European Journal of Operational Research 244 (2015) 766–777

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Improved integer linear programming formulations for the job Sequencing and tool Switching Problem

Daniele Catanzaro[a,b,*], Luis Gouveia[c,1], Martine Labbé[d,2]

[a] Louvain School of Management, Université Catholique de Louvain, Chausse de Binche 151, bte M1.01.01, 7000 Mons, Belgium
[b] Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain, Voie du Roman Pays 34, L1.03.01, B-1348 Louvain-la-Neuve, Belgium
[c] Department of Statistics and Operations Research, Centro de Investigação Operacional, Universidade de Lisboa, Campo Grande, Bloco C6, 1749-016 Lisboa, Portugal
[d] Graphs and Mathematical Optimization Unit, Computer Science Department, Université Libre de Bruxelles, Boulevard du Triomphe, CP 210/01, B-1050 Brussels, Belgium

ARTICLE INFO

Article history:
Received 7 April 2014
Accepted 10 February 2015
Available online 15 February 2015

Keywords:
Job sequencing
Tool switching
Traveling salesman problem
Combinatorial optimization
Integer programming

ABSTRACT

In this article we investigate the *job Sequencing and tool Switching Problem* (SSP), a $\mathcal{NP}$-hard combinatorial optimization problem arising from computer and manufacturing systems. Starting from the results described in Tang and Denardo (1987), Crama et al. (1994) and Laporte et al. (2004), we develop new integer linear programming formulations for the problem that are provably better than the alternative ones currently described in the literature. Computational experiments show that the lower bounds obtained by the linear relaxation of the considered formulations improve, on average, upon those currently described in the literature and suggest, at the same time, new directions for the development of future exact solution approaches.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

In manufacturing systems it often happens that a set of jobs, each one requiring a specific set of tools, has to be sequentially processed in a flexible machine with a capacitated tool magazine (Crama, 1997; Crama, Moonen, Spieksma, & Talloen, 2007; Tang & Denardo, 1987). The capacity of the magazine is usually suited to store the tools required by any job, but it is usually insufficient to store, at the same time, all of the involved tools. Hence, operations of *tools switching* (i.e., the removal of a tool from the magazine and the insertion of a new one in its place) may be necessary to sequentially process the considered set. This situation often occurs e.g., in metal working industries, where automatic machines are used to manufacture parts. Each machine performs operations by using the tools placed in its limited capacity tool magazine and requires tool switching operations when completing a job and moving to the next one in the sequence (Tang & Denardo, 1987). This situation also occurs in other contexts, such as computer systems. Specifically, a modern computer system has k pages of fast memory and $n - k$ pages of slow memory. The system usually has to satisfy a sequence of requests to pages in their order of occurrence. A request can be addressed only if the required page is in the fast memory. If the request cannot be addressed, a page fault occurs and a page in the fast memory must be selected and moved to the slow memory in order to make room for the requested page (McGeoch & Sleator, 1994).

Tools switching operations can be performed either in groups or one at a time, depending on the specific application. However, they are generally time consuming and/or expensive, hence it is usually convenient to process the jobs in such a way that the overall number of tool switches is minimized.

The switching cost may be tool and/or application dependent as shown in Tang and Denardo (1987), Crama (1997), Crama et al. (2007), Crama, Kolen, Oerlemans, and Spieksma (1994), Balakrishnan and Chakravarty (2001), Belady (1966), Blazewicz and Finke (1994), McGeoch and Sleator (1994), Privault and Finke (2000). In this article we just focus on the simplest (although the main) version of the switching cost, characterized by being uniform and tool independent, and we investigate the problem of determining the job processing sequence inducing the minimum overall switching cost. Such a problem is known in the literature as the *job Sequencing and tool Switching*

* Corresponding author at: Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain, Voie du Roman Pays 34, L1.03.01, B-1348 Louvain-la-Neuve, Belgium. Tel.: +32 65 323 313; fax: +32 65 323 279.
E-mail addresses: daniele.catanzaro@uclouvain.be (D. Catanzaro), legouveia@fc.ul.pt (L. Gouveia), mlabbe@ulb.ac.be (M. Labbé).
[1] Tel.: +351 217 500 409; fax: +351 217 500 081.
[2] Tel.: +32 2 650 38 36; fax: +32 2 650 5970.

http://dx.doi.org/10.1016/j.ejor.2015.02.018
0377-2217/© 2015 Elsevier B.V. All rights reserved.

Table 1
An example of an instance of the SSP (Tang & Denardo, 1987).

Jobs	1	2	3	4	5	6	7	8	9	10
Tools	1	1	2	1	3	1	1	6	3	5
	4	3	6	5	5	2				7
	8	5	7	7	8	4				
	9		8	9						
Magazine capacity: 4										

Table 2
An example of a solution to the instance of the SSP shown in Table 1.

Jobs	8	1	6	4	2	5	10	3	9	7
Tools	④	4	4	5	5	5	5	2	3	1
	6	1	1	1	1	⑦	7	7	⑦	⑦
	⑧	8	2	7	3	3	⑥	6	⑥	⑥
	⑨	9	⑨	9	⑧	8	⑧	8	⑧	⑧
Magazine capacity: 4										

Table 3
The instance of SSP can be reduced in size by applying the dominance rule described in Tang and Denardo (1987).

Jobs	1	2	3	4	5	6
Tools	1	1	2	1	3	1
	4	3	6	5	5	2
	8	5	7	7	8	4
	9		8	9		
Magazine capacity: 4						

Problem (SSP) (Crama et al., 1994; Ghiani, Grieco, & Guerriero, 2010; Laporte, Salazar-Gonzáles, & Semet, 2004; Tang & Denardo, 1987). An example of an instance of the SSP is shown in Table 1. In particular, the first row in the table provides the labels of the jobs and the corresponding processing order; the last row provides the capacity of the magazine. Each column in the table refers to a particular job in the considered set and specifies the tools necessary to its processing. The empty entries in each column denote jobs requiring a number of tools smaller than the capacity of the magazine. A solution to an instance of the SSP is obtained by specifying both the sequence in which the jobs have to be processed and the tools that have to be in the magazine when a given job is processed. For example, Table 2 shows a feasible solution to the instance shown in Table 1. In this case, the job processing sequence is $\{8, 1, 6, 4, 2, 5, 10, 3, 9, 7\}$; the circled tools represent tools that are unnecessary to process a specific job, but that may prove useful to decrease the overall number of switches induced by a specific job sequence. The circled tools can be added only if a job requires a number of tools smaller than the capacity of the magazine. The solution induces an overall number of tool switches equal to 15, in particular: 4 tool switches to load the tools required by job 8; 1 tool switch in the transition from job 8 to job 1; 1 tool switch in the transition from job 1 to job 6; 2 tool switches in the transition from job 6 to job 4; 2 tool switches in the transition from job 4 to job 2; and 1 tool switch in each of the remaining transitions. The size (i.e., the number of jobs and tools) of an instance of the SSP depends on the nature of the considered application and can easily range from dozens (e.g., in manufacturing systems) to hundreds (e.g., in computer systems) of jobs and tools.

Although numerous applications of the SSP appeared in the literature since 1966 (see e.g., Balakrishnan & Chakravarty, 2001; Belady, 1966; Blazewicz & Finke, 1994; Crama et al., 1994, 2007; McGeoch & Sleator, 1994; Privault & Finke, 2000), the problem was formally stated only in 1987 by Tang and Denardo (1987). The authors first observed that the optimal solution to an instance of the SSP does not change if one removes from the instance a job whose tool set is a subset of the tool set required by another job. The authors called this property *the dominance rule*. For example, job 10 in Table 1 requires tools 5 and 7 i.e., a subset of the tool set required by job 4. Hence, job 10 can be removed from the instance without affecting the corresponding optimal solution. If we apply the dominance rule also to jobs 7, 8 and 9, we obtain a new instance of the SSP (see Table 3) characterized by a smaller size. Unless not explicitly stated, in this article we will always assume to work with instances of the SSP containing only jobs requiring non-dominated tool sets.

Tang and Denardo (1987) also investigated a number of practical aspects of the problem and first pointed out the relationship between the SSP and the *Traveling Salesman Problem* (TSP) (see Garey & Johnson, 2003). In particular, the authors observed that an instance of the SSP degenerates into an instance of the TSP when (i) the tool set of each job has cardinality constant and equal to the capacity of the magazine and (ii) the switching cost is uniform and tool independent. This observation constitutes the earliest proof of NP-hardness of the SSP; a more formal and thorough analysis of the complexity of the SSP was performed seven years later by Crama et al. (1994).

Interestingly, Tang and Denardo showed that the SSP can be decomposed into two nested subproblems called:

1. *the Sequencing Problem (SP)*, consisting of identifying an (optimal) job processing sequence;
2. *the Tooling Problem (TP)*, consisting of determining, for a given job processing sequence, what tools should be loaded in the tool magazine at each position of the sequence, in order to minimize the total number of tool switches.

The authors showed that the TP can be solved in polynomial time via a greedy algorithm called *Keep Tool Needed Soonest* (KTNS); in contrast, the SP constitutes the hardest part of the problem. Tang and Denardo developed an integer linear programming formulation to exactly solve the problem and a constructive heuristic to approximate its optimal solution. However, the performances of both approaches proved to be rather disappointing in practice (Laporte et al., 2004). Hence, numerous research efforts focused on developing solution approaches aiming at solving, at least heuristically, instances of the SSP, see e.g., Crama (1997), Crama et al. (2007), Crama et al. (1994), Askin, Sodhi, and Sen (1994), Avci and Akturk (1996), Bard (1988), Gray, Seidmann, and Stecke (1993), Hertz and Widmer (1993), Hertz, Laporte, Mittaz, and Stecke (1998). Surprisingly enough, however, in the last two decades only a limited number of exact solution approaches have been proposed in the literature, namely Laporte et al. (2004) and Ghiani et al. (2010).

Laporte et al. (2004) proposed an Integer Linear Programming (ILP) formulation and a branch-and-bound algorithm to exactly solve the SSP. The authors showed that the proposed ILP formulation was tighter than Tang and Denardo's but, at the same time, unable to solve instances containing more than 10 jobs within 1 hour of computing time. In contrast, the branch-and-bound algorithm and the corresponding combinatorial lower bounds proved to be more successful, allowing the solution of instances of the SSP containing up to 25 jobs within the same runtime limit. Ghiani et al. (2010) proposed a different exact solution approach for the SSP based on modeling the problem as a nonlinear TSP. The authors improved the KTNS greedy algorithm for the TP and developed a branch-and-cut algorithm for the SSP exploiting a number of symmetry breaking techniques, dominance rules, and specific lower bounds. Their computational experiments showed that the proposed algorithm outperforms the performance of Laporte et al., being able to solve instances of the SSP containing up to 45 jobs within 1 hour of computing time.

The weak lower bounds provided by the linear programming relaxations of the ILP formulations currently described in the literature on the SSP possibly are one of the main reasons for explaining the limited number of exact solution approaches for the problem. In this article, we address this major limitation and investigate new ILP formulations for the SSP able to provide tighter lower bounds to the problem. Starting from the results discussed in Tang and Denardo (1987), Crama (1997), Crama et al. (1994), Laporte et al. (2004), Ghiani et al. (2010), in Section 2 we briefly review the main ILP formulations for the SSP currently described in the literature. In Section 3 we present new ILP formulations for the SSP that have a Linear Programming (LP) relaxation that is provably tighter than the LP relaxation of the previous ones described in the literature. We investigate the combinatorial interpretation of the proposed formulations and we propose valid inequalities to further improve the lower bounds provided by their linear programming relaxations. In Sections 4.4 and 4.5 we present computational results obtained by running the proposed formulations on a number of benchmark instances for the SSP and discuss the performances of each formulation. The theoretical findings discussed in this article may provide new insights about the combinatorial aspects of the problem and may suggest, at the same time, new directions for the development of future exact solution approaches for the problem.

Before proceeding, in the next section we introduce some notation that will prove useful throughout the article.

2. Notation and previous integer linear programming formulations

Consider a set $J = \{1, 2, \dots, n\}$ of n jobs to process in the flexible machine and let $T = \{1, 2, \dots, m\}$ be the set of m tools necessary to process the jobs in J. Denote T_j as the set of tools necessary to process job $j \in J$; J_t as the set of jobs using tool $t \in T$; ${}^cJ_t$ as the complement of J_t; and $C \geq \max_j\{|T_j|\}$ as the *capacity of the magazine*, i.e., the maximum number of tools that can be stored in the magazine of the machine at any time. A *job processing sequence s*, is a sequence of n naturals representing the processing order for the jobs. For example, by referring to the example in Table 1, $s = \{1, 2, 3, 4, 5, 6\}$ represents a job processing sequence having job 1 as first job to process, job 2 as second job to process, and so on. Tang and Denardo proposed the following formulation for the SSP based on the assignment of jobs to the positions of a processing sequence (Tang & Denardo, 1987).

2.1. Tang and Denardo's formulation

Denote $K = \{1, 2, \dots, n\}$ as the set of n positions in the job sequence to be determined. Let u_j^k be a decision variable equal to 1 if job j is processed in position k and 0 otherwise. Similarly, let v_t^k be a decision variable equal to 1 if tool t is present when processing a job in position k and 0 otherwise. Finally, let w_t^k be a decision variable equal to 1 if tool t is present when processing job k and not when processing job $k-1$, and 0 otherwise. Then, a valid formulation for the problem is the following:

Formulation 1. *(Tang & Denardo, 1987),*

$$\min \sum_{t\in T} v_t^1 + \sum_{t\in T}\sum_{k\in K\setminus\{1\}} w_t^k \tag{1a}$$

$$\text{s.t.} \sum_{j\in J} u_j^k = 1 \quad \forall\, k \in K \tag{1b}$$

$$\sum_{k\in K} u_j^k = 1 \quad \forall\, j \in J \tag{1c}$$

$$\sum_{j\in J_t} u_j^k \leq v_t^k \quad \forall\, k \in K, t \in T \tag{1d}$$

$$\sum_{t\in T} v_t^k \leq C \quad \forall\, k \in K \tag{1e}$$

$$v_t^k - v_t^{k-1} \leq w_t^k \quad \forall\, k \in K \setminus \{1\}, t \in T \tag{1f}$$

$$u_j^k \in \{0, 1\} \quad \forall\, k \in K, j \in J \tag{1g}$$

$$v_t^k, w_t^k \in \{0, 1\} \quad \forall\, k \in K, t \in T. \tag{1h}$$

The objective function (1a) minimizes the total number of tool switches. In particular, the addend $\sum_{t\in T} v_t^1$ takes into account the loading of the magazine when processing the first job in the sequence. Constraints (1b) and (1c) are assignment constraints. Constraints (1d) impose that no job requiring tool t can be processed in position k if such a tool is not present in the magazine at that position. Constraints (1e) guarantee that the magazine capacity is never exceeded. Note that these constraints can be stated as equalities since there is no need to remove unnecessary tools from the magazine (Laporte et al., 2004). Finally, constraints (1f) impose that a tool switch must be counted whenever a tool is present in position k but was not present in position $k-1$.

2.2. Laporte et al.'s formulation

It is worth noting that the solution $u_j^k = 1/n$, for all $j \in J$ and $k \in K$, $v_t^k = |J_t|/n$ for all $k \in K$ and $t \in T$, and $w_t^k = 0$, for all $k \in K$ and $t \in T$, is feasible and optimal for the linear programming relaxation of Formulation 1. To overcome such a drawback, Laporte et al. (2004) proposed an alternative integer linear programming formulation that exploits the analogies between the SP and the TSP. Specifically, the authors introduced a dummy job 0 to represent the start and stop operations and set by convention $T_0 = \emptyset$. Subsequently, they modeled the SSP as a particular version of the TSP in which the optimal hamiltonian circuit has to (i) start and end at vertex (job) 0, (ii) visit all the remaining vertices (jobs) exactly once and (iii) have as a weight on the edge (i, j) of the circuit the corresponding number of tool switches relative to the transition from job i to job j. In order to describe in detail the proposed formulation, denote $J_0 = J \cup \{0\}$, $J^i = J \setminus \{i\}$, for all $i \in J$, and $J_0^i = J_0 \setminus \{i\}$, for all $i \in J$. Moreover, denote G as the complete graph having as vertexset J_0.

Let x_{ij} be a decision variable equal to 1 if job j follows immediately job i in the sequence and 0 otherwise, for all $i, j \in J_0$, $i \neq j$. Moreover, let y_j^t be a decision variable equal to 1 if tool $t \in T$ is in the magazine when job j is processed and 0 otherwise, for all $j \in J_0$, $t \in T$. Similarly, let z_j^t be a decision variable equal to 1 if tool $t \in T$ is inserted when processing job j and 0 otherwise, for all $j \in J_0$, $t \in T$. Then, Laporte et al.'s formulation can be stated as follows:

Formulation 2. *(Laporte et al., 2004),*

$$\min \sum_{j\in J}\sum_{t\in T_j} z_j^t \tag{2a}$$

$$\text{s.t.} \sum_{j\in J_0^i} x_{ij} = 1 \quad \forall\, i \in J_0 \tag{2b}$$

$$\sum_{i\in J_0^j} x_{ij} = 1 \quad \forall\, j \in J_0 \tag{2c}$$

$$\sum_{\substack{i,j\in S:\\ i\neq j}} x_{ij} \leq |S| - 1 \quad \forall\, S \subset J_0, 2 \leq |S| \leq n-1 \tag{2d}$$

$$\sum_{t\in T} y_j^t \leq C \quad \forall\, j \in J \tag{2e}$$

$$x_{ij} + y_j^t - y_i^t \leq z_j^t + 1 \quad \forall\, i, j \in J, i \neq j, t \in T \tag{2f}$$

$$x_{0j} + y_j^t \leq z_j^t + 1 \quad \forall\, j \in J, t \in T \tag{2g}$$

$$y_j^t = 1 \quad \forall\, j \in J, t \in T_j \tag{2h}$$

$$z_j^t = 0 \quad \forall\, j \in J, t \in T \setminus T_j \tag{2i}$$

$$y_j^t, z_j^t \in \{0,1\} \quad \forall\, j \in J, t \in T \tag{2j}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i,j \in J_0. \tag{2k}$$

The objective function (2a) minimizes the overall number of tool switches. Constraints (2b) and (2c) are assignment constraints. Constraints (2d) prevent the arising of subtours. Constraints (2e) are capacity constraints. Constraints (2f) prevent that job j follows job i in the optimal sequence if, at the same time, job j requires tool t and tool t is neither carried from job i nor inserted for job j. Constraints (2g) are a special case of constraints (2f) and account for the loading of the magazine relative to the transition from the dummy job 0 to the first job in the optimal job processing sequence. Finally, constraints (2h)–(2i) ensure that a tool switch is only performed when the tool is required to immediately execute a job.

As shown in Laporte et al. (2004), the linear programming relaxation of Formulation 2 dominates the relaxation of Tang and Denardo's model. However, computational experiments showed that the lower bounds provided by Formulation 2 are still generally poor (Laporte et al., 2004). This fact in turn prevents Formulation 2 from solving instances of the SSP containing more than 10 jobs within 1 hour of computing time (Laporte et al., 2004). In order to improve the lower bounds obtained by linear programming, in the next section we shall present a number of alternative ILP formulations for the SSP whose LP relaxations are provably tighter than the LP relaxations of Formulation 2.

3. New integer linear programming formulations for the SSP

3.1. A tighter formulation

Consider a decision variable y_{ij}^t equal to 1 if in the processing sequence a tool $t \in T$ is carried in the transition from job i to job j, $i,j \in J_0$, $i \neq j$, and 0 otherwise. Similarly, let z_{ij}^t be a decision variable equal to 1 if in the processing sequence a tool $t \in T$ has to be added in the transition from job i to job j, $i,j \in J_0$, $i \neq j$, and 0 otherwise. Then, a valid formulation for the SSP is the following:

Formulation 3.

$$\min \sum_{\substack{i,j \in J_0:\\ i \neq j}} \sum_{t \in T_j} z_{ij}^t \tag{3a}$$

$$\text{s.t.} \sum_{j \in J_0^i} x_{ij} = 1 \quad \forall\, i \in J_0 \tag{3b}$$

$$\sum_{i \in J_0^j} x_{ij} = 1 \quad \forall\, j \in J_0 \tag{3c}$$

$$\sum_{\substack{i,j \in S:\\ i \neq j}} x_{ij} \le |S| - 1 \quad \forall\, S \subset J_0, 2 \le |S| \le n-1 \tag{3d}$$

$$\sum_{i \in J_0^j} \left(y_{ij}^t + z_{ij}^t\right) = 1 \quad \forall\, j \in J_0, t \in T_j \tag{3e}$$

$$y_{ij}^t + z_{ij}^t \le x_{ij} \quad \forall\, i,j \in J_0 : i \neq j, t \in T : t \notin T_i, t \in T_j \tag{3f}$$

$$\sum_{t \in T} \left(y_{ij}^t + z_{ij}^t\right) \le C x_{ij} \quad \forall\, i,j \in J_0 : i \neq j \tag{3g}$$

$$\sum_{k \in J_0^i} (y_{ki}^t + z_{ki}^t) \ge \sum_{j \in J_0^i} y_{ij}^t \quad \forall\, i \in J_0,\ t \in T \tag{3h}$$

$$y_{ij}^t \ge x_{ij} \quad \forall\, i,j \in J_0 : i \neq j, t \in T_i \cap T_j \tag{3i}$$

$$z_{ij}^t = 0 \quad \forall\, i,j \in J_0 : i \neq j, t \in T \setminus T_j \tag{3j}$$

$$y_{0j}^t = 0 \quad \forall\, j \in J, t \in T \tag{3k}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i,j \in J_0 : i \neq \tag{3l}$$

$$y_{ij}^t, z_{ij}^t \in \{0,1\} \quad \forall\, i,j \in J_0 : i \neq j, t \in T. \tag{3m}$$

Constraints (3b)–(3d) are equivalent to constraints (2b)–(2d) in Formulation 2. Constraints (3e) impose that in the transition from job i to job j a given tool $t \in T_j$ is either carried from job i or it is added in the transition. Constraints (3f) impose that a tool $t \in T_j$ can be either carried from a previous job or added in the transition, only if such a transition is in the optimal job processing sequence. Constraints (3g) are similar to the capacity constraints (2e) in Formulation 2. Constraints (3h) impose that if a tool $t \in T$ is carried in the transition from job i to any job j then it must be either carried or added in the transition from another job k to job i, $k \neq i$. Constraints (3i) impose to carry all of the tools in $T_i \cap T_j$ when considering the transition from job i to job j. Constraints (3j) are analogous to constraints (2i) in Formulation 2. Finally, constraints (3k) account for the loading of the magazine relative to the transition from the dummy job 0 to the first job in the optimal job processing sequence.

Variables y_j^t and z_j^t in Formulation 2 and variables y_{ij}^t and z_{ij}^t in Formulation 3 are related by the following equations:

$$y_j^t = \sum_{i \in J_0^j} \left(y_{ij}^t + z_{ij}^t\right) \quad \forall\, j \in J, t \in T \tag{4}$$

$$z_j^t = \sum_{i \in J_0^j} z_{ij}^t \quad \forall\, j \in J, t \in T. \tag{5}$$

These relationships can be used to show that Formulation 3 is tighter than Formulation 2. To this end, we add (4), (5) and (2j) to Formulation 3. Note that this addition does not alter the LP relaxation of Formulation 3 since the new constraints only define the old variables y_j^t and z_j^t. After adding these constrains, (3e) and (2h) become equivalent. Similarly, the surrogates of constraints (3g) and (3j) with respect to index i become equivalent to constraints (2e) and (2i), respectively. Now, if we combine Eqs. (4) and (5) together with (3c), (3f) and (3h) then we obtain an inequality that dominates constraints (2f) in Formulation 2. Specifically, define $J_0^{ij} = J_0 \setminus \{i,j\}$ and add up constraints (3f) for all jobs in J_0^{ij} by using constraints (3c). Then, it holds that

$$\sum_{k \in J_0^{ij}} \left(y_{kj}^t + z_{kj}^t\right) \le \sum_{k \in J_0^{ij}} x_{kj} = 1 - x_{ij}. \tag{6}$$

By adding $y_{ij}^t + z_{ij}^t$ to both sides of (6) we get

$$\sum_{k \in J_0^j} \left(y_{kj}^t + z_{kj}^t\right) \le y_{ij}^t + z_{ij}^t + x_{ij}. \tag{7}$$

It is worth noting that constraint (3h) can be rewritten as

$$\sum_{k \in J_0^i} (y_{ki}^t + z_{ki}^t) \sum_{k \in J_0^{ij}} y_{ik}^t \ge y_{ij}^t. \tag{8}$$

Then, it holds that

$$\sum_{k \in J_0^j} \left(y_{kj}^t + z_{kj}^t\right) \le y_{ij}^t + z_{ij}^t + 1x_{ij} \le \sum_{k \in J_0^i} (y_{ki}^t + z_{ki}^t) \sum_{k \in J_0^{ij}} y_{ik}^t + z_{ij}^t + x_{ij}. \tag{9}$$

Now, note that: (i) the first term in the right-hand-side of (9) is equal to y_i^t; (ii) the second term is less than or equal to 0;and (iii) the third term is less than or equal to z^t_j. Hence, we have

$$y^t_j = \sum_{k\in J^j_0}\left(y^t_{kj}+z^t_{kj}\right) \le y^t_{ij}+z^t_{ij}+x_{ij} \le \sum_{k\in J^i_0}(y^t_{ki}+z^t_{ki}) \times \sum_{k\in J^{ij}_0} y^t_{ik}+z^t_{ij}+1x_{ij} \le y^t_i+z^t_j+x_{ij}, \tag{10}$$

which implies that

Proposition 1. *The objective function value of the linear programming relaxation of Formulation 3 is greater than or equal to the linear programming relaxation of Formulation 2.*

3.2. Reducing formulation size

By just exploiting the definitions of variables y^t_{ij} and z^t_{ij}, it is possible to replace constraints (3f) by the following three new sets of constraints:

$$y^t_{ij}+z^t_{ij}=x_{ij} \quad \forall\, i,j\in J_0 : i\ne j, t\in T : t\notin T_i, t\in T_j \tag{11a}$$

$$y^t_{ij}=x_{ij} \quad \forall\, i,j\in J_0 : i\ne j, t\in T : t\in T_i\cap T_j \tag{11b}$$

$$y^t_{ij}\le x_{ij} \quad \forall\, i,j\in J_0 : i\ne j, t\in T : t\notin T_j. \tag{11c}$$

Constraints (11a) force a tool $t\in T$ such that $t\notin T_i$ and $t\in T_j$ to be either carried from job i or added in job j if the transition from job i to job j is actually considered in the optimal job processing sequence. Constraints (11b) force a tool $t\in T$ such that $t\in T_i\cap T_j$ to be carried from job i if the transition from job i to job j is considered in the optimal job processing sequence. Finally, constraints (11c) allow a tool $t\in T$ such that $t\notin T_j$ to be carried from job i if the transition from job i to job j is considered in the optimal job processing sequence. Interestingly, these new sets of constraints, in particular the equality constraints, can be used to reduce the size of Formulation 3. Specifically, we can use constraints (11a) to eliminate all of variables z^t_{ij} and constraints (11b) to eliminate variables y^t_{ij} when $t\in T_i\cap T_j$. In this way, we have to consider only variables y^t_{ij} defined for all $i,j\in J_0$ such that $i\ne j$ and for all $t\in T$ such that either $t\notin T_i$ or $t\in T_i$ and $t\notin T_j$. In the light of this observation, constraints (3e) become redundant. In fact, for all $j\in J_0$ and $t\in T_j$, it holds that

$$1=\sum_{i\in J^j_0}\left(y^t_{ij}+z^t_{ij}\right)=\sum_{i\in J^j_0}y^t_{ij}+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}z^t_{ij} =\sum_{i\in J^j_0}y^t_{ij}+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}\left(x_{ij}-y^t_{ij}\right),$$

which is equivalent to

$$\sum_{i\in J^j_0}y^t_{ij}+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}x_{ij}=1+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}y^t_{ij},$$

which can be rewritten as

$$\sum_{\substack{i\in J^j_0:\\ t\in T_i}}y^t_{ij}+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}x_{ij}=1,$$

which, in turn, due to (11b), becomes

$$\sum_{\substack{i\in J^j_0:\\ t\in T_i}}x_{ij}+\sum_{\substack{i\in J^j_0:\\ t\notin T_i}}x_{ij}=1.$$

Similarly, for all $i,j\in J_0$, $i\ne j$, constraints (3g) can be rewritten as

$$\sum_{t\in T}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\in T_j}}z^t_{ij}\le Cx_{ij},$$

which is equivalent to

$$\sum_{t\in T}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\in T_j}}\left(x_{ij}-y^t_{ij}\right)\le Cx_{ij},$$

which can be rewritten as

$$\sum_{\substack{t\in T:\\ t\in T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\in T_j}}x_{ij}\le Cx_{ij}. \tag{12}$$

The first term of (12) can in turn be expanded as

$$\sum_{\substack{t\in T:\\ t\in T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\in T_i, t\in T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\in T_j}}x_{ij}\le Cx_{ij}.$$

By using (11b) we get

$$\sum_{\substack{t\in T:\\ t\in T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\in T_i, t\in T_j}}x^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\in T_j}}x_{ij}\le Cx_{ij},$$

which is equivalent to

$$\sum_{\substack{t\in T:\\ t\in T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\in T_j}}x^t_{ij}\le Cx_{ij},$$

which leads to the constraint

$$\sum_{\substack{t\in T:\\ t\in T_i, t\notin T_j}}y^t_{ij}+\sum_{\substack{t\in T:\\ t\notin T_i, t\notin T_j}}y^t_{ij}\le (C-|T_j|)x_{ij}.$$

Finally, consider constraints (3h) and first assume that $t\in T_i$. Then, it holds that

$$\sum_{k\in J^i_0}y^t_{ki}+\sum_{\substack{k\in J^i_0:\\ t\notin T_k}}z^t_{ki}\ge\sum_{j\in J^i_0}y^t_{ij}.$$

By using (11a) we get

$$\sum_{k\in J^i_0}y^t_{ki}+\sum_{\substack{k\in J^i_0:\\ t\notin T_k}}(x_{ij}-y^t_{ki})\ge\sum_{j\in J^i_0}y^t_{ij},$$

which, in turn, can be rewritten as

$$\sum_{\substack{k\in J^i_0:\\ t\in T_k}}y^t_{ki}+\sum_{\substack{k\in J^i_0:\\ t\notin T_k}}x_{ij}\ge\sum_{j\in J^i_0}y^t_{ij}.$$

By using (11b) and (3b) we obtain

$$1=\sum_{\substack{k\in J^i_0:\\ t\in T_k}}x_{ki}+\sum_{\substack{k\in J^i_0:\\ t\notin T_k}}x_{ij}\ge\sum_{j\in J^i_0}y^t_{ij},$$

which, in turn, implies that the constraint is redundant. In contrast, if we assume that $t\notin T_i$, then variables z^t_{ij} are not defined and the constraint reduces to

$$\sum_{k\in J^i_0}y^t_{ki}\ge\sum_{j\in J^i_0}y^t_{ij}.$$

Thus, Formulation 3 reduces to

Formulation 4.

$$\min\sum_{j\in J}|T_j|x_{0j}+\sum_{\substack{i,j\in J:\\ i\ne j}}\sum_{\substack{t\in T:\\ t\notin T_i\\ t\in T_j}}\left(x_{ij}-y^t_{ij}\right) \tag{13a}$$

$$\text{s.t.}\sum_{j\in J^i_0}x_{ij}=1 \quad \forall\, i\in J_0 \tag{13b}$$

$$\sum_{i\in J^j_0}x_{ij}=1 \quad \forall\, j\in J_0 \tag{13c}$$

$$\sum_{\substack{i,j\in S:\\ i\neq j}} x_{ij} \le |S|-1 \quad \forall\, S\subset J_0,\, 2\le |S|\le n-1 \tag{13d}$$

$$\sum_{k\in J_0^i} y^t_{ki} - \sum_{j\in J_0^i} y^t_{ij} \ge 0 \quad \forall\, i\in J_0,\, t\in T\setminus T_i \tag{13e}$$

$$\sum_{\substack{t\in T:\\ t\in T_i\\ t\notin T_j}} y^t_{ij} + \sum_{\substack{t\in T:\\ t\notin T_i\\ t\notin T_j}} y^t_{ij} \le (C-|T_j|)x_{ij} \quad \forall\, i,j\in J_0 : i\neq j \tag{13f}$$

$$y^t_{ij} \le x_{ij} \quad \forall\, i,j\in J_0 : i\neq j,\, t\in T : t\notin T_j \tag{13g}$$

$$x_{ij}\in\{0,1\} \quad \forall\, i,j\in J_0 : i\neq j \tag{13h}$$

$$y^t_{ij}\in\{0,1\} \qquad \forall\, i,j\in J_0 : i\neq j,\, t\in T : t\notin T_i \\ \text{or } (t\in T_i \text{ and } t\notin T_j)\,. \tag{13i}$$

In the light of these results, the following proposition holds:

Proposition 2. *The linear programming relaxations of Formulations 3 and 4 have the same optimal solution value.*

The following proposition holds for Formulation 4:

Proposition 3. *Assume that variables x_{ij} are fixed and determine a hamiltonian circuit in G. Then, constraints (13e)–(13g) determine a matrix that is totally unimodular.*

Proof. Consider the capacity constraints (13f) and introduce the slack variable q_{ij}, for all $i, j \in J_0$, $i \neq j$. Then we have

$$\sum_{\substack{t\in T:\\ t\in T_i\\ t\notin T_j}} y^t_{ij} + \sum_{\substack{t\in T:\\ t\notin T_i\\ t\notin T_j}} y^t_{ij} + q_{ij} = (C-|T_j|)x_{ij} \qquad \forall\, i,j\in J_0 : i\neq j. \tag{14}$$

Without loss of generality, assume both that $x_{i,i+1} = 1$, for all $i = 0, \dots, n-1$, and $x_{n0} = 1$. Then, it holds that $y^t_{ij} = 0$, for all $j \in J_0$ such that $j \neq i+1$. Hence, the system reduces to

$$y^t_{j-1,j} - y^t_{j,j+1} \ge 0 \qquad \forall\, j\in J_0\;\, t\in T\setminus T_j \tag{15}$$

$$\sum_{\substack{t\in T:\\ t\in T_{j-1}\\ t\notin T_j}} y^t_{j-1,j} + \sum_{\substack{t\in T:\\ t\notin T_{j-1}\\ t\notin T_j}} y^t_{j-1,j} + q_{j-1,j} = (C-|T_j|) \qquad \forall\, j\in J_0 \tag{16}$$

$$y^t_{j,j+1} \le 1 \qquad \forall\, j\in J_0,\, t\in T : t\notin T_j. \tag{17}$$

Consider constraints (16) for a fixed $j \in J_0$ and subtract the same constraint for $j+1$. Then it holds that

$$\sum_{\substack{t\in T:\\ t\in T_{j-1}\\ t\notin T_j}} y^t_{j-1,j} + \sum_{\substack{t\in T:\\ t\notin T_{j-1}\\ t\notin T_j}} y^t_{j-1,j} + q_{j-1,j} - \sum_{\substack{t\in T:\\ t\in T_j\\ t\notin T_{j+1}}} y^t_{j,j+1} \\ - \sum_{\substack{t\in T:\\ t\notin T_j\\ t\notin T_{j+1}}} y^t_{j,j+1} - q_{j,j+1} = |T_j| - |T_{j+1}|. \tag{18}$$

Now observe that the system constituted by (15), (17) and (18) is totally unimodular because each variable appears exactly twice with coefficients both +1 and −1. □

3.3. Combinatorial interpretation of Formulation 4

The decomposition of the SSP in two nested subproblems (namely, the Sequencing Problem (SP) and the Tooling Problem (TP)) described in Tang and Denardo (1987) proves useful to provide a combinatorial interpretation of Formulation 4. Before proceeding, we briefly review Crama et al.'s results and insights (Crama et al., 1994) and introduce some notations that will prove useful in the remainder of the article.

Table 4
An example of P matrix associated to the example shown in Table 1.

	Jobs					
Tools	1	2	3	4	5	6
1	1	1	0	1	0	1
2	0	0	1	0	0	1
3	0	1	0	0	1	0
4	1	0	0	0	0	1
5	0	1	0	1	1	0
6	0	0	1	0	0	0
7	0	0	1	1	0	0
8	1	0	1	0	1	0
9	1	0	0	1	0	0

A *tooling policy* for a given job processing sequence s is any determination of the tools that have to be loaded in the tool magazine at each position of s. A tooling policy is *feasible* if, at each position of s, the capacity constraint is respected, i.e., if the sum of the tools present in the magazine at any position of s is smaller than or equal to C. A feasible tooling policy is *optimal* if it minimizes the overall tool switches in s.

Let a job processing sequence s be fixed and consider a $n \times m$ boolean matrix P whose generic entry p_{ij} is equal to 1 if tool i is required for the jth job in s and 0 otherwise. For example, Table 4 shows the P matrix corresponding to the example in Table 1.

A *0-block* of P is a maximal subset of consecutive zeros in a row of P or, more formally, a set of the form $\{(i,j), (i,j+1), \dots, (i,j+k)\}$ for which the following conditions hold:

1. $1 < j \le j+k < n$;
2. $p_{ij} = p_{i,j+1} = \dots = p_{i,j+k} = 0$;
3. $p_{i,j-1} = p_{i,j+k+1} = 1$.

Thus, a 0-block denotes the maximal interval before and after which a tool $t \in T$ is needed, but during which it is not needed. This fact suggests, as a possible approach to solve the SSP, to find a job processing sequence inducing the minimum number of 0-blocks during which each tool $t \in T$ is not kept. This is precisely what Formulation 4 does. In fact, it is worth noting that, for each t, Formulation 4 determines a sequence of jobs $i_1, i_2, \dots, i_p$, that are defined by the variables y^t_{ij} being equal to one, more precisely $y^t_{i_1 i_2}, y^t_{i_2 i_3}, \dots, y^t_{i_{p-1} i_p}$. Such a sequence can be seen as a path starting at job i_1 and ending at job i_p. This path is *maximal* if it cannot be augmented, i.e., if the variables y associated with all of the arcs incoming into job (vertex) i_1 and all of the arcs leaving job (vertex) i_p are equal to zero. It is easy to see that these maximal paths correspond to Crama et al.'s 0-blocks.

3.4. Strengthening inequalities for Formulation 4

In this section we provide valid inequalities to strengthen Formulation 4.

Proposition 4. *The inequality*

$$\sum_{i\in J\setminus\{j,k\}} y^t_{ij} \ge y^t_{jk} \quad \forall\, k,j\in J : k\neq j, \qquad t\in T_k\setminus T_j \tag{19}$$

is valid for Formulation 4.

Proof. In a feasible solution to the problem, variable y^t_{jk} can assume values in $\{0, 1\}$. If $y^t_{jk} = 0$ the inequality is trivially valid. If $y^t_{jk} = 1$ then the pair (j, k) belongs to a 0-block during which tool t is kept (see Fig. 1), hence there exists at least one $\hat{i} \in J \setminus \{j, k\}$ such that $y^t_{\hat{i}j} = 1$. Thus, the inequality is again valid. □

We refer to inequalities (19) as the *1-arc inequalities*. It is worth noting that, with arguments similar to those used in Proposition 4, the 1-arc

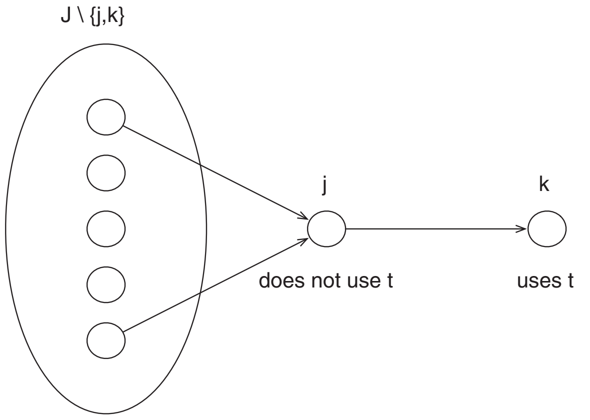

Fig. 1. An example of situation in which the pair (j, k) belongs to a 0-block during which tool t is kept.

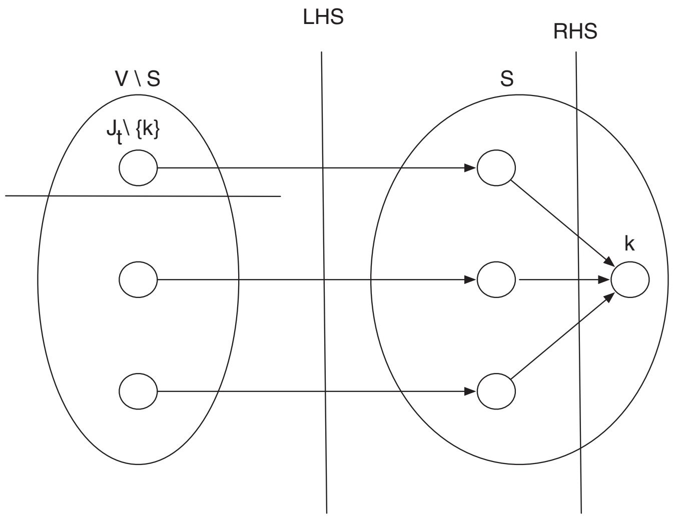

Fig. 2. An example showing how the 1-arc inequalities can be generalized.

inequalities can be generalized as in Fig. 2. Specifically, let $S_1, S_2 \subset J$ and denote $y^t(S_1, S_2) = \sum_{i\in S_1}\sum_{j\in S_2} y^t_{ij}$. Then the following proposition holds:

Proposition 5. *The inequality*

$$y^t(J \setminus S, S \setminus \{k\}) \geq \sum_{i\in S\setminus\{k\}} y^t_{ik} \quad \forall\, S \subset J, k \in S,\ t \in T_k : J_t \cap S = \{k\} \tag{20}$$

is valid for Formulation 4.

We refer to inequalities (20) as the *cut inequalities.*

3.5. An arc-flow integer linear programming formulation for the SSP

The structural properties of the tooling subproblem, and in particular the concept of a 0-block, suggest an alternative network-based integer linear programming formulation for the SSP that can be viewed as a disaggregation of Formulation 4. Specifically, by definition a 0-block with respect to a tool t is a sequence of jobs starting at a job i ending at a job j and such that no job in between i and j (but i and j themselves) needs tool t. Such a sequence can be seen as a path in the induced directed subgraph $G^t_{ij} = (V^t_{ij}, A^t_{ij})$ of G having as vertexset $V^t_{ij} = {}^cJ_t \cup \{i, j\}$, for all distinct $i, j \in J$ such that $t \in T_i \cap T_j$, and as arcset A^t_{ij} the subset of arcs having endpoints in V^t_{ij} except: (i) arc (i, j), (ii) arcs incoming into i, and (iii) arcs diverging from j. By denoting $f^t_{ij,kl}$ as a decision variable equal to 1 if vertex l immediately follows vertex k in a path of G^t_{ij} and 0 otherwise, it is easy to see that the following relationships hold between variables y^t_{ij} in Formulation 4 and variables $f^t_{ij,kl}$:

$$y^t_{kl} = \sum_{i\in J_t} f^t_{il,kl} \qquad \forall\, t \in T,\ \forall\, k \in {}^cJ_t, l \in J_t \tag{21a}$$

$$y^t_{kl} = \sum_{j\in J_t} f^t_{kj,kl} \qquad \forall\, t \in T,\ \forall\, k \in J_t, l \in {}^cJ_t \tag{21b}$$

$$y^t_{kl} = \sum_{\substack{i,j\in J_t:\\ i\neq j}} f^t_{ij,kl} \qquad \forall\, t \in T,\ \forall\, k, l \in {}^cJ_t : k \neq l. \tag{21c}$$

These relationships suggest the following formulation for the SSP.

Formulation 5.

$$\min \sum_{j\in J} |T_j| x_{0j} + \sum_{\substack{i,j\in J:\\ i\neq j\\ t\in T_j\setminus T_i}} \left(x_{ij} - \sum_{\substack{k\in J:\\ k\neq j\\ t\in T_k}} f^t_{kj,ij} \right) \tag{22a}$$

$$s.t. \sum_{j\in J^i_0} x_{ij} = 1 \quad \forall\, i \in J_0 \tag{22b}$$

$$\sum_{i\in J^j_0} x_{ij} = 1 \quad \forall\, j \in J_0 \tag{22c}$$

$$\sum_{\substack{i,j\in S:\\ i\neq j}} x_{ij} \leq |S| - 1 \quad \forall\, S \subset J_0, 2 \leq |S| \leq n-1 \tag{22d}$$

$$\sum_{\substack{l\in V^t_{ij}:\\ l\neq k}} f^t_{ij,kl} - \sum_{\substack{l\in V^t_{ij}:\\ l\neq k}} f^t_{ij,lk} \geq 0 \qquad \forall\, i, j \in J : i \neq j, \quad \forall\, t \in T_i \cap T_j,\ \forall\, k \in {}^cJ_t \tag{22e}$$

$$\sum_{i\in J_t} f^t_{il,kl} \leq x_{kl} \quad \forall\, t \in T,\ \forall\, k \in {}^cJ_t, l \in J_t \tag{22f}$$

$$\sum_{j\in J_t} f^t_{kj,kl} \leq x_{kl} \quad \forall\, t \in T,\ \forall\, k \in J_t, l \in {}^cJ_t \tag{22g}$$

$$\sum_{\substack{i,j\in J_t:\\ i\neq j}} f^t_{ij,kl} \leq x_{kl} \quad \forall\, t \in T,\ \forall\, k, l \in {}^cJ_t : k \neq l \tag{22h}$$

$$\sum_{\substack{t\in T:\\ t\in T_k\\ t\notin T_l}} \sum_{\substack{j\in J_t:\\ j\neq k}} f^t_{kj,kl} + \sum_{\substack{t\in T:\\ t\notin T_k\\ t\notin T_l}} \sum_{\substack{i,j\in J_t:\\ i\neq j}} f^t_{ij,kl} \leq (C - |T_l|) x_{kl} \quad \forall\, k, l \in J : k \neq l \tag{22i}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, i, j \in J_0 : i \neq j \tag{22j}$$

$$f^t_{ij,kl} \geq 0 \qquad \forall\, t \in T, \forall\, i, j \in J : i \neq j, \quad \forall\, k, l \in V^t_{ij} : k \neq l, \quad \{k, l\} \neq \{i, j\}. \tag{22k}$$

Using Eqs. (21a)–(21c) it is easy to see that the objective function (22a) is analogous to (13a) in Formulation 4; constraints (22b)–(22d) correspond to constraints (13b)–(13d) of Formulation 4; constraints (22e) allow the existence of a path (0-block) in V^t_{ij} if such a path has been chosen in the optimal solution to the problem; constraints (22f)–(22h) correspond to constraints (13g) of Formulation 4; and the capacity constraints (22i) correspond to constraints (13f) of Formulation 4. It is easy to see that, if for a fixed triple t, k and l, we add constraint (22e) for all i and j then we obtain constraint (13e) of Formulation 4. Thus, based on the previous observations, the following proposition holds:

Proposition 6. *The objective function value of the linear programming relaxation of Formulation 5 is greater than or equal to the objective function value of the linear programming relaxation of Formulation 4.*

We will see in Section 4 that Formulation 5 usually provides tighter lower bounds than Formulation 4.

4. Computational experiments

In this section we analyze the performances of the formulations described in the previous sections. As the performances of Formulation 1 and Formulation 2 have been already discussed in Laporte et al. (2004), here we focus on the comparison between Formulations 2, 4 and 5. We excluded from the analysis Formulation 3 as it provides the same lower bounds as Formulation 4 and contains more variables and constraints.

Our experiments were motivated by a number of goals, namely: to compare the lower bounds of the formulations; to evaluate, with respect to Formulation 4, the benefits obtained by including the valid inequalities previously described; and to allow the analysis of datasets larger than the ones currently handled by Formulation 2. We do not attempt to compare the runtime performances of approaches that are non-linear or that are not based on integer linear programming, as those analyses have been already performed in Laporte et al. (2004) and Ghiani et al. (2010).

4.1. Datasets

In order to test the performances of Formulations 2, 4 and 5, we considered the datasets described in Laporte et al. (2004), downloadable at http://homepages.ulb.ac.be/~dacatanz/Site/Software_files/JSTSPinstances.zip. Each dataset contains 10 random instances of the SSP, characterized by having the same number of jobs, tools, and magazine capacity. Moreover, each dataset is also characterized by two positive numbers, T^*_m and T^*_M, such that $T^*_m = \min_{j' \in J} |T_{j'}|$ and $T^*_M = \max_{j' \in J} |T_{j'}|$. As described in Laporte *et al.*, a generic instance in a given dataset is created by generating at random, for each job $j \in J$, the set T_j in such a way that T_j is non-dominated and $T^*_m \le |T_j| \le T^*_M$. Datasets belonging to the same group (e.g., datAx, datBx, and so on) differ from one another by the capacity of the magazine. The characteristics of the considered datasets are summarized in Table 5.

4.2. Implementation

We implemented Formulations 2, 4 and 5 in Mosel 3.4.3 of FICO Xpress, Optimizer libraries v25.01.05, running on an IMac Core i7, 3.50 gigahertz, equipped with 16 gigabyte RAM and operating system Mac Os X, Darwin version 10.9.1. We considered three implementations of Formulation 4: one including no strengthening valid inequality (hereafter denoted as Formulation 4); one including the 1-arc inequalities (hereafter denoted as Formulation 4+1-Arc); and one including all of the strengthening valid inequalities described in Section 3.4 (hereafter denoted as Formulation 4+Val. Ieq.), separated in an implicit way as described in the next section. Moreover, for each implementation we considered two kinds of runtime settings. Specifically, we first run each formulation by deactivating Fico Xpress proprietary cuts and presolving strategies. This test aimed at empirically validating the theoretical results discussed in the previous sections without incurring in possible alterations introduced by the optimizer. Conservatively, we also considered a second runtime setting in which we run each formulation by activating both Fico Xpress proprietary cuts and presolving strategies. As default setting, in each implementation we activated both Fico Xpress primal heuristic to generate the first upper bound to the problem and the multithread capabilities of the optimizer in order to exploit the multiple cores of the CPU. Finally, we set 3 hours as maximum runtime for each formulation.

Table 5
Characteristics of the 10 datasets of the SSP considered in the experiments.

Dataset	Jobs	Tools	T^*_M	T^*_m	C	T^*_M/C
datA1	10	10	4	2	4	1
datA2	10	10	4	2	5	0.8
datA3	10	10	4	2	6	0.67
datA4	10	10	4	2	7	0.57
datB1	15	20	6	2	6	1
datB2	15	20	6	2	8	0.75
datC1	30	40	15	5	15	1
datC2	30	40	15	5	17	0.88
datD1	40	60	20	7	20	1
datD2	40	60	20	7	22	0.90

4.3. Separation oracles and branch-and-cut settings

As suggested by Grötschel and Padberg (1985), we used the maximum flow algorithm to search for the most violated subtour elimination constraints (2d), (3d) or (13d) (if any) at any node of the search tree. After a few preliminary experiments, we have decided to allow the introduction of at most 15 violated constraints at the root node and at most 5 violated constraints at any other node.

When solving Formulation 2, we computed the lower bound on the optimal solution to the subproblem corresponding to a generic node of the search tree by solving a linear program in which (i) the subtour elimination constraints and the integrality constraints have been relaxed and (ii) the strengthening valid inequalities described in Laporte et al. (2004) have been added. We searched for violated subtour elimination constraints by using Grötschel and Padberg's (1985) 2 procedure when the solution of the strengthened linear relaxation proved not to be feasible. We experienced no computational benefit in following either the branching strategy or the proposed depth-first tree search strategy described in Laporte et al. (2004). Possibly, this result is due to the improved proprietary search strategies implemented in the Xpress Optimizer. Hence, we let Xpress Optimizer decide both the best candidate binary variable for branching and the best strategy to explore the tree search. We followed similar approaches also for the other implementations.

Concerning Formulation 4+Val. Ieq., we separated the cut inequalities (20) at each node of the search tree by using the following procedure. Consider inequality (20) and add $y^t(V \setminus S, k)$ to both sides. Then, it holds that

$$y^t(V \setminus S, S \setminus \{k\}) + y^t(V \setminus S, k) \ge \sum_{i \in S \setminus \{k\}} y^t_{ik} + y^t(V \setminus S, k). \qquad (23)$$

In turn, inequality (23) is equivalent to

$$y^t(V \setminus S, S) \ge \sum_{i \in J^k} y^t_{ik} \qquad (24)$$

Hence, the separation of the cut inequalities can be performed by finding the largest set $\hat{S}$ for which (24) holds. The search can be carried out by solving $n-1$ max flows from any node in $J_t \setminus \{k\}$ to node k. After preliminary tests, we allowed the introduction of at most 10 violated constraints at the root node and 5 violated constraints at any other node.

Finally, we used a particular branching strategy for binary variables consisting of giving priority to x variables with respect to y variables during the branching process.

4.4. Computational results: analysis of the gap

Figs. 3 and 4 show the results of our computational experiments in terms of box and whisker plots. Specifically, the bottom and the top of each box represent the first and third quartiles; the band inside

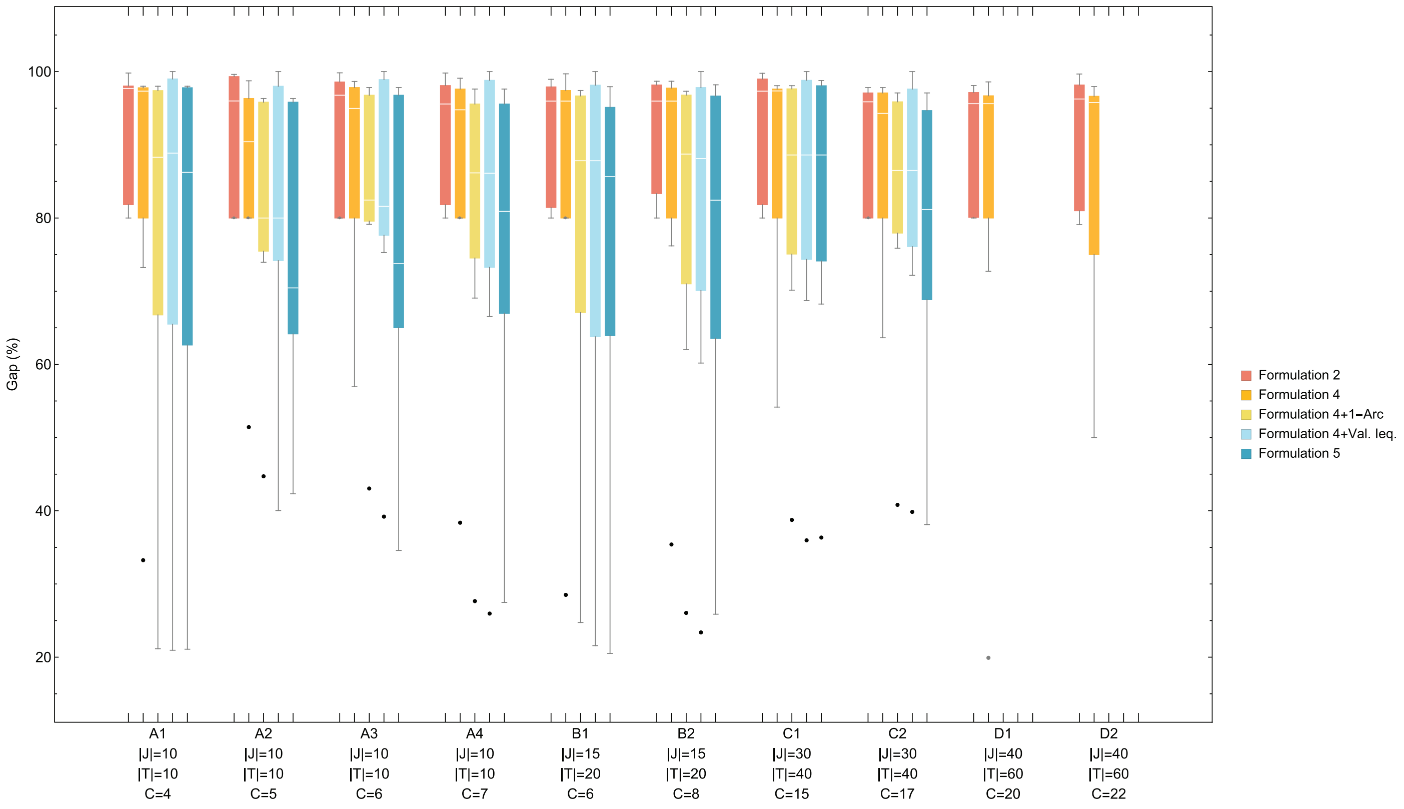

Fig. 3. Comparison of the gaps (expressed in percentage) of Formulations 2, 4 and 5 on the datasets from Laporte et al. (2004) when disabling Fico Xpress proprietary cuts and presolving strategies. In particular, "1-Arc" refers to the activation of the 1-arc inequalities in Formulation 4; similarly, "Val. Ieq." refers to the activation of the strengthening valid inequalities for Formulation 4 described in Section 3.4.

the box represents the second quartile (the median), and the ends of the whiskers represent the 9th percentile and the 91st percentile. Outliers are plotted as individual points. We analyze the formulations with respect to the gap, i.e., the difference between the optimal value to a given instance of the SSP in a specific dataset and the objective function value of the linear programming relaxation at the root node of the respective search tree, divided by the optimal value. In particular, Fig. 3 shows the performances of the formulations when disabling Fico Xpress proprietary cuts and presolving strategies. In contrast, Fig. 4 shows the performances of the formulations when enabling the proprietary strategies. The gap is expressed in percentage; the performances are (i) represented by means of box and whiskers plots and (ii) shown in function of the capacity of the magazine and the size of the datasets.

Fig. 3 shows that at the root node of the search tree Formulation 4 provides median gaps that are generally smaller than those of Formulation 2. This fact confirms the theoretical results discussed in Section 3.1. The trend is more marked in datasets datA2 and datA3 and less in the others, for which the improvements are marginal. In particular, we have observed that the lower bounds (and more in general the computational performances) of the formulations deteriorate in a manner directly proportional to the ratio T^*_M/C, where T^*_M denotes the maximum number of tools per job. Specifically, the smaller the ratio is, the poorer (i.e., the larger) are the gaps. This phenomenon is not surprising, as smaller ratios increase the combinatorial possibilities for the management of the tools in the magazine.

The results showed that the lower bound provided by Formulation 4 at the root node of the search tree increases, on average, when introducing the valid inequalities described in Section 3.4. In particular, we observed that the 1-arc inequalities increase the objective function value of the linear programming relaxation of Formulation 4 up to 17 percent and the combination of the 1-arc inequalities with the cut inequalities may lead to improvements of up to 20 percent. However, with smaller T^*_M/C ratios, the benefits provided by the valid inequalities become less significant and the corresponding objective function values of the linear programming relaxations at the root node tend to prescribe the same value. Adding the valid inequalities to Formulation 4 may not be a trivial task. In fact, due to the large number of inequalities involved, the task entails the use of a cut generation approach, which in turn introduces a computational overhead that may prevent the convergence of the simplex algorithm within the time limit.

The results showed that Formulation 5 is the tightest formulation for the SSP, as it provides on average the best lower bounds. In particular, in the datasets datA2, datA3, datA4, datB2 and datC2, the lower bounds provided by Formulation 5 are on average from 6 percent to 10.7 percent better than the ones provided by Formulation 4+Val. Ieq. In the datasets datA1 and datB1, the lower bounds provided by Formulation 5 are on average from 2 percent to 1.4 percent better than the ones provided by Formulation 4+Val. Ieq. In contrast, in datC1 the lower bounds of Formulation 5 are on average equivalent to the ones provided by Formulation 4+Val. Ieq. Formulation 5 was unable to compute, within the limit time, the linear programming relaxations at the root node of any instance in the datasets datDx, due to the large number of variables and inequalities involved in the formulation. As a general trend, we observed that the lower bounds provided by Formulation 5 have a behavior similar to the ones provided by Formulation 4 and its valid inequalities, i.e., they tend to deteriorate in a manner directly proportional to the ratio T^*_M/C.

Fig. 4 shows that the activation of Fico Xpress proprietary cuts and presolving strategies from one hand causes a general decrement of the median gaps related to the formulations but on the other hand does

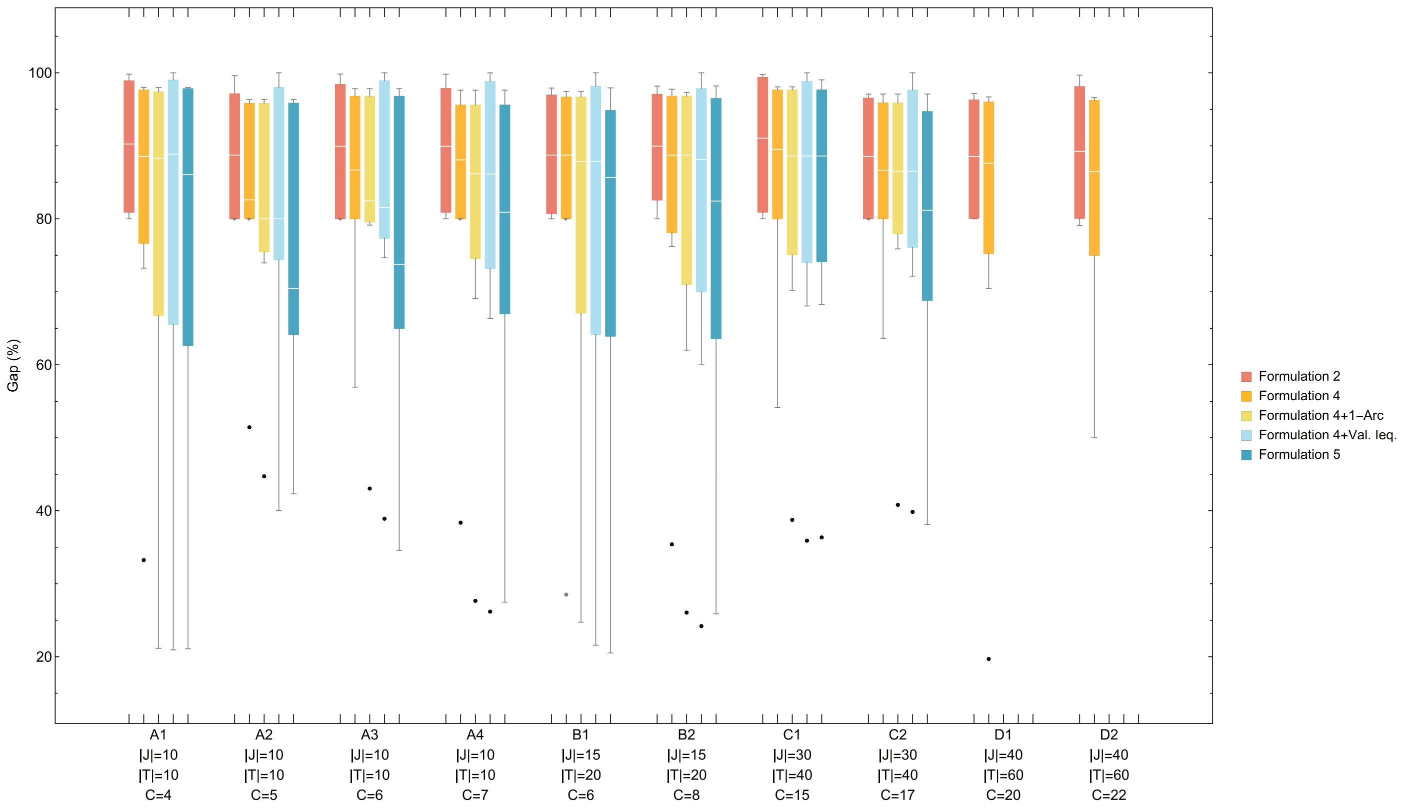

Fig. 4. Comparison of the gaps (expressed in percentage) of Formulations 2, 4 and 5 on the datasets from Laporte et al. (2004) when enabling Fico Xpress proprietary cuts and presolving strategies. In particular, “1-Arc” refers to the activation of the 1-arc inequalities in Formulation 4; similarly, “Val. Ieq.” refers to the activation of the strengthening valid inequalities for Formulation 4 described in Section 3.4.

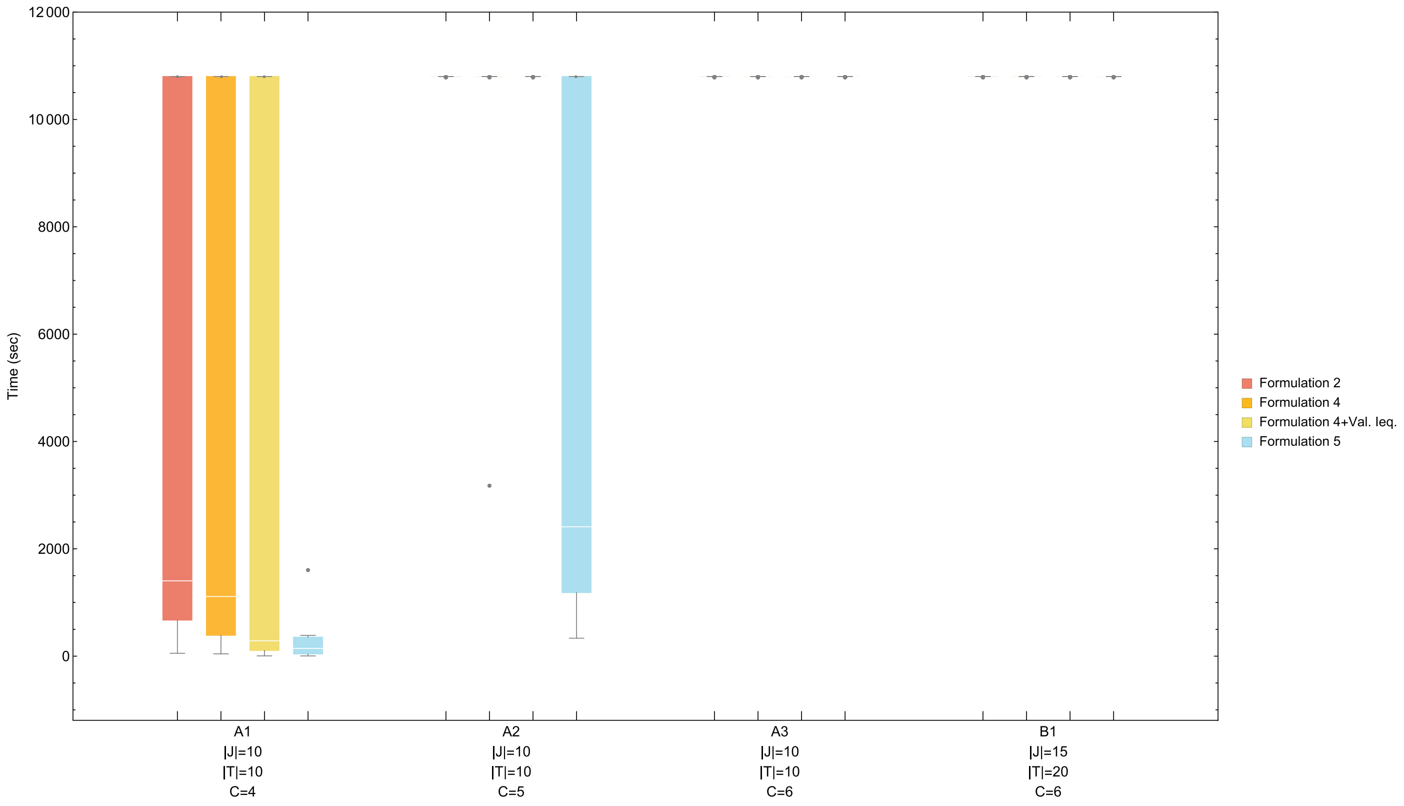

Fig. 5. Comparison of the solution times (expressed in seconds) of Formulations 2, 4 and 5 on the datasets from Laporte et al. (2004) when disabling Fico Xpress proprietary cuts and presolving strategies. In particular, “Val. Ieq.” refers to the activation of the strengthening valid inequalities for Formulation 4 described in Section 3.4.

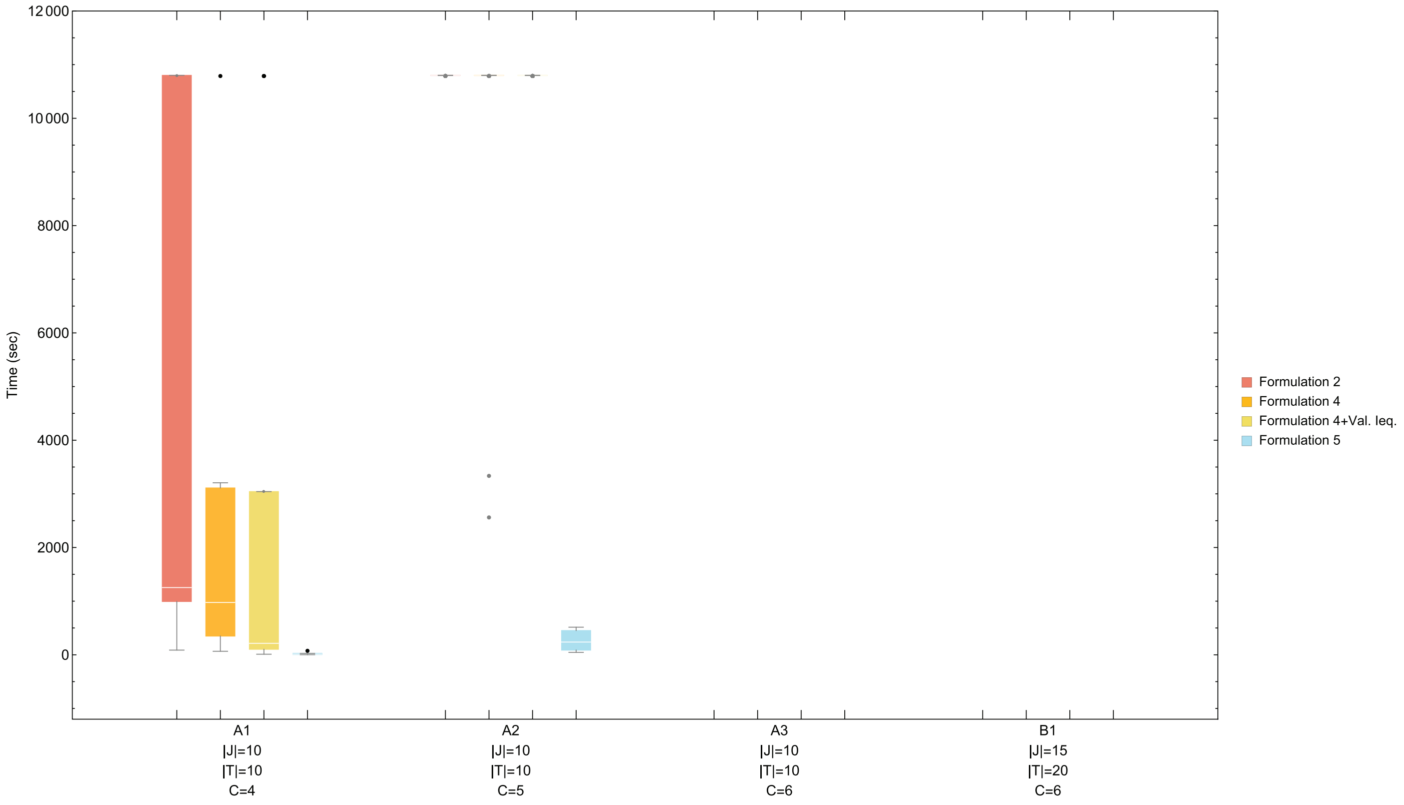

Fig. 6. Comparison of the solution times (expressed in seconds) of Formulations 2, 4 and 5 on the datasets from Laporte et al. (2004) when enabling Fico Xpress proprietary cuts and presolving strategies. In particular, "Val. Ieq." refers to the activation of the strengthening valid inequalities for Formulation 4 described in Section 3.4.

not change their general trends. In particular, the figure shows that the use of the proprietary strategies has a major impact on Formulations 2 and 4, minor in Formulation 4+1-Arc and Formulation 4+Val. Ieq., and becomes negligible or absent in Formulation 5.

4.5. Computational results: analysis of the solution times

Figs. 5 and 6 show the solution time (expressed in seconds) taken by the formulations to exactly solve Laporte et al.'s datasets. In particular, Fig. 5 shows the results obtained when disabling Fico Xpress proprietary cuts and presolving strategies. In contrast, Fig. 6 shows the results obtained when enabling those strategies. As for Figs. 3 and 4, the performances are (i) represented by means of box and whiskers plots and (ii) showed as a function of the capacity of the magazine and the size of the datasets. In preliminary tests we have observed that Formulation 4+1-Arc and Formulation 4+Val. Ieq tend to have similar solution times; hence, we preferred to exclude Formulation 4+1-Arc from this analysis.

The results showed that Formulation 2 was able to solve (within the limit time) only 6 instances in the dataset datA1, either when activating or deactivating Fico Xpress proprietary cuts and presolving strategies (see Table 6). Formulation 4 was able to solve 7 instances when deactivating the proprietary strategies and 8 instances when activating them. When using also the valid inequalities, Formulation 4 was able to solve 7 instances when deactivating the proprietary strategies and 9 instances when activating them. Finally, Formulation 5 was able to solve 10 instances either when activating or deactivating the proprietary strategies. As general trend, the results showed that the use of Fico Xpress proprietary cuts and presolving strategies reduces the solution time and may increase the number instances optimally solved.

Formulation 2 was unable to solve any instance in the dataset datA2 within the time limit, either when activating or deactivating Fico Xpress proprietary cuts and presolving strategies. Formulation 4 was able to solve 1 instance in datA2 when deactivating the proprietary strategies and 2 instances when activating the proprietary strategies. Unfortunately, the combination of Formulation 4 with the valid inequalities did not prove successful. In fact, independently of the use of the proprietary cuts and presolving strategies, Formulation 4+Val. Ieq proved to be unable to solve any instance in the dataset datA2 within the time limit, mainly due to the computational overhead introduced by the separation oracle. Finally, Formulation 5 was able to solve all of the instances in datA2 both when activating and deactivating the proprietary strategies. The

Table 6
Number of instances optimally solved in Laporte et al.'s datasets. The columns "With XPC" and "No XPC" refer to the activation and the deactivation, respectively, of Fico Xpress proprietary cuts and presolving strategies.

Dataset	Formulation 2		Formulation 4		Formulation 4+Val. Ieq.		Formulation 5	
	No XPC.	With XPC	No XPC	With XPC	No XPC	With XPC	No XPC	With XPC
datA1	6	6	7	8	7	9	10	10
datA2	0	0	1	2	0	0	6	10
datA3	0	0	0	0	0	0	0	0
datB1	0	0	0	0	0	0	0	0

results showed that no formulation was able to solve instances having characteristics larger than or equal to those contained in the dataset datA3.

5. Conclusion

In this article we have investigated the job Sequencing and tool Switching Problem (SSP), a $\mathcal{NP}$-hard combinatorial optimization problem arising from manufacturing systems. In particular, starting from the results discussed in Tang and Denardo (1987), Crama (1997), Crama et al. (1994), Laporte et al. (2004) and Ghiani et al. (2010), we have developed three integer programming formulations for the SSP that are provably tighter than the ones previously described in the literature. Computational experiments have confirmed the theoretical results and suggested Formulation 5 as the best ILP formulation for the SSP, being characterized by the fastest running times and largest number of solved instances. The results showed however that the new formulations are characterized by very large gaps. Moreover, the large number of involved variables and valid inequalities slows down the solution times of the formulations and may limit their practical use. In our opinion, a possible way to reduce the computational overhead could consist of both optimizing the runtime performances of the separation oracles and using faster programming languages such as C/C++. We also believe that the use of projective approaches may prove indispensable to overcome the current limitations. Investigating these approaches warrants additional research efforts.

Acknowledgments

This work is supported by National Funding from FCT - Fundação para a Ciência e a Tecnologia, Portugal, under the project: PEst-OE/MAT/UI0152. The last author was supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. The authors thank the anonymous reviewers for their valuable comments on previous version of the manuscript.

References

Askin, R. G., Sodhi, M. S., & Sen, S. (1994). Multiperiod tool and production assignment in flexible manufacturing systems. *International Journal of Production Research*, *32*, 1281–1294.

Avci, S., & Akturk, M. S. (1996). Tool maganize arrangement and operations sequencing on CNC machines. *Computers and Operations Research*, *23*, 1069–1081.

Balakrishnan, N., & Chakravarty, A. K. (2001). Opportunistic retooling of a flexible machine subject to failure. *Naval Research Logistics*, *48*, 79–97.

Bard, J. F. (1988). A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, *20*, 382–391.

Belady, L. A. (1966). A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, *5*, 78–101.

Blazewicz, J., & Finke, G. (1994). Scheduling with resource management in manufacturing systems. *European Journal of Operational Research*, *76*, 1–14.

Crama, Y. (1997). Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, *99*, 136–153.

Crama, Y., Kolen, A. W. J., Oerlemans, A. G., & Spieksma, F. C. R. (1994). Minimizing the number of tool switches on a flexible machine. *The International Journal of Flexible Manufacturing System*, *6*, 33–54.

Crama, Y., Moonen, L. S., Spieksma, F. C. R., & Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, *182*, 952–957.

Garey, M. R., & Johnson, D. S. (2003). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.

Ghiani, G., Grieco, A., & Guerriero, E. (2010). Solving the job sequencing and tool switching problem as a nonlinear least cost hamiltonian cycle problem. *Networks*, *55*(4), 379–385.

Gray, A. E., Seidmann, A., & Stecke, K. E. (1993). A synthesis of decision models for tool management in automated manufacturing. *Management Science*, *39*, 549–567.

Grötschel, M., & Padberg, M. W. (1985). Polyhedral theory. In E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, & D. B. Shmoys (Eds.), *The traveling salesman problem: A guided tour of combinatorial optimization* (pp. 251–305). Chichester, UK: Wiley.

Hertz, A., Laporte, G., Mittaz, M., & Stecke, K. E. (1998). Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE Transactions*, *30*, 689–694.

Hertz, A., & Widmer, M. (1993). An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics*, *65*, 319–345.

Laporte, G., Salazar-Gonzáles, J. J., & Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, *36*, 37–45.

McGeoch, L. A., & Sleator, D. D. (1994). A strongly competitive randomized paging algorithm. *Algorithmica*, *6*, 816–825.

Privault, C., & Finke, G. (2000). K-server problems with bulk requests: An application to tool switching in manufacturing. *Annals of Operations Research*, *96*, 255–269.

Tang, C. S., & Denardo, E. V. (1987). Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. *Operations Research*, *36*(5), 767–777.