraph in the graph $G$, and therefore by relaxing a smaller number of arcs in the graph.
<table>
<thead>
<tr>
<th>Rotation</th>
<th>Nodes</th>
<th>$LB$</th>
<th>$Opt$</th>
<th>MDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1327.64</td>
<td>1477</td>
<td>1491</td>
</tr>
<tr>
<td>1</td>
<td>325</td>
<td>1748.42</td>
<td>1983</td>
<td>2005</td>
</tr>
<tr>
<td>2</td>
<td>141</td>
<td>1280.8</td>
<td>1378</td>
<td>1429</td>
</tr>
<tr>
<td>3</td>
<td>617</td>
<td>1378.23</td>
<td>1518</td>
<td>1527</td>
</tr>
<tr>
<td>4</td>
<td>5000</td>
<td>1397.36</td>
<td>1550*</td>
<td>1561</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>1151.7</td>
<td>1688*</td>
<td>1688</td>
</tr>
<tr>
<td>6</td>
<td>5000</td>
<td>1444.75</td>
<td>1864*</td>
<td>1978</td>
</tr>
<tr>
<td>7</td>
<td>5000</td>
<td>2020.42</td>
<td>2243*</td>
<td>2277</td>
</tr>
<tr>
<td>8</td>
<td>1149</td>
<td>1768.6</td>
<td>2077</td>
<td>2101</td>
</tr>
<tr>
<td>9</td>
<td>714</td>
<td>1460.64</td>
<td>1684</td>
<td>1690</td>
</tr>
<tr>
<td>10</td>
<td>418</td>
<td>1176.6</td>
<td>1349</td>
<td>1769</td>
</tr>
<tr>
<td>11</td>
<td>832</td>
<td>1159.63</td>
<td>1451</td>
<td>1460</td>
</tr>
<tr>
<td>12</td>
<td>3906</td>
<td>1234.23</td>
<td>1443</td>
<td>1443</td>
</tr>
<tr>
<td>13</td>
<td>1012</td>
<td>1172.51</td>
<td>1326</td>
<td>1326</td>
</tr>
<tr>
<td>14</td>
<td>82</td>
<td>1286.23</td>
<td>1472</td>
<td>1474</td>
</tr>
<tr>
<td>15</td>
<td>320</td>
<td>1260.54</td>
<td>1409</td>
<td>1514</td>
</tr>
</tbody>
</table>

Table 4: Performance of Algo 1 for the 29 nodes-35 arcs graph

<table>
<thead>
<tr>
<th>arc id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>arc weight</td>
<td>10</td>
<td>5</td>
<td>24</td>
<td>4</td>
<td>15</td>
<td>26</td>
<td>1</td>
<td>7</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>arc id</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>arc weight</td>
<td>26</td>
<td>34</td>
<td>5</td>
<td>9</td>
<td>18</td>
<td>5</td>
<td>42</td>
<td>7</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>arc id</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>arc weight</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>48</td>
<td>47</td>
<td>55</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td>arc id</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arc weight</td>
<td>17</td>
<td>9</td>
<td>30</td>
<td>75</td>
<td>60</td>
<td>44</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Arc weights for the 29 nodes-37 arcs graph
Unfortunately there is not a polynomial algorithm able to delete the smallest number of arcs in a graph in such a way to obtain a series-parallel graph, even if we can recognize series-parallel graphs in polynomial time (see [22]). A possible heuristic to improve the algorithm can therefore start from a spanning rooted tree of $G$ and then add to $G$ as many arcs as possible by preserving the property of being series-parallel. A further improvement to the algorithm can be obtained by using the results of Sidney [21]. In fact, in [21] a decomposition approach is shown to provide the optimal solution of MSTS for any acyclic graph $G$. This decomposition approach allows to reduce the original problem into several smaller problems which can be solved, hopefully, with a smaller computational effort than the original one.

References


