A New Multicommodity Flow Model for the Job Sequencing and Tool Switching Problem

Tiago Tiburcio da Silva
Antônio Augusto Chaves
Horacio Hideki Yanasse

Univ. Fed. de São Paulo*, São José dos Campos, SP, Brazil

Abstract

In this paper a new multicommodity flow mathematical model for the Job Sequencing and Tool Switching Problem (SSP) is presented. The proposed model has a LP relaxation lower bound equal to the number of tools minus the tool machine's capacity. Computational tests were performed comparing the new model with the models of the literature. The proposed model performed better, both in execution time and in the number of instances solved to optimality.

combinatorial optimization; job sequencing and tool switching; multicommodity flow; integer programming; linear programming relaxation.

1 Introduction

Flexible Manufacturing Systems (FMS) are automated production systems, capable of producing a wide variety of products. These systems are designed to adapt and react to changes in industrial processes, including failures (Gamila and Motavalli (2003); Zeballos (2010)). FMS has to satisfy requirements of efficiency and flexibility. Therefore the problems of production planning are more difficult compared to those found in other production systems.

According to Stecke and Solberg (1981) there are five possible classifications for production planning problems in a FMS: Part type selection problems, Machine grouping problems, Product ratio problems, Resource allocation problems, and Loading problems. The Job Sequencing and Tool Switching problem (SSP) is classified as a Loading problem. The SSP consists in determining a sequencing of jobs, so the total number of tool switches is minimized. A tool switch is computed when we have the removal of a tool from the magazine and the insertion of another tool in its place. The magazine can accommodate any tools

*MeMO-2020

needed to process any job, but its capacity is limited. All the tools required cannot be all together in the magazine; therefore, tool switches are required.

The SSP arises in settings where unnecessary tool switches result in underutilization of the machine, an unacceptable level of machine downtime, and more time spent on tool switches than any other setup time (Agapiou (1991); Van Hop and Nagarur (2004); Melnyk, Ghosh, and Ragatz (1989); Zhang and Hinduja (1995)).

In this paper, we consider that (Widmer (1991); Zhou, Xi, and Cao (2005)):

- There are a set of jobs to be processed and each job requires a fixed set of specific tools;
- No job requires a set of tools that exceeds the capacity of the machine;
- All tools are always available;
- A single machine is available to process all jobs;
- It is an *offline* version of the problem;
- Once the machine has started processing a job, it must be finished completely;
- Only one tool switch is held at time;
- Each tool fits in any *slot* of the magazine and occupies only a single *slot*;
- The time associated with the removal and insertion of a tool (a switch) is independent and constant;
- No breaks, no wear and no maintenance of the tools are considered.

The SSP is a $\mathcal{NP}$-hard problem (Crama et al. (1994); Tang and Denardo (1988a)). Tang and Denardo (1988a) showed that the SSP and the Traveling Salesman Problem (TSP, Dantzig, Fulkerson, and Johnson (1954)) are related. They viewed the SSP as a composition of two inter-related subproblems:

1. The *Sequencing problem* (SP) that consists in determining an optimum sequence of processing the jobs;
2. The *Tooling problem* (TP) that consists in determining the tools that must be in the machine during the processing of each job in order to minimize the tool switches given a fixed sequence of processing the jobs.

For the solution of the Tooling problem, there is an optimal policy proposed by Tang and Denardo (1988a). This policy called KTNS (Keep Tool Needed Sooner) defines that whenever a new tool is inserted and a tool from the magazine must be removed, the tools that should be kept are those that will be used soonest.

Table 1: An example of an instance of the SSP with 6 jobs, 9 tools and capacity equal to 4 (Catanzaro, Gouveia, and Labbé (2015)).

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		

Table 2: An optimal solution of the instance of the SSP shown in Table 1.

Jobs	4	2	5	3	6	1
Tools	1	1	$\underline{8}$	8	**8**	8
	5	5	5	$\underline{6}$	$\underline{1}$	1
	7	**7**	**7**	7	$\underline{4}$	4
	9	$\underline{3}$	3	$\underline{2}$	2	$\underline{9}$
Number of Tool Switches	0	1	1	2	2	1

In Table 1 we show an example of an instance of the SSP. In the first row in this table the jobs that will be sequenced are identified with numbers. In each column, the set of tools necessary to produce the corresponding job is presented. The same information given in Table 1 can be provided using a binary job-tool matrix where each row corresponds to a tool and each column corresponds to a job. In this matrix, a 1 in row i and column j indicates that to process job j, we need tool i and, a 0 indicates otherwise.

In Table 2 we show an optimal solution of the instance presented in Table 1. In the first row, we have the order that the jobs are processed. An optimum processing sequence is $4-2-5-3-6-1$. The tools in bold are unnecessary to process the specific job, but maintaining them in the magazine result in the minimum number of tool switches induced by the specific sequence of the jobs. This solution results in a total number of tool switches equal to 7: 1 switch between jobs 4 and 2; 1 switch between jobs 2 and 5; 2 switches between jobs 5 and 3; 2 switches between jobs 3 and 6; and 1 switch between jobs 6 and 1. The underscored tools in Table 2 are the tools that were inserted into the machine in place of the others that were withdrawn from the magazine.

In this article we propose a new multicommodity flow model for the SSP, which has improved Linear Programming (LP) relaxation compared to the existing models of the literature. The main achievements obtained are:

- The lower bound for the objective function value given by the new model is at least the difference between the number of different tools and the capacity of the tool magazine in the machine;
- The lower bound provided by the new model is, in general, better than

the lower bounds given by other models of the literature;

- A constraint is proposed that eliminates half of the possible symmetric solutions for the problem;
- The CPLEX solver using the new model was able to solve 17.01% more instances of the literature compared to other previous known models.

The remainder of the paper is organized as follows. In Section 2 we present a literature review of the SSP. In Section 3 we present the new multicommodity flow based formulation for the SSP. Computational results for the proposed model are presented and analyzed in Section 4 followed by some conclusions in Section 5.

2 Related works

Several solution approaches and some mathematical models have been proposed for the SSP. The SSP is a $\mathcal{NP}$-hard problem Crama et al. (1994); Tang and Denardo (1988a); hence, particular attention has been given to heuristics and metaheuristics strategies to solve it. The number of exact solution approaches proposed in the literature to solve the SSP is quite limited.

One of the earliest and impactful studies on the SSP was produced by Tang and Denardo (1988a). In this work, they showed that the problem is $\mathcal{NP}$-hard (as in Crama et al. (1994)) through a reduction to the Traveling Salesman problem. The authors presented an integer linear programming model with binary variables for the problem that was solved using the IBM MPSX/MIP and PIPX codes in an IBM 370. The authors proposed a Greedy Perturbation (GP) heuristic based on three steps. First, it determines a sequence of jobs. Second, using the KTNS policy, the minimum number of tool switches corresponding to this sequence is determined. Finally, the sequence is perturbed in order to determine a sequence that produces the least number of tool switches. The conclusion was that the GP procedure is efficient for small instances but inefficient for large instances.

Bard (1988) proposed an integer nonlinear programming model with binary variables for the SSP. The objective function was linearized and several codes were used to solve the linearized model, among them the IBM MPSX/MIP code in an IBM 3081. The codes, however, did not obtain good results in practical acceptable times for one instance tested by the authors with $N = 10$ jobs, $M = 9$ tools and $C = 4$ slots. By dualizing machine capacity constraints and the tool demand constraints using Lagrange multipliers, two subproblems were created, one solved by the Hungarian Algorithm and the other by Dynamic Programming. Tests were performed with instances with a maximum of 25 jobs and the results obtained were compared with an (incomplete) enumerative procedure, reporting in some cases the same values. Bard's linearized model is the same model proposed by Tang and Denardo (1988a). The author mentions two applications of the SSP in industries, and he discusses an extension of the problem to multiple machines where the sequence of jobs remains fixed and the same in all machines.

Crama et al. (1994) and Oerlemans (1992) presented a proof that the SSP is $\mathcal{NP}$-hard whenever the machine capacity is greater than or equal to 2. They also presented an integer formulation with binary variables, based on maximum 0-block subsets, i.e. a maximal subset of consecutive zeros delimited by $1's$ in a row of a job-tool matrix for the Tooling problem when a job sequence is fixed. They proved that this formulation can be solved in polynomial time and provided a new proof for the validity of the KTNS policy developed by Tang and Denardo (1988a). The authors also established links between the SSP and other combinatorial optimization problems: the matrix permutation problem (Gate Matrix Layout, Möhring (1990)), optimization with greedy constraint matrices (Hoffman, Kolen, and Sakarovitch (1985)), the block minimization problem (Kou (1977)), and recognition of interval matrices (Fulkerson and Gross (1965)). As the Tooling problem is well solved by the KTNS policy, the authors focused on the Sequencing problem and proposed eleven heuristics divided into two classes, construction heuristics and refinement heuristics. Among the construction heuristics are four heuristics based on the TSP, where each node represents a job and the edge cost is the cost of sequencing two successive jobs.

Hertz et al. (1998) modeled the Sequencing problem as a TSP and used the KTNS policy to solve the Tooling problem. They presented nine heuristics to solve the resultant TSP. The authors used several metrics to estimate the cost of an edge on the TSP graph, including one that takes into account the overall vision of the entire solution, in addition to the interactions between two jobs at a time. To test the effect of the metrics on the heuristics, extensive computational tests were performed, using instances generated in a similar way as Crama et al. (1994). The authors concluded that the GENI and GENIUS heuristics generated better solution values than those presented by Crama et al. (1994). The GENI heuristic is a construction heuristic for solving the TSP proposed by Gendreau, Hertz, and Laporte (1992) and, the GENIUS heuristic is the GENI heuristic combined with a post optimization phase.

Privault and Finke (1995, 2000) modeled the SSP as a K-Server Problem with Bulk Request, where the slots on the machine represent the servers and tool sets required by a job corresponding to a bulk request. The authors proposed a heuristic adapted from the partitioning algorithm of McGeogh and Sleator (1991) for the K-Server Problem and concluded that their modified heuristic provided better upper bounds when compared to job grouping heuristics Algo_1 and Algo_2. These grouping heuristics iteratively group jobs that use common tools, creating news fictitious jobs that require a total number of tools that do not exceed the capacity of the tool magazine. The new jobs are scheduled according to the Shortest Edge heuristic (Algo_1) or the Farthest Insertion heuristic (Algo_2) and afterwards, a 2-opt heuristic is applied to modify the sequence obtained to improve the objective function (Bentley (1992)).

Djellab, Djellab, and Gourgand (2000) reformulated the SSP using a particular representation of hypergraphs. Based on this formulation, the authors proposed an iterative best insertion heuristic (IBI). The authors concluded that their results improved the results of Crama et al. (1994), in terms of both computational execution time and solution quality. Moreover, the performance of IBI heuristic did not vary with the sparsity of the job-tool matrix.

Shirazi and Frizelle (2001) developed an empirical study of the SSP in seven industries that use high technology and manufacturing processes. The main objective of the study was to compare the behavior and solutions obtained by the heuristics of the literature against the methods used in the industries. The authors used six heuristics of the literature that produced significantly better results than existing industry methods, getting an average tool switch saving of 21.1%.

Yanasse and Pinto (2002) proposed a network-based model for the SSP, and developed preprocessing operations to decrease the size of the problem. In Yanasse and Lamosa (2005, 2006), the authors modeled the SSP as a Generalized Traveling Salesman Problem (GTSP) and proposed a solution method to solve it, but they did not perform any computational studies.

To overcome the limitations of Tang and Denardo's model (Tang and Denardo (1988a)), Laporte, Salazar-Gonzáles, and Semet (2004) formulated an integer programming model based on the TSP, inserting a dummy job, indicating the start and the end of the processing of the jobs, and modeling the subtour elimination constraints as in Dantzig, Fulkerson, and Johnson (1954). The authors showed that their new formulation outperformed the one proposed by Tang and Denardo (1988a), regarding the values of their linear programming relaxations, after showing that the linear programming relaxation of Tang and Denardo's model is equal to zero. The authors proposed several valid inequalities to strengthen their model and two exact enumeration algorithms to solve the SSP. The first one was a Branch-and-cut algorithm, where, in the root node, the authors used the GENIUS heuristic with the KTNS policy to construct a feasible solution. Subtour elimination constraints of their model were not considered in the beginning. They were introduced as needed. At each node of the search tree, the most violated subtour elimination constraints were introduced. When no violated inequality could be identified, broad branching was carried out; i.e. given a sequence of fixed jobs, new nodes, corresponding to the jobs that are not in this sequence, are created. Computational tests showed that this branch-and-cut algorithm was able to solve to optimality some instances with 9 jobs. In addition, the lower bounds obtained by their model were better than the ones provided by Tang and Denardo's model, but they were still weak. The second algorithm was a Branch-and-Bound that enumerated all possible sequences of the jobs. An upper bound for the SSP was obtained by a greedy heuristic. The authors proposed two values for the lower bounds: a simple lower bound, based on the partial sequence of the jobs sequenced and, the result of the Minimum Spanning Tree Problem for the jobs not yet sequenced. The branching rule uses the order in which the jobs appear in the greedy algorithm. Computational tests showed that this branch-and-bound algorithm was able to solve to optimality a few instances with 25 jobs.

Ghiani, Grieco, and Guerriero (2007) demonstrated that the SSP has the following symmetry property: a sequence of jobs and its reverse order sequence produces the same minimum number of tool switches. The authors used this property in the Branch-and-bound method proposed by Laporte, Salazar-Gonzáles, and Semet (2004), modifying the branching rule so that the algorithm considered a job sequence only once. An advantage of their method was indicated in the case where the instances were close to the instances of the TSP (i.e., the higher the percentage of tool magazine slots occupied by jobs was, on average, the closer to the TSP the SSP instance was).

Yanasse (2009) proposed a new lower bound to the SSP based on the similarity of this problem with the Minimization of Open Stacks Problem (MOSP, Becceneri, Yanasse, and Soma (2004)). The new lower bound was not outperformed by previous lower bounds in the literature.

Yanasse, Rodrigues and Senne (2009) proposed an enumerative scheme based on partial ordering of the jobs to determine the optimal solution of the SSP. The authors proposed the determination of lower bounds using subsets of the original set of jobs. The proposed method defined an initial partial subset of jobs, and successively added jobs neatly to this set in such a way that the number of tool switches remained the same or increased by a small amount. The method was tested in randomly generated instances and was able to solve 1233 instances, against the 1136 instances solved in Crama et al. (1994).

Ghiani, Grieco, and Guerriero (2010) formulated the SSP as a minimal cost Hamiltonian cycle problem with a nonlinear objective function, and subtour elimination constraints modeled by Dantzig, Fulkerson, and Johnson (1954). The authors used the symmetry property of the SSP (Ghiani, Grieco, and Guerriero (2007)) in the proposed branch-and-cut algorithm, establishing rules of dominance and using separation techniques to detect violated subtour elimination constraints in its formulation. The proposed method was able to solve a few instances with 45 job and 35 tools.

Al-Fawzan and Al-Sultan (2002) proposed 5 different versions of a Tabu Search heuristic to solve the SSP. The authors used short- and long-term memory structures combined with probabilistic and deterministic oscillations to guide the heuristic. All the proposed versions were promising, but the long-term memory structure had a more significant effect on the performance of the Tabu Search.

Zhou, Xi, and Cao (2005) developed a Beam Search algorithm and data preprocessing for the SSP. This algorithm used heuristic to estimate some best paths on the search tree of a branch-and-bound algorithm to be explored. The algorithm improved the results of Bard (1988) both in computational execution time and solution quality.

Salonen, Raduly-Baka, and Nevalainen (2006) proposed a multistart heuristic, combining a job grouping technique, the GENIUS algorithm, and the KTNS policy. The authors tested their method with instances of the SSP provided by Crama et al. (1994) and Smed et al. (1999), and concluded that it performed well, considering the tradeoff between solution quality and execution time.

Amaya, Cotta, and Fernández (2008) proposed a local search algorithm, a permutative genetic algorithm, and a memetic algorithm that combined the two previous methods. The computational results showed that the memetic algorithm outperformed the other two algorithms, and also the Beam Search method proposed by Zhou, Xi, and Cao (2005).

Amaya, Cotta, and Fernández-Leiva (2010) developed four cooperative models, performed empirical tests, and concluded that one of the cooperative models had superior performance compared to the Beam Search Algorithm of Zhou, Xi, and Cao (2005) and the Tabu Search Algorithm of Al-Fawzan and Al-Sultan (2002). Later on, Amaya, Cotta, and Fernández-Leiva (2011) considered 36 memetic cooperative models, that differed among themselves in the particular combination of metaheuristics assigned to agents and their connection topology. Fully memetic models, i.e. cooperative models in which each agent is endowed with a (possibly different) memetic algorithm, were the ones that provided the best results, improving the results of the algorithm proposed in Amaya, Cotta, and Fernández (2008).

In Amaya, Cotta, and Fernández-Leiva (2012), three different local search techniques (Hill Climbing, Tabu Search, and Simulated Annealing) were embedded in evolutionary algorithms. Computational tests showed that the memetic algorithm embedded with the Hill Climbing search yielded the best results, performing better than its stand-alone constituents, including the Beam Search heuristic defined in Zhou, Xi, and Cao (2005).

Amaya, Cotta, and Fernández-Leiva (2013) developed a memetic algorithm based on the Cross Entropy technique, which is a Monte Carlo approach to optimization (Rubinstein (1999)), which evolves a population based on a probability mass function, which is then updated in each generation. Computational tests showed an improved performance of this new algorithm compared to the algorithms presented in Amaya, Cotta, and Fernández-Leiva (2011). Amaya et al. (2019) proposed a deep meta-cooperation model to solve SSP, and computational results showed better performance in relation to the results achieved by Amaya, Cotta, and Fernández-Leiva (2013).

Senne and Yanasse (2009) developed three Beam Search metaheuristics based on an enumeration scheme that considered the partial ordering of jobs. All variations used the depth search for branching the enumeration tree. The authors generated 1,350 instances according to the parameters in Laporte, Salazar-Gonzáles, and Semet (2004), and they concluded that the algorithm that maintained only the 3 best branches at each node of the enumeration tree, and randomly chose the next node to be branched, presented the best results among the proposed methods.

Chaves, Senne, and Yanasse (2012) proposed a two-phase heuristic method. In the first phase, a solution was constructed from a SSP graph. In the second refinement phase, the solution generated in the first phase was improved by applying an Iterative Local Search metaheuristic (Stützle (1999)). A SSP graph is a graph where the vertices are the tools and an edge exists if, and only if, the two tools that are incident to this edge are used together in the processing of some job. This graph is also known as a MOSP graph (Yanasse (1997)). The solution found was then used as the initial upper bound in the enumerative method proposed in Yanasse, Rodrigues and Senne (2009). The heuristic method found the optimal solution for the great majority of the instances tested.

Burger et al. (2015) modeled the color print scheduling problem (CPSP) as a SSP, where each job was packaging, and each tool was an ink cartridge. They proposed and implemented a heuristic method based on Salonen et al. (2006). Random problem instances were created and solved with the proposed heuristic. The instances were also solved using CPLEX 11.0 with Tang and Denardo (1988a) and Laporte, Salazar-Gonzáles, and Semet (2004) formulations. The heuristic proposed produced relatively good results, especially in cases where the number of tools required for processing a job was close to the tool magazine capacity. The authors also applied the proposed heuristic to a real case study of a printing company located in the South African Western Cape, obtaining an improvement, on average, of 61.6% on the number of ink cartridge switches.

Laporte, Salazar-Gonzáles, and Semet (2004) showed that the linear relaxation of their formulation outperformed the linear relaxation of the formulation proposed by Tang and Denardo (1988a). According to Catanzaro, Gouveia, and Labbé (2015), the reason that few authors have attempted exact methods for the SSP, is due to the fact that the lower bounds obtained by the formulations proposed in the literature are weak. Hence, Catanzaro, Gouveia, and Labbé (2015) developed three integer linear programming formulations for the SSP to strengthen the lower bounds of existing formulations in the literature. They showed that the lower bounds obtained by the linear relaxations of their formulations were at least equal to those obtained by Laporte, Salazar-Gonzáles, and Semet (2004). The best formulation among the proposals consisted of a flow formulation in arcs based in structural properties of the Tooling subproblem and 0-blocks.

Chaves et al. (2016) developed a hybrid method, combining Clustering Search and a Biased Random-Key Genetic Algorithm (BRKGA). This method detected promising regions in the search space, i.e. regions that could contain good solutions, and applied a local search only those regions. The local search heuristic used was the Variable Neighborhood Descent. The authors used 1,510 instances (Senne and Yanasse (2009), Crama et al. (1994)) to test the proposed method, and found an optimal solution in 1,360 of the instances. For the remaining 150 instances, the method found better solutions compared to those of the literature.

Beezao (2016) studied the SSP as a Generalized Traveling Salesman Problem (Yanasse and Lamosa (2005, 2006)) and proposed four versions of a domain reduction algorithm that were: greedy, reverse, approximation of isolated nodes and path relinking strategies. The author concluded that her method was competitive when applied to small and medium size instances.

Paiva and Carvalho (2017) introduced a new graph-based representation to the SSP called the Tool Graph, in which the set of nodes represented the tools, an edge represented a pair of tools required by the same job, and the weight of each edge was the number of jobs that required both tools. The authors proposed an Iterated Local Search metaheuristic, where the initial solution was given by a Tool Graph search-based heuristic that analyzed the relationship between the tools, and proposed a local search method based on a 1-block grouping, i.e. a maximal subset of consecutive $1'$s delimited by zeros in a row of a job-tool matrix for the Tooling problem, when a job sequence was fixed. The authors evaluated the proposed metaheuristic, applying it to 1,670 instances of the literature (Crama et al. (1994), Yanasse, Rodrigues and Senne (2009) and Catanzaro, Gouveia, and Labbé (2015)), and they concluded that the performance of their metaheuristic was superior to the one proposed by Chaves et al. (2016).

Ahmadi et al. (2018) modeled the Sequencing problem by a second-order TSP (Jäger and Molitor (2008)) called 2-JSeP, and showed that this induced version of problem SSP is $\mathcal{NP}$-hard. In a second-order TSP, the costs do not depend on arcs, but on each sequence of three consecutive vertices in the tour. The authors used the obtained solution by 2-JSeP to seed a genetic algorithm combined with machine learning (Dynamic Q-learning-based Genetic Algorithm) and compared their results with the ones in Paiva and Carvalho (2017). The proposed algorithm had an improved performance, both in execution time and quality of the solutions, compared to the algorithm of Paiva and Carvalho (2017).

In his Master's dissertation, Dehaybe (2018) addressed the SSP and proposed a metaheuristic, hybridizing an Ant Colony algorithm with a new local search method named 2.75-opt, that combined 2-opt, two simple 3-opt moves and job swap (Bentley (1992)). The proposed metaheuristic outperformed the metaheuristics of Ahmadi et al. (2018) and Paiva and Carvalho (2017) in most of the datasets, determining better solutions compared to the ones of the literature for almost every non-trivial dataset.

There are many variations of the SSP. Among them, we can mention those that consider parallel machines (Fathi and Barnette (2002), Ghrayeb, Phojanamongkolkij, and Finch (2003), Gökgür, Hnich, and Özpeynirci (2018)), machines with modular feeders (Raduly-Baka et al. (2018)), tool switching instants (Tang and Denardo (1988b)), capacity constraint variations (Rupe and Kuo (1997)), job shop scheduling (Gao and Moon (2019)), non-uniform tool sizes (Matzliach and Tzur (1998)), tool transporter movements (Song and Hwang (2002), Karzan and Azizoglu (2008)), tool indexing time (Baykasoğlu and Ozsoydan (2016)), tool wear (Dadashi, Moslemi, and Mirzazadeh (2016)), tool duplications (Baykasoğlu and Ozsoydan (2017)), dynamic job arrival and tool duplications (Baykasoğlu and Ozsoydan (2018)). A comprehensive literature review of SSP and its many variations is provided in Calmels (2018).

In the next section, we introduce a new model formulation for the SSP.

3 The new multicommodity flow formulation for the SSP

Next, we introduce the notation that will be used in the new proposed model. The total number of jobs to be processed is N and the total number of different tools available to process the jobs is M. Consider that the machine has the capacity to accommodate up to C tools. Let $J = \{1, \ldots, N\}$ be the set of N jobs to be sequenced, $T = \{1, \ldots, M\}$ the set of M tools necessary to process the jobs in J, and T_i, the set of tools necessary to process the job i, $i \in J$.

Consider a graph G(V,A), where the set V is composed by $N + 3$ nodes, numbered 0 to $N + 2$, so that, $V = \{0, 1, \dots, N + 2\}$. We define node 0 as the origin (or supply) node, node $N+2$ as an auxiliary node, and node $N+1$ as the destination (or sink) node. Set A is composed by arcs of the form $(i, i+1)$, for $i = 0, 1, \dots, N$; $(i, N+2)$ for $i = 0, 1, \dots, N-2$; $(N+2, i)$ for $i = 1, 2, \dots, N-1$, and $(i, N+1)$ for $i = 1, 2, \dots, N-1$. The capacity on arc $(i, i+1)$, for $i = 0, 1, \dots, N-1$ is C, and the other arcs have unlimited capacity. The unit cost of transporting any commodity in arc $(i, N+1)$ for $i = 1, \dots, N-1$ and, in arc $(i, N+2)$, for $i = 1, \dots, N-2$ is 1; the costs on all other arcs are 0.

An illustration of graph G(V,A) is presented in Figure 1.

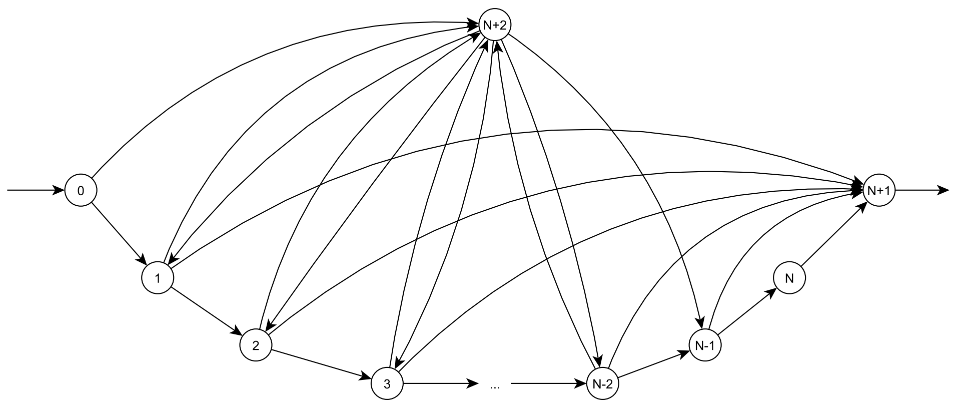

Figure 1: Capacitated graph G(V,A).

Consider a multicommodity flow problem in G(V,A), where at the origin (node 0) there are M types of commodities to be transported, one unit of each commodity type. These commodities must be sent to the destination node $N+1$ at a minimum cost. We are interested in solutions where each commodity uses a single path to transport 1 unit from node 0 to node $N+1$; therefore, the flow of each commodity in any arc cannot be fractional values. Let us associate a tool of the SSP to each commodity. Thus, we can associate a feasible multicommodity flow solution passing through the arcs $(i, i+1)$, for $i = 0, 1, \dots, N-1$ of this graph, with tools in the magazine of a SSP at instants $i = 1, \dots, N$, respectively.

In a SSP, we can always find an optimal solution where the tool magazine is always full (complete) with C tools, when processing any of the N jobs in sequence. Hence, we assume, without loss of generality, that the tool magazine is always full with C tools. This means that the minimum flow on arcs $(i, i+1)$, for $i = 0, 1, \dots, N$ is C and this condition is imposed on the flow model.

A solution to the SSP problem can be obtained by determining a multicommodity flow through the arcs of this graph G(V,A), such that for any job J in the SSP, in at least one arc $(i, i+1)$, $i = 0, 1, \dots, N-1$, we have a unit of the commodity corresponding to each one of the tools that are required to process job J flowing all together, simultaneously. A unit flow of a commodity on an arc $(i, N+2)$, for $i = 1, \ldots, N-2$ indicates a tool taken out of the magazine but still needed to process some job J later on. A unit flow of a commodity on arc $(N+2, i)$ for $i = 1, 2, \ldots, N-1$ indicates a tool that is being inserted in the magazine, a unit flow of a commodity on arc $(i, N+1)$, $i = 1, 2, \ldots, N$ indicates a tool taken out of the magazine and not required anymore to process any job not yet processed, and a unit flow of a commodity on arc $(0, N+2)$ indicates a tool that was left out of the magazine when the processing of the jobs are initiated in the machine to be inserted later on. Since M is greater or equal to C and the flow on arc $(0,1)$ is C, then the flow on arc $(0, N+2)$ is equal to $M - C$.

We next formulate the SSP as a multicommodity network flow problem considering the previous explanation. We denote this model by SSPMF (Job Sequencing and Tool Switching Problem modeled as a Multicommodity Flow Problem). Let x_{ik} be a binary decision variable equal to one, if job i is the k-th job to be processed, $i, k = 1, 2, \ldots, N$. Let y_{ikt} be a binary decision variable equal to one, if there is a flow of the commodity t in arc (i, k), $(i, k) \in$ A, $t = 1, 2, \ldots, M$. The formulation SSPMF is given by (1)-(16).

$$\text{Min } \; Z_M = \sum_{t=1}^{M} \sum_{i=1}^{N-1} y_{i(N+1)t} + \sum_{t=1}^{M} \sum_{i=1}^{N-2} y_{i(N+2)t} \tag{1}$$

subject to:

$$\sum_{k=1}^{N} x_{ik} = 1, \qquad i \in J, \tag{2}$$

$$\sum_{i=1}^{N} x_{ik} = 1, \qquad k \in J, \tag{3}$$

$$y_{01t} + y_{0(N+2)t} = 1, \qquad t \in T, \tag{4}$$

$$y_{(i-1)it} + y_{(N+2)it} - y_{i(N+1)t} - y_{i(i+1)t} - y_{i(N+2)t} = 0, \qquad t \in T, \; i = 1, \ldots, N-2, \tag{5}$$

$$y_{(N-2)(N-1)t} + y_{(N+2)(N-1)t} - y_{(N-1)Nt} - y_{(N-1)(N+1)t} = 0, \qquad t \in T, \tag{6}$$

$$y_{(N-1)Nt} - y_{N(N+1)t} = 0, \qquad t \in T, \tag{7}$$

$$\sum_{i=1}^{N} y_{i(N+1)t} = 1, \qquad t \in T, \tag{8}$$

$$\sum_{i=0}^{N-2} y_{i(N+2)t} - \sum_{i=1}^{N-1} y_{(N+2)it} = 0, \qquad t \in T, \tag{9}$$

$$x_{ik} \leq y_{(k-1)kt}, \qquad i, k \in J, \; t \in T_i, \tag{10}$$

$$\sum_{t=1}^{M} y_{(k-1)kt} = C, \qquad k \in J, \tag{11}$$

$$x_{ik} \in \{0,1\}, \qquad i,k \in J, \tag{12}$$

$$y_{i(i+1)t} \in \{0,1\}, \qquad i = 0,\dots,N,\ t \in T, \tag{13}$$

$$y_{i(N+1)t} \in \{0,1\}, \qquad i = 1,\dots,N-1,\ t \in T, \tag{14}$$

$$y_{i(N+2)t} \in \{0,1\}, \qquad i = 0,\dots,N-2,\ t \in T, \tag{15}$$

$$y_{(N+2)it} \in \{0,1\}, \qquad i = 1,\dots,N-1,\ t \in T. \tag{16}$$

The objective function (1) minimizes the cost of the flow on arcs $(i, N+2)$, $i = 0,1,\dots,N-2$, and $(i, N+1)$, $i = 1,2,\dots,N-1$. In any feasible solution, the unit flows in these arcs correspond to tools that are taken out of the tool magazine, that is, the tools that had to leave the magazine to make space for other tools inserted in the magazine. Therefore, the flows in these arcs correspond to tool switches. Constraints in (2), (3) and (12) require that all N jobs will be sequenced in some order. Constraints in (4)-(9) are the flow conservation constraints in nodes 0, $1,\dots,N-2$, $N-1$, N, $N+1$, and $N+2$, respectively. Constraints in (10) require that, if job i is the k-th job to be processed, then the flow on arc $(k-1,k)$ must include a unit flow of the commodities corresponding to tools required to process job i. Constraints in (11) require that the flow in arc $(i,i+1)$, $i = 0,1,\dots,N$ is C, or, in other terms, the total number of tools in the magazine is always C. Constraints in (12)-(16) are the integrality conditions on all the decision variables. The number of variables in this model is $N^2 + 4MN - 2M$, whereas the number of the constraints is $N(\sum_{i=1}^{N} |T_i| + M + 3) + 3M$.

In the following theorem, a lower bound for the linear programming relaxation of model SSPMF is established.

Theorem 3.1. *The objective function value of the linear programming relaxation of Model SSPMF is at least $M - C$.*

Proof. By adding up the constraints (8) for $t = 1,\dots,M$, we obtain the equality (17).

$$\sum_{t=1}^{M} \sum_{i=1}^{N} y_{i(N+1)t} = M. \tag{17}$$

Equality (17) can be rewritten as

$$\sum_{t=1}^{M} \sum_{i=1}^{N-1} y_{i(N+1)t} + \sum_{t=1}^{M} y_{N(N+1)t} = M. \tag{18}$$

By Constraints (7) and considering $k = N$ in Constraints (11), we get

$$\sum_{t=1}^{M} \sum_{i=1}^{N-1} y_{i(N+1)t} = M - C. \tag{19}$$

Since all flows in the arcs are non-negative, the result follows. □

$M - C$ is a trivial lower bound for SSP. There is a total of M tools that are all needed for processing the jobs. C is the capacity of the tool magazine; therefore, at most, this number of tools can be in the machine in the beginning and at least $M - C$ tools will be left out. Since all the tools that are left out in the beginning must enter the machine at some moment later on, at least $M - C$ tools will be switched. To the best of our knowledge, all the previous models proposed in the literature provide, in the great majority of the cases, a LP relaxation lower bound smaller than $M - C$.

We observe that a feasible solution to the linear relaxation of model SSPMF can be easily constructed whose optimal value is $M - C$.

4 Computational experiments

Computational experiments were carried out, comparing model SSPMF with models TD, LSS and CGL, suggested in Tang and Denardo (1988a), Laporte, Salazar-Gonzáles, and Semet (2004) and Catanzaro, Gouveia, and Labbé (2015), respectively. Bard's (Bard (1988)) model was not considered in our tests, since its linearized model is exactly TD. We considered the datasets described in Yanasse, Rodrigues and Senne (2009) and Catanzaro, Gouveia, and Labbé (2015). We changed the subtour elimination constraints used in models LSS and CGL (Dantzig, Fulkerson, and Johnson (1954)) by the subtour elimination constraints proposed by Desrochers and Laporte (1991), that are polynomially sized.

These instances can be found at (`https://sites.google.com/site/antoniochaves/publications/data`). Each dataset contains i random instances with the same number of jobs, tools and machine capacity. The values of i are presented in Tables 6 and 7.

We implemented all the models in the syntax of AMPL (Fourer, Gay, and Kernighan (2002)), and the *solver* CPLEX 12.6.3 (ILOG CPLEX (2016)) was used to solve them on an Intel Core i7 3.6 GHz processor with 16GB of RAM in a Windows 10 environment. We performed tests with later versions of the CPLEX, but in all of them we observed a poorer performance. The instances were solved by the *"branch-and-cut"* method with the default setting of CPLEX, considering the parameter `branch` = `1` (i.e., the up branch direction, restricted to higher value, should be taken first at each node).

Additional constraints can be added to model SSPMF to reduce the search space, thereby guaranteeing that the solution obtained is optimal. Based on the symmetry property (Ghiani, Grieco, and Guerriero (2007); Yanasse, Rodrigues and Senne (2009)) of the SSP, we have the following result:

Proposition 4.1. *Given any job p, there is an optimal solution in which p is processed in one of the first $\lceil N/2 \rceil$ positions.*

Proposition 4.1 is valid for any job p. Constraint (20) imposes that this job p has to be processed in one of the first $\lceil N/2 \rceil$ positions. In our computational tests, we choose p such that $p = \arg\max_{j \in J} |T_j|$.

$$\sum_{k=1}^{\lceil N/2 \rceil} x_{pk} = 1. \tag{20}$$

The following equality imposes that a tool t can permanently leave the magazine only after it is used at least the amount of times it is required, i.e. $|J_t|$ times.

$$y_{k(N+1)t} = 0, \qquad t = 1, 2, \ldots, M, \; k = 1, \ldots, |J_t| - 1. \tag{21}$$

In Tables 3, 4, and 5 we present the number of jobs (N), the number of tools (M), the machine capacity (C), and the number of instances (i) in each one of groups (Group) of the instances used in the tests. For each one of the models, we present the average objective value of the associated linear programming (LP) relaxation. We consider that all insertion of a tool in a tool magazine is a switch, as in Catanzaro, Gouveia, and Labbé (2015), and, at the start of operations in the machine, there will be C tools in the machine. Therefore, we add the value of C in the linear relaxation programming values of the models TD, LSS, and SSPMF. We considered subtracting the value C in the linear relaxation programming values of the model CGL; but, in some cases, the resulting value would have been negative, like in instance $B4 - 1$ of Catanzaro, Gouveia, and Labbé (2015), whose LP value is 5 and the C value is 12.

In Tables 6 and 7, we present the number of jobs (N), the number of tools (M), the machine capacity (C), and the number of instances (i) in each one of the groups (Group) of instances used in the tests. For each one of the models, we present the number of instances solved to optimality within the time limit set, 3600 seconds (O), and the average time of the instances solved to optimality (T).

Yanasse, Rodrigues and Senne (2009) proposed 1,350 instances divided into 5 groups (A, B, C, D and E) according to the configurations of N, M, and C. In Tables 3, 4 and 6, we summarize the results obtained for this set of instances. Catanzaro, Gouveia, and Labbé (2015) proposed 160 instances divided into 4 groups (A, B, C and D) according to the configurations of N, M, and C. In Tables 5 and 7 we summarize the results obtained for this set of instances.

4.1 Linear programming relaxation

In Tables 3, 4, and 5 we present the lower bounds given by the linear programming relaxation of models SSPMF, TD, LSS and CGL, for the instances introduced in Yanasse, Rodrigues and Senne (2009) and Catanzaro, Gouveia, and Labbé (2015), respectively.

The results of the tests indicate that the greater the machine capacity, the more the linear relaxation programming value of model LSS approximates the linear relaxation programming value of model TD, becoming equal in some instances and, when the machine's capacity is tightened, the linear relaxation programming value of model LSS is stronger.

The CGL model is based on model LSS hence, it presents the same behavior; when the machine capacity is tight, the linear relaxation value is stronger. However, when machine capacity increases, the linear relaxation programming values are poor, worse than the linear relaxation programming values obtained by model TD in 23 (Table 3, 4) and 6 (Table 5) instance sets. The sign "-" used in the tables indicates that the CPLEX was not able to solve the LP relaxation model within the 1-hour time limit set.

In general, model SSPMF provided the best lower bounds in 62 out of the 66 instance sets.

4.2 Performance of the models

In Tables 6 and 7 we present the results for models SSPMF, TD, LSS, and CGL, for the instances introduced by Yanasse, Rodrigues and Senne (2009) and Catanzaro, Gouveia, and Labbé (2015), respectively.

Groups A and B provided by Yanasse, Rodrigues and Senne (2009) are trivial and CPLEX solved all the instances for all four models, except using model CGL that did not solve one instance of Group B. Of the total of 19 sets of instances of these two groups, model SSPMF presented better average running time in 11 sets, and in the other sets, the average running time was close to the other models that obtained better average running time.

In Table 6 we can see that, out of 340 instances, CPLEX was able to solve 242 instances at optimality for Group C using model SSPMF, while using models TD, LSS and CGL, it was able to solve 168, 3, and 25 instances at optimality, respectively. In class ($N = 15$, $M = 25$, $C = 5$) model CGL had more instances solved than model SSPMF, since model CGL had better lower bounds for the instances of this class (according to Table 3). Considering Group D, the set that contains the largest instances provided by Yanasse, Rodrigues and Senne (2009), CPLEX was able to solve 79, 13, 0, and 1 instances at optimality using models SSPMF, TD, LSS, and CGL, respectively. For Group E of 80 instances, CPLEX was able to solve 73 instances at optimality using model SSPMF, while using models TD, LSS and CGL, it was only able to solve 60, 41, and 33 instances at optimality, respectively.

The sets provided by Catanzaro, Gouveia, and Labbé (2015) produced similar outcomes to those provided by Yanasse, Rodrigues and Senne (2009), as

Table 3: Results of linear relaxation programming values for Groups A, B, and C provided by Yanasse, Rodrigues and Senne (2009).

Group	N	M	C	i	SSPMF	TD	LSS	CGL
					LP	LP	LP	LP
A	8	15	5	10	**15.00**	5.00	6.59	13.74
	8	15	10	30	**15.00**	10.00	10.20	9.89
	8	20	5	10	**20.00**	5.00	7.16	19.02
	8	20	10	30	**20.00**	10.00	10.71	16.42
	8	20	15	60	**20.00**	15.00	15.25	12.22
	8	25	5	10	**25.00**	5.00	7.21	23.03
	8	25	10	30	**25.00**	10.00	11.10	22.02
	8	25	15	60	**25.00**	15.00	15.42	17.83
	8	25	20	100	**25.00**	20.00	20.06	14.03
B	9	15	5	10	**15.00**	5.00	6.19	12.95
	9	15	10	30	**15.00**	10.00	10.45	9.69
	9	20	5	10	**20.00**	5.00	7.09	19.03
	9	20	10	30	**20.00**	10.00	10.91	16.25
	9	20	15	60	**20.00**	15.00	15.24	11.96
	9	25	5	10	**25.00**	5.00	6.99	22.42
	9	25	10	30	**25.00**	10.00	10.88	21.59
	9	25	15	50	**25.00**	15.00	15.60	18.84
	9	25	20	100	**25.00**	20.00	20.10	13.56
C	15	15	5	10	**15.00**	5.00	6.43	14.88
	15	15	10	30	**15.00**	10.00	10.66	8.63
	15	20	5	10	**20.00**	5.00	6.45	18.19
	15	20	10	30	**20.00**	10.00	11.01	15.00
	15	20	15	60	**20.00**	15.00	15.40	10.82
	15	25	5	10	25.00	5.00	7.30	**27.00**
	15	25	10	30	**25.00**	10.00	11.01	20.80
	15	25	15	60	**25.00**	15.00	15.59	15.64
	15	25	20	100	**25.00**	20.00	20.25	11.17

shown in Tables 6 and 7. Of 40 instances in Group A, CPLEX was able to solve 40 instances at optimality using model SSPMF, while using models TD, LSS, and CGL, it was able to solve 40, 40, and 32 instances at optimality, respectively. However, in Group B, of 40 instances, CPLEX was able to solve 38 instances at optimality using model SSPMF, while using the models TD, LSS, and CGL, it was only able to solve 25, 0, and 2 instances at optimality, respectively.

CPLEX was not able to solve at optimality any instance of Groups C and D provided by Catanzaro, Gouveia, and Labbé (2015) within the time limit set, for any of the models.

In all sets of instances, our model produced better results when the machine capacity increased, both in the number of instances solved at optimality and in average running time. In general, of the 1,510 instances tested, CPLEX was able to resolve 1,142 (75.63%), 976 (64.64%), 754 (49.93%), and 762 (50.46%) instances at optimality considering the models SSPMF, TD, LSS, and CGL, respectively.

Table 4: Results of linear relaxation programming values for Groups D and E provided by Yanasse, Rodrigues and Senne (2009).

Group	N	M	C	i	SSPMF	TD	LSS	CGL
					LP	LP	LP	LP
D	20	15	5	10	15.00	5.00	6.65	**17.15**
	20	15	10	20	**15.00**	10.00	10.08	3.49
	20	20	5	10	**20.00**	5.00	6.43	19.44
	20	20	10	10	**20.00**	10.00	10.00	3.55
	20	20	15	30	**20.00**	15.00	15.00	3.93
	20	25	5	10	25.00	5.00	6.65	**25.09**
	20	25	10	10	**25.00**	10.00	10.00	6.30
	20	25	15	40	**25.00**	15.00	15.96	16.36
	20	25	20	40	**25.00**	20.00	20.00	4.96
	25	15	10	10	**15.00**	10.00	10.00	2.10
	25	20	10	10	**20.00**	10.00	10.00	2.45
	25	20	15	10	**20.00**	15.00	15.00	2.55
	25	25	10	10	**25.00**	10.00	10.00	4.45
	25	25	15	10	**25.00**	15.00	15.00	3.25
	25	25	20	30	**25.00**	20.00	20.00	3.98
E	10	10	4	10	10.00	4.00	5.58	**11.41**
	10	10	5	10	**10.00**	5.00	5.00	5.46
	10	10	6	10	**10.00**	6.00	6.00	2.95
	10	10	7	10	**10.00**	7.00	7.00	2.58
	15	20	6	10	**20.00**	6.00	7.46	18.34
	15	20	8	10	**20.00**	8.00	8.00	6.64
	15	20	10	10	**20.00**	10.00	10.00	5.23
	15	20	12	10	**20.00**	12.00	12.00	4.70

Table 5: Results of linear relaxation programming values for the instances introduced by Catanzaro, Gouveia, and Labbé (2015).

Group	N	M	C	i	SSPMF	TD	LSS	CGL
					LP	LP	LP	LP
A	10	10	4	10	**10.00**	4.00	4.89	9.02
	10	10	5	10	**10.00**	5.00	5.00	4.32
	10	10	6	10	**10.00**	6.00	6.00	2.92
	10	10	7	10	**10.00**	7.00	7.00	2.88
B	15	20	6	10	**20.00**	6.00	7.28	17.65
	15	20	8	10	**20.00**	8.00	8.00	5.37
	15	20	10	10	**20.00**	10.00	10.00	3.75
	15	20	12	10	**19.20**	12.00	12.00	3.75
C	30	40	15	10	**40.00**	15.00	17.10	-
	30	40	17	10	**40.00**	17.00	17.00	-
	30	40	20	10	**40.00**	20.00	20.00	-
	30	40	25	10	**40.00**	25.00	25.00	-
D	40	60	20	10	**60.00**	20.00	23.36	-
	40	60	22	10	**60.00**	22.00	22.00	-
	40	60	25	10	**60.00**	25.00	25.00	-
	40	60	30	10	**60.00**	30.00	30.00	-

Table 6: Results for Groups A, B, C, D, and E provided by Yanasse, Rodrigues and Senne (2009).

Group	N	M	C	i	SSPMF		TD		LSS		CGL	
					O	T	O	T	O	T	O	T
	8	15	5	10	**10**	**0.99**	**10**	1.16	**10**	2.74	**10**	**0.99**
	8	15	10	30	**30**	**0.82**	**30**	0.84	**30**	6.70	**30**	15.71
	8	20	5	10	**10**	1.48	**10**	1.91	**10**	4.09	**10**	**0.35**
	8	20	10	30	**30**	1.74	**30**	**1.26**	**30**	10.80	**30**	11.91
A	8	20	15	60	**60**	**1.02**	**60**	2.03	**60**	8.85	**60**	46.17
	8	25	5	10	**10**	0.54	**10**	2.85	**10**	5.32	**10**	**0.25**
	8	25	10	30	**30**	3.01	**30**	**2.18**	**30**	15.31	**30**	6.31
	8	25	15	60	**60**	2.55	**60**	**2.46**	**60**	15.23	**60**	50.93
	8	25	20	100	**100**	**1.21**	**100**	3.22	**100**	11.41	**100**	86.27
	9	15	5	10	**10**	3.11	**10**	**2.14**	**10**	12.15	**10**	4.47
	9	15	10	30	**30**	**1.82**	**30**	3.04	**30**	43.55	**30**	198.17
	9	20	5	10	**10**	2.69	**10**	2.95	**10**	15.48	**10**	**1.47**
	9	20	10	30	**30**	**4.31**	**30**	7.74	**30**	73.53	**30**	145.69
B	9	20	15	60	**60**	**2.47**	**60**	8.75	**60**	63.42	**60**	532.38
	9	25	5	10	**10**	**1.22**	**10**	5.16	**10**	31.11	**10**	1.68
	9	25	10	30	**30**	**5.57**	**30**	6.73	**30**	115.65	**30**	62.43
	9	25	15	50	**50**	**5.78**	**50**	18.80	**50**	113.12	**50**	412.17
	9	25	20	100	**100**	**2.50**	**100**	11.20	**100**	90.03	99	1087.51
	15	15	5	10	**9**	697.26	8	1225.59	1	960.24	2	491.55
	15	15	10	30	**28**	380.14	**28**	840.36	0	-	1	573.47
	15	20	5	10	**9**	1092.87	7	1363.13	0	-	4	1882.45
	15	20	10	30	**20**	735.65	12	1383.68	0	-	2	2101.17
C	15	20	15	60	**48**	754.72	35	1275.42	0	-	4	1779.36
	15	25	5	10	3	1258.49	1	794.52	1	182.76	**8**	657.53
	15	25	10	30	**15**	620.30	8	699.41	1	2379.97	2	1376.73
	15	25	15	60	**30**	977.98	19	1637.43	0	-	2	1952.06
	15	25	20	100	**80**	565.50	50	1212.19	0	-	0	-
	20	15	5	10	0	-	0	-	0	-	**1**	185.74
	20	15	10	20	**3**	**1.65**	**3**	1686.84	0	-	0	-
	20	20	5	10	0	-	0	-	0	-	0	-
	20	20	10	10	0	-	0	-	0	-	0	-
	20	20	15	30	**10**	**0.35**	0	-	0	-	0	-
	20	25	5	10	0	-	0	-	0	-	0	-
	20	25	10	10	0	-	**3**	**512.74**	0	-	0	-
D	20	25	15	40	9	1.27	0	-	0	-	0	-
	20	25	20	40	**17**	2070.18	3	1300.58	0	-	0	-
	25	15	10	10	**7**	1089.98	2	36.89	0	-	0	-
	25	20	10	10	**1**	33.91	0	-	0	-	0	-
	25	20	15	10	0	-	0	-	0	-	0	-
	25	25	10	10	**1**	**55.02**	1	426.92	0	-	0	-
	25	25	15	10	**10**	25.86	0	-	0	-	0	-
	25	25	20	30	**21**	38.08	1	3217.48	0	-	0	-
	10	10	4	10	**10**	3.34	**10**	4.44	**10**	6.62	**10**	**2.43**
	10	10	5	10	**10**	**1.79**	**10**	3.07	**10**	69.31	**10**	263.78
	10	10	6	10	**10**	**0.38**	**10**	1.99	**10**	156.11	8	2328.55
E	10	10	7	10	**10**	**0.07**	**10**	1.06	**10**	271.27	2	3037.55
	15	20	6	10	**8**	901.99	6	1154.33	1	43.42	3	1779.86
	15	20	8	10	**7**	768.90	4	838.22	0	-	0	-
	15	20	10	10	**9**	363.83	6	154.15	0	-	0	-
	15	20	12	10	**9**	180.81	4	1134.79	0	-	0	-

5 Conclusion

In this paper we proposed a new integer linear programming formulation for the Job Sequencing and Tool Switching Problem, using a multicommodity flow

Table 7: Results for Groups A and B provided by Catanzaro, Gouveia, and Labbé (2015).

Group	N	M	C	i	SSPMF		TD		LSS		CGL	
					O	T	O	T	O	T	O	T
A	10	10	4	10	**10**	**2.15**	**10**	3.41	**10**	19.23	**10**	15.15
	10	10	5	10	**10**	**0.95**	**10**	2.64	**10**	75.57	**10**	390.34
	10	10	6	10	**10**	**0.21**	**10**	1.39	**10**	152.39	**10**	2207.79
	10	10	7	10	**10**	**0.07**	**10**	1.12	**10**	233.29	2	2881.08
B	15	20	6	10	**9**	1454.27	6	1376.85	0	-	2	1073.07
	15	20	8	10	**9**	1049.72	8	1665.75	0	-	0	-
	15	20	10	10	**10**	557.97	6	698.36	0	-	0	-
	15	20	12	10	**10**	4.25	5	793.61	0	-	0	-

model. We demonstrated that the lower bound provided by our model coincides with the natural lower bound for this problem, consisting of the difference between the number of tools and the machine capacity.

One of the difficulties in solving the SSP is the symmetry property of its solutions. In the proposed multicommodity flow model, it was easy to propose a constraint that eliminates half of the possible solutions of the problem by requiring that some given job be processed in the first half of the production sequence.

For the computational experiments, we considered 1,510 instances from Yanasse, Rodrigues and Senne (2009) and Catanzaro, Gouveia, and Labbé (2015). We compared our results with the models proposed by Tang and Denardo (1988a), Laporte, Salazar-Gonzáles, and Semet (2004), and Catanzaro, Gouveia, and Labbé (2015).

Our model outperformed the others in number of instances solved at optimality, quality of lower bound, and running time. However, CPLEX was not able to solve large instances for our model, or for the other models.

The higher the capacity of the machine, the better the performance of the proposed model compared to the other models of the literature. The use of our model is therefore recommended particularly for larger values of C.

Future studies could be to develop other valid inequalities and exact methods to solve the proposed formulation, that eliminate or reduce symmetric solutions, thus reducing the search space.

This formulation has an advantage that can be exploited in exact methods and/or matheuristics, since by fixing some variables as integers, others variables automatically become integers. Hence, a relax-and-fix method seems to be a promising approach to pursue. Another alternative is to design some enumerative branch-and-bound method aiming to reduce symmetries of the solutions space searched. For instance, considering ordered subsets of combinations of machine tools that leave the magazine during processing, or, use a similar idea in relation to subset of jobs.

Lagrangean relaxation could also be explored, since by relaxing the linking constraint (10), the problem is decomposed into two subproblems: an assignment problem and a network flow problem. Extensions of the multicommodity flow model to other variations of the SSP can also be studied, for instance, considering parallel machines, non-uniform size of tools, tool sharing between machines. Studies are being conducted in this sense.

Acknowledgment

We are grateful to Dr. Daniele Catanzaro for sending us the code and instances for model CGL from Catanzaro, Gouveia, and Labbé (2015), and we thank the three anonymous reviewers for valuable comments and suggestions that improved the quality of the presentation of this paper.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, FAPESP (grants 2018/15417-8 and 2016/01860-1), and CNPq (grants 423694/2018-9, 303736/2018-6, and 303144/2014-9).

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Ahmadi, E., Goldengorin, B., Suer, G.A., Mosadegh, H. 2018. "A hybrid method of 2-TSP and novel learning-based GA for job sequencing and tool switching problem." *Applied Soft Computing* 65, 214–229.

Agapiou, J. S. 1991. "Sequence of operations optimization in single-stage multifunctional systems." *Journal of Manufacturing Systems* 10(3), 194-208.

Al-Fawzan, M.A., Al-Sultan, K.S. 2002. "A tabu search based algorithm for minimizing the number of tool switches on a flexible machine." *Computers & Industrial Engineering* 44, 35–47.

Amaya, J.E., Cotta, C., Fernández, A.J. 2008. "A Memetic Algorithm for the Tool Switching Problem." In: Blesa M.J. et al. (eds) *Hybrid Metaheuristics* HM 2008. Lecture Notes in Computer Science, 5296. Springer, Berlin, Heidelberg.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J. 2010. "Hybrid Cooperation Models for the Tool Switching Problem." In: González J.R., Pelta D.A., Cruz C., Terrazas G., Krasnogor N. (eds) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, 284. Springer, Berlin, Heidelberg.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J. 2011. "Memetic cooperative models for the tool switching problem." *Memetic Computing* 3(3), 199–216.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J. 2012. "Solving the tool switching problem with memetic algorithms." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 26, 221–235.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J. 2013. "Cross entropy-based memetic algorithms: An application study over the tool switching problem." *International Journal of Computational Intelligence Systems* 6(3), 559–584.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J., García-Sánchez, P. 2019. "Deep memetic models for combinatorial optimization problems: application to the tool switching problem." *Memetic Computing* 1–20.

Bard, J.F. 1988. "A heuristic for minimizing the number of tool switches on a flexible machine." *IIE Transactions* 20:4, 382–391.

Baykasoğlu, A., Ozsoydan, F.B. 2016. "An improved approach for determination of index positions on CNC magazines with cutting tool duplications by integrating shortest path algorithm." *International Journal of Production Research* 54:3, 742–760.

Baykasoğlu, A., Ozsoydan, F.B. 2017. "Minimizing tool switching and indexing times with tool duplications in automatic machines." *The International Journal of Advanced Manufacturing Technology* 89, 1775-–1789.

Baykasoğlu, A., Ozsoydan, F.B. 2018. "Minimisation of non-machining times in operating automatic tool changers of machine tools under dynamic operating conditions." *International Journal of Production Research* 56:4, 1548–1564.

Becceneri, J.C., Yanasse, H.H., Soma, N.Y. 2004. "A method for solving the minimization of the maximum number of open stacks problem within a cutting process." *Computers & Operations Research* 31(14), 2315–2332.

Beezao, A.C. 2016. "O problema de minimização de troca de ferramentas." Phd Thesis, University of São Paulo.

Bentley, J. J. 1992. "Fast Algorithms for Geometric Traveling Salesman Problems." *ORSA Journal on Computing* 4(4), 387-411.

Burger, A.P., Jacobs, C.G., van Vuuren, J.H., Visagie, S.E. 2015. "Scheduling multi-colour print jobs with sequence-dependent setup times." *Journal of Scheduling* 18(2), 131–145.

Calmels, D. 2018 "The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends." *International Journal of Production Research* 57:15–16, 5005–5025.

Catanzaro, D., Gouveia, L., Labbé, M. 2015. "Improved integer linear programming formulations for the job sequencing and tool switching problem." *European Journal of Operational Research* 244, 766–777.

Chaves, A.A., Senne, E.L.F., Yanasse, H.H. 2012. "Uma nova heurística para o problema de minimização de trocas de ferramentas." *Gestão & Produção* 19(1), 17–30.

Chaves, A. A., Lorena, L. A. N., Senne, E. L. F., Resende, M. G. C. 2016. "Hybrid method with CS and BRKGA applied to the minimization of tool switches problem." *Computers & Operations Research* 67, 174–183.

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

Dadashi, H., Moslemi, S., Mirzazadeh, A. 2016 "Optimization of a New Tool Switching Problem in Flexible Manufacturing Systems with a Tool Life by a Genetic Algorithm." *International Journal of Industrial and Manufacturing Systems Engineering* 1(3), 52–58.

Dantzig, G.B., Fulkerson, D.R., Johnson, S.M. 1954. "Solution of a large-scale traveling salesman problem." *Operations Research* 2, 393–410.

Dehaybe, H. 2018. "An Ant Colony System for solving the Job Sequencing and Tool Switching Problem". Dissertation, Louvain School of Management. Université catholique de Louvain.

Desrochers. M., Laporte. G. 1991. "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints." *Operations Research Letters* 10, 27–36.

Djellab, H., Djellab, K., Gourgand, M. 2000. "A new heuristic based on a hypergraph representation for the tool switching problem." *International Journal of Production Economics* 64, 165–176.

Fathi, Y., Barnette, K.W. 2002 "Heuristic procedures for the parallel machine problem with tool switches." *International Journal of Production Research* 40(1), 151–164.

Fourer, R., Gay, D. M.,Kernighan, B. W. 2002. "AMPL - A Modeling Language for Mathematical Programming." Second Edition, Cengage Learning.

Fulkerson, D.R., Gross, D.A. 1965. "Incidence Matrices and Interval Graphs." *Pacific Journal of Mathematics* 15(3), 835 - 855.

Gamila, M.A., Motavalli, S. 2003. "A modeling technique for loading and scheduling problems in FMS." *Robotics and Computer Integrated Manufacturing* 19, 45–54.

Gao, N., Moon, S.K. 2019 "An Integrated Two-Stage Optimization Method for Job-Shop Bottleneck Planning and Scheduling." *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* Macao, 855–859.

Gendreau, M., Hertz, A., e Laporte, G. 1992. "New insertion and postoptimization procedures for the traveling salesman problem." *Operations Research* 40, 1086–1094.

Ghiani, G., Grieco, A., Guerriero, E. 2007. "An exact solution to the TLP problem in an NC machine." *Robotics and Computer-Integrated Manufacturing* 23, 645–649.

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.

Ghrayeb, O.A., Phojanamongkolkij, N., Finch, P.R. (2003) "A mathematical model and heuristic procedure to schedule printed circuit packs on sequencers." *International Journal of Production Research* 41(16), 3849–3860.

Gökgür, B., Hnich, B., Özpeynirci, S. (2018) "Parallel Machine Scheduling with Tool Loading: A Constraint Programming Approach." *International Journal of Production Research* 54, 1–17.

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(8), 689–694.

Hoffman, A.J., Kolen, A.W.J., Sakarovitch, M. 1985. "Totally Balanced and Greedy Matrices." *SIAM Journal on Algebraic and Discrete Methods* 6(4), 721 - 730.

IBM ILOG CPLEX Optimization Studio. CPLEX User's Manual, Copyright, ILOG, 2016.

Jäger, G., Molitor, P. 2008. "Algorithms and Experimental Study for the Traveling Salesman Problem of Second Order." In: Yang B., Du DZ., Wang C.A. (eds) Combinatorial Optimization and Applications. COCOA 2008. Lecture Notes in Computer Science, 5165. Springer, Berlin, Heidelberg.

Karzan, F.K., Azizoglu, M. 2008 "The tool transporter movements problem in flexible manufacturing systems." *International Journal of Production Research* 46(11), 3059–3084.

Kou, L.T. 1977. "Polynomial Complete Consecutive Information Retrieval Problems." *SIAM Journal on Computing* 6, 67 - 75.

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

Matzliach, B., Tzur, M. 1998 "The online tool switching problem with non-uniform tool size." *International Journal of Production Research* 36(12), 3407–3420.

McGeogh, L., Sleator, D. 1991. "A strongly competitive randomized paging algorithm." *Algorithmica* 6, 816–825.

Melnyk, S. A., Ghosh , S., Ragatz , G. L. 1989. "Tooling constraints and shop floor scheduling: a simulation study." *Journal of Operations Management* 8, 69–89.

Möhring, R.H. 1990. "Graph Problems Related to Gate Matrix Layout and PLA Folding." In: Tinhofer G., Mayr E., Noltemeier H., Syslo M.M. (eds) Computational Graph Theory. Computing Supplementum, vol 7. Springer, Vienna.

Oerlemans, A. 1992. "Production planning for flexible manufacturing systems." PhD Thesis, University of Limburg, Maastricht, Limburg, Netherlands.

Paiva, G.S., Carvalho, M.A.M. 2017. "Improved Heuristic Algorithms for the Job Sequencing and Tool Switching Problem." *Computers & Operations Research* 88, 208–219.

Privault, C., Finke, G. 1995. “Modelling a tool switching problem on a single NC-machine.” *Journal of Intelligent Manufacturing* 6, 87–94.

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.

Raduly-Baka, C., Mäkilä, J., Johnsson, M., Nevalainen, O.S. 2018 “Efficient tool loading heuristic for machines with modular feeders.” *Procedia Manufacturing* 17, 968–975.

Rubinstein, R. 1999. “The cross-entropy method for combinatorial and continuous optimization.” *Methodology and Computing in Applied Probability* 1, 127–190.

Rupe, J., Kuo, W. 1997 “Solutions to a modified tool loading problem for a single FMM.” *International Journal of Production Research* 35(8), 2253–2268.

Salonen, K., Raduly-Baka, C., Nevalainen, O.S. 2006. “A note on the tool switching problem of a flexible machine.” *Computers & Industrial Engineering* 50, 458–465.

Salonen, K., Smed, J., Johnsson, M., Nevalainen, O. 2006. “Grouping and sequencing PCB assembly jobs with minimum feeder setups.” *Robotics and Computer Integrated Manufacturing* 22, 297–305.

Senne, E.L.F., Yanasse, H.H. 2009. “Beam Search Algorithms for Minimizing Tool Switches on a Flexible Manufacturing System.” Proceedings of the XI WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering. 1, 68–72, Baltimore, USA. WSEAS Press.

Shirazi, R., Frizelle, G. D. M. 2001. “Minimizing the number of tool switches on a fexible machine: an empirical study.” *International Journal of Production Research* 39:15, 3547–3560.

Smed, J., Johnsson, M., Puranen, M., Leipãlã, T., Nevalainen, O. 1999. “Job grouping in surface mounted component printing.” *Robotics and Computer-Integrated Manufacturing* 15(1), 39–49.

Song, C.Y., Hwang, H. 2002 “Optimal tooling policy for a tool switching problem of a flexible machine with automatic tool transporter.” *International Journal of Production Research* 40(4), 873–883.

Stecke, K.E., Solberg, J.J. 1981. “Loading and control policies for a flexible manufacturing system.” *International Journal of Production Research* 19(5), 481–190.

Stützle, T. 1999. “Iterated local search for the quadratic assignment problem.” Phd Thesis, TU Darmstadt.

Tang, C. S., Denardo, E. V. 1988. “Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches.” *Operational Research* 36(5), 767–777.

Tang, C.S., Denardo, E.V. 1988. “Models Arising from a Flexible Manufacturing Machine, Part II: Minimization of the Number of Switching Instants.” *Operations Research* 36(5), 778–784.

Van Hop, N., Nagarur, N.N. 2004. “The scheduling problem of PCBs for multiple non-identical parallel machines.” *European Journal of Operational Research* 158, 577–594.

Widmer, M. 1991. “Job Shop Scheduling with Tooling Constraints: A Tabu Search Approach.” *The Journal of the Operational Research Society* 42(1), 75–82.

Wolsey, L. A. 1998. “Integer Programming.” John Wiley & Sons.

Yanasse, H.H., Lamosa, M. 2005. “An application of the generalized travelling salesman problem: the minimization of tool switches problem.” In: International Annual Scientific Conference of the German Operations Research Society, Bremen, Germany, 90–100.

Yanasse, H., Lamosa, M. 2006. “On solving the minimization of tool switches problem using graphs.” XII ICIEOM - Fortaleza, CE, Brazil, 1–9.

Yanasse, H. H., Pinto, M. J. 2002. “The minimization of tool switches problem as a network flow problem with side constraints.” Annals of XXXIV Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro, Brasil.

Yanasse, H. H. 2009. “Um novo limitante inferior para o problema de minimização de trocas de ferramentas.” Annals of XLI Simpósio Brasileiro de Pesquisa Operacional, 2841–2848, Porto Seguro, Brasil.

Yanasse, H. H. 1997. “A transformation for solving a pattern sequencing problem in the wood cut industry.” *Pesquisa Operacional* 17(1), 57–70.

Yanasse, H. H., Rodrigues, R. d. C. M., Senne, E. L. F. 2009. “Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas.” *Gestão & Produção* 16 (3), 370–381.

Zeballos, L. 2010. “A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems.” *Robotics and Computer-Integrated Manufacturing* 26 (6), 725–743.

Zhang, J. H., Hinduja , S. 1995. “Determination of the optimum tool set for a given batch of turned components.” *Annals of the CIRP* 44 (1), 445–450.

Zhou, B.H., Xi, L.F., Cao, Y.S. 2005. “A beam-search-based algorithm for the tool switching problem on a flexible machine.” *Int J Adv Manuf Technol* 25, 876–882.