

Job Sequencing and Tool Switching Problem

Simon Vermeir, Greet Vanden Berghe

KU Leuven, Department of Computer Science, CODES & imec, 9000 Gent, Belgium

hello@simonvermeir.com, greet.vandenbergh@cs.kuleuven.be

August 2020

Abstract

Using tools for an extended amount of time without repeated switching is an efficient way to execute tasks or jobs. The Job Sequencing and Tool Switching Problem wants to take advantage of this fact. Frequently machines are capable of handling various jobs that each require their own set of tools. It is not conceivable to store these all at once. Hence switching is required; a time-consuming process. However, it is possible to significantly lower these switching times. By precisely arranging jobs and accommodating tools that are required later. Utilizing a heuristic approach it is possible to solve problems that would otherwise be impracticable to solve. This work introduces an elegant Ruin and Recreate method that allows for the exploration of a broad neighborhood while keeping tabs on the relationships between jobs. A significant speedup in the decoder and local Search procedure is achieved by using unique problem properties. Allowing a wide range of solutions to be explored in an acceptable time. Finally, choosing an appropriate objective function avoids plateaus in the search landscape.

1 Introduction

Frequently machines are capable of handling various jobs that each require their own set of tools. An example would be machining a part to a precise shape using different drills. Tools can be kept inside of a magazine, this way the machine can complete jobs without stopping.

Unfortunately, there is often not enough capacity for all tools of all jobs. Thus, switching is required when transitioning. A considerable amount of time could be wasted, tools may have to be calibrated or could be very heavy.

Choosing a good job sequence may therefore prove beneficial. Moreover, if sufficient capacity is left in the magazine tools could simply be kept in the magazine for later jobs. The goal of this paper is to develop a solution in order to minimize these switches as much as possible.

2 Problem Overview

Known as the "Job Sequencing and Tool Switching Problem" it can be stated formally in the following way: *Given a set of N jobs each requiring a specific set of M tools and a magazine with a certain tool capacity C . Determine the order each job is carried out in and which tools are needed during that job, in order to minimize the number of tool switches needed.*

Many variations and applications exist. Parallel jobs [1], Tool dependent switching cost [2] Although these variations are interesting, the focus here is on the uniform variant: *Tools are uniform in size and can be put anywhere in the magazine, switching time is constant, one machine is used and all tools of one job can fit in the magazine at once..*

2.1 Formulation

Let $J = \{1, 2, \dots, N\}$ be the set of jobs, $T = \{1, 2, \dots, M\}$ the set of all tools and T_j the set of tools required for a job where $j \in J, T_j \subseteq T$. The magazine size is C and $C \leq \max(|T_j|), j \in J$ [3].

Table 1: SSP instance example 1

Jobs	0	1	2	3	4
Tools	0	0	2	1	0
	1	2	3	2	3
	3		4	4	5
Magazine size = 3					

Example from **Paiva Et. al** [4]

An instance can be modeled by way of a $M \times N$ binary matrix A , with $a_{ij} = 1$ if job j requires tool i , 0 otherwise. [3]. Thus, each column j can be seen as T_j . As can be seen in table 2

Table 2: Example 1: binary matrix A

	Jobs	0	1	2	3	4
Tools	0	1	1	0	0	1
	1	1	0	0	1	0
	2	0	1	1	1	0
	3	1	0	1	0	1
	4	0	0	1	1	0
	5	0	0	0	0	1

The solution to the problem is some sequence σ which can be seen as a permutation of the set of jobs J . And a matrix A^* , with $a_{ij}^* = 1$ if job j requires or keeps tool i , 0 otherwise.

Alternatively, by permutating the columns of A^* according to the sequence σ , the solution can be modeled as a $M \times N$ matrix P . This provides an intuitive way to interpret the state of the magazine at any time, as can be seen in figure 1[3].

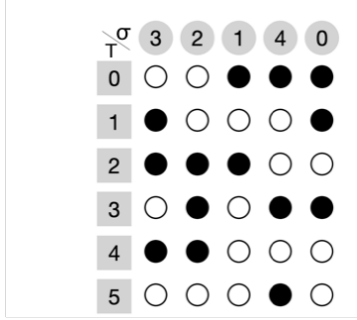


Figure 1: Example 1: Permutation matrix P

Finally a tool setup can be considered as the insertion of a tool in the magazine. A tool switch is the combination of a tool setup and tool removal [3]. The objective is to minimize the number of tool switches.

2.2 Properties

Tang and Denardo formally introduced the problem for the first time. They separated the problem in two subproblems:

- The sequencing problem (SP): Determine the order in which the jobs have to be executed. The sequencing problem is $\mathcal{NP}$ hard. As shown by [3]
- The tooling problem (TP): Given the job sequence, determine which tools are needed at every position. Solvable in *Polynomial* time. As shown by [5]

Although the Sequencing Problem is hard to solve, they demonstrated that the tooling problem is trivial. They introduced the "Keep Tool Needest Soonest" policy or "KTNS", which states the following:

- Tools are only inserted when they are needed.
- When tools need to be removed to accommodate new tools, remove the ones needed the least soon. i.e. "keep the tools needed soonest"

Applying this policy ensures that every job is able to use its required tools, additionally it uses the left over space in the magazine to accommodate tools that are required later. Beyond that, they proved that given a sequence σ (SP), the *optimal* tool configuration (TP) can be found by using KTNS. *This important fact enables us to model the problem as finding the optimal job sequence. Once a sequence is found the tool switches can be reduced by applying KTNS.*

Moreover, by applying this policy the number of tool switches can easily be recognized each time a one turns into a zero in every row within P . Which is equivalent to replacing a tool by another one. [6].

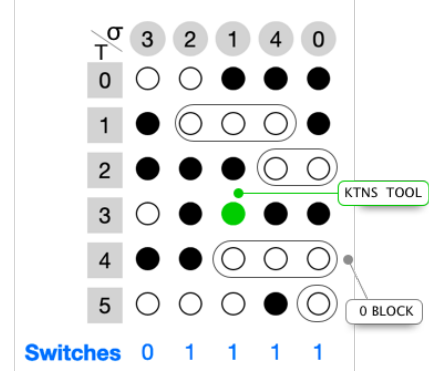


Figure 2: Example 1: Permutation matrix with KTNS applied

Additionally **Tang and Denaro** proposed modeling the SSP as an instance of the Traveling Salesman Problem. They defined a weighted undirected graph $G = (V, E, lb)$, where the vertices V represent the set of jobs J , and the edges $\{i, j\} \in E$ represent job j following job i . Finally the weight of an edge $\{i, j\}$ is $lb(i, j) = \max(|T_i \cup T_j| - C, 0)$, a lower bound for the amount of switches between job i and j [3].

Crama Et al. [3] put forward the idea of recognizing 0/1 blocks within the problem, as can be seen in figure 2. Interestingly the number of tool switches can only be reduced when we are able to eliminate a complete 0 block. This means we can reformulate the objective as reducing the amount of 0 blocks. Additionally, they reformulated KTNS as filling in the maximum amount of complete 0 blocks while respecting the constraints earlier mentioned.

Moreover they noticed that solving the problem as a TSP using the Lower bound proposed by **Tang and Denaro** does not work well when considering a sparse matrix. Due to the fact that a lot of tools can be added by KTNS.

Ghiani Et al. [7] recognized a symmetry property within the problem. They stated and proved that thanks to KTNS the inverse of every job sequence has the same amount of tool switches. Which can be used to reduce the search space.

Ultimately, a lot of attempts at the SSP have been made. Yet the difficulty of the SSP limits the ability of exact methods to be used on instances with more than 25 jobs [8]. This illustrates the fact that a large portion of the research has put a focus on heuristic methods recently.

3 Methodology

3.1 Local Search

Local search is used to solve the SSP. Figure ?? illustrates the mechanism used. A solution to an instance of the SSP is represented by $S = \{\sigma, A^*\}$. The constructive by **Paiva Et al.** [4] is used.

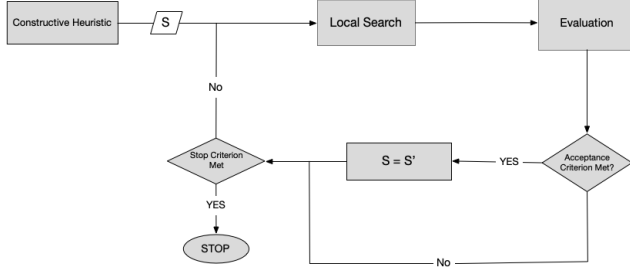


Figure 3: Local search method

3.2 Decoder

An indirect representation of the problem is used [6]. When a solution is changed to a neighbor solution it is changed by permutating its job sequence. The full solution can be computed afterwards by performing $KTNS(\sigma)$ on the job sequence. Afterwards the solution is evaluated by counting the number of switches. We call the combination of these two steps the decoding of our solution. Algorithm 1 gives the algorithm used to perform KTNS. The developed algorithm does not use recursion as opposed to the algorithm proposed by **Tang and Denardo** [5].

4 Ruin and Recreate

A new method is proposed for computing a candidate solution S' .

This method is based on **Slack Induction by String Removals for Vehicle Routing Problems** by [9]. Ruin and recreate works by using one or more operators to destroy parts of a solution, and recreating it into a new candidate solution. The advantage of this method is that a large neighborhood can be explored. In our case the sequence σ gets destroyed by removing certain jobs and recreated by inserting them again in a different place.

The method proposed here is called SMRI and uses 4 distinct phases: select, match, remove and insert. The method developed is visually illustrated on the next page step by step.

Algorithm 1: KTNS

Input: New sequence σ'

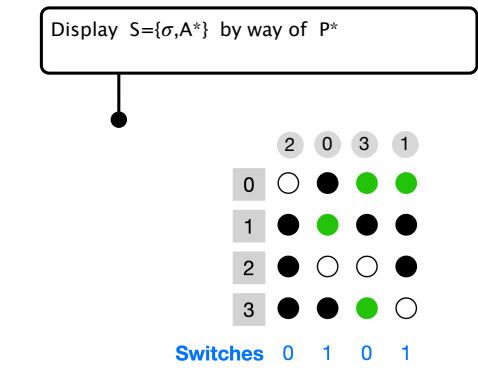
Output: New Solution $S' = \{\sigma', A'^*\}$

```

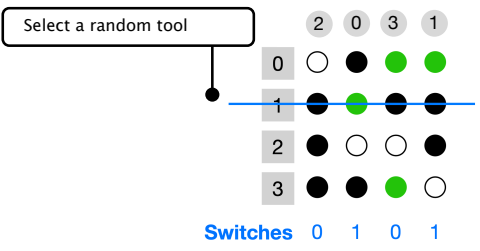
1 function KTNS( $\sigma'$ )
2    $A'^* = A^*$ 
3    $ktnsId = ktnsId + 1$ 
4    $prevJob \leftarrow job - 1$ 
5    $nextJob \leftarrow job + 1$ 
6   if job not first in  $\sigma$  then
7      $nToolsAdd \leftarrow \max(0, |T_{job}| - C)$ 
8      $diff : LinkedList \leftarrow T_{prevJob}^* \setminus T_{job}$ 
9     while  $nToolsAdd > 0$  &  $diff.length > 0$  &
        $nextJob < N$  do
10      for tool  $\in diff$  do
11        if tool  $\in T_{nextJob}$  then
12           $A_{tool, job} \leftarrow ktnsId$ 
13           $diff.removeTool(tool)$ 
14           $nToolsAdd \leftarrow nToolsAdd - 1$ 
15           $nextJob \leftarrow nextJob + 1$ 
16      for tool  $\in diff$  do
17         $A_{tool, job} \leftarrow ktnsId$ 
18         $diff.removeTool(tool)$ 
19         $nToolsAdd \leftarrow nToolsAdd - 1$ 
20        if  $nToolsAdd == 0$  &  $diff.length == 0$ 
          then
21          break
22    $S' = \{\sigma', A'^*\}$ 
23   return  $S'$ 

```

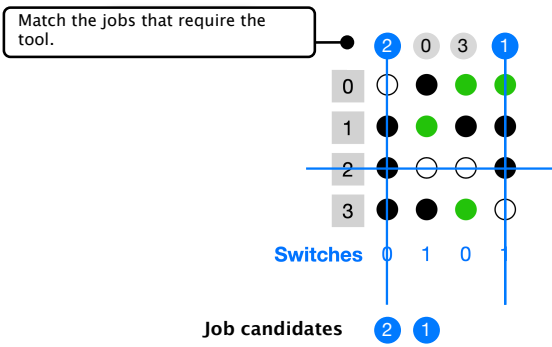
START



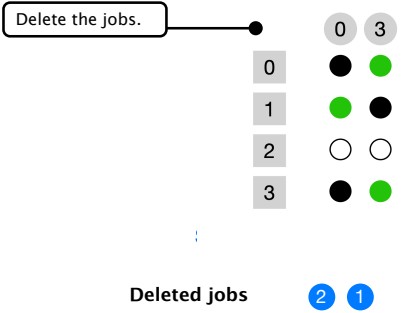
PHASE 1
SELECT



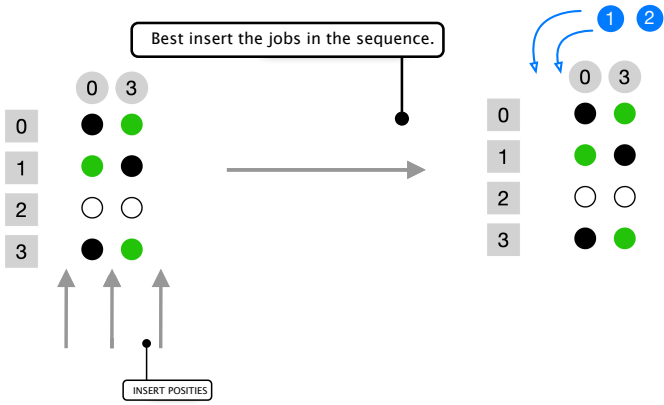
PHASE 2
MATCH



PHASE 3
REMOVE

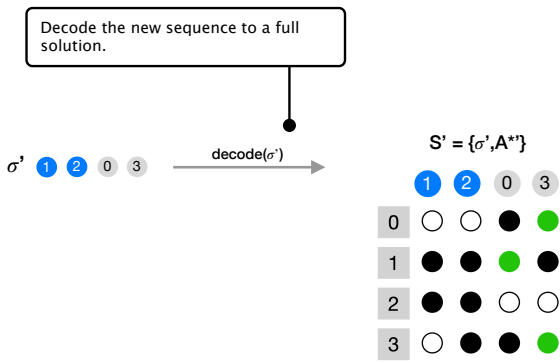


PHASE 4
INSERT



END

decode(σ')



4.1 Meta Heuristic

Kirkpatrick Et. Al. Kirkpatrick1983 introduced Simulated Annealing, a meta heuristic based on the static model of Annealing.

Annealing or annealing is a process used in the metal industry. It tries to get irregularities out of steel.

The systematic approach by **Christiaens and Vanden Berghe [9]** from **Slack Induction by String Removals for Vehicle Routing Problems** has been applied for the implementation of Simulated Annealing. Adjustments were made as needed for the application to this problem. The algortihm used is provided in 2.

Algorithm 2: Simulated Annealing

```

Input: Intital solution  $S = \{\sigma, A^*\}$ 
Input: Best solution  $S_{best} = \{\sigma, A^*\}$ 
1 function SimulatedAnnealing( $S$ )
2   for  $f$  iterations do
3      $\sigma' = \text{ruinAndRecreate}(S)$ 
4      $S' = \text{decode}(\sigma')$ 
5     if
6        $\text{cost}(S) < \text{cost}(S') - \mathcal{T} \ln(\text{random}(0,1))$ 
7       then
8          $S \leftarrow S'$ 
9       if  $n\text{Switches}(S') < n\text{Switches}(S_{best})$ 
10        then
11           $S_{best} \leftarrow S'$ 
12        if  $S' == S_{best}$ , for  $n$  iterations then
13          return  $S_{best}$ 
14         $\mathcal{T} \leftarrow \mathcal{T} \cdot c$ 
15 return  $S_{best}$ 

```

5 Results

Our new method is compared against the state of the art, using the dataset provided by **Catanzaro Et al.**. The results can be found in table 3 , last page. Our results (SMRI) are compared against the results of **Paiva Et al.**[4] and **Mecler Et al.**[6]. The results where obtained from **Mecler Et al.**[6]. DIF stands for the percentage difference between the best solution and our solution, negative means ours is worse performing.

6 Conclusion

A new ruin and recreate method was introduced. As a ruin step, a tool is selected stochastically. Then all jobs requiring that tool are deleted. The logic here: these jobs are related. As a recreate step they are placed in the best place according to a greedy insert. Our methode did not exceed state of the art results. However, it was clear that the method proposed here has potential. It should therefore be further investigated.

References

- [1] A. C. Beezão, J. F. Cordeau, G. Laporte, and H. H. Yanasse, "Scheduling identical parallel machines with tooling constraints," *European Journal of Operational Research*, volume 257, number 3, pages 834–844, 2017, ISSN: 03772217. DOI: 10.1016/j.ejor.2016.08.008.
- [2] D. Calmels, "The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends," *International Journal of Production Research*, volume 57, number 15-16, pages 5005–5025, Aug. 2019, ISSN: 0020-7543. DOI: 10.1080/00207543.2018.1505057. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/00207543.2018.1505057>.
- [3] Y. Crama, A. W. Kolen, A. G. Oerlemans, and F. C. Spieksma, "Minimizing the number of tool switches on a flexible machine," *International Journal of Flexible Manufacturing Systems*, volume 6, number 1, pages 33–54, 1994, ISSN: 09206299. DOI: 10.1007/BF01324874.
- [4] G. S. Paiva and M. A. M. Carvalho, "Improved heuristic algorithms for the Job Sequencing and Tool Switching Problem," *Computers and Operations Research*, volume 88, pages 208–219, 2017, ISSN: 03050548. DOI: 10.1016/j.cor.2017.07.013.
- [5] C. S. Tang, "Models Arising from a Flexible Manufacturing Machine , Part I : Minimization of the Number of Tool Switches.," volume 36, number 5, pages 767–777, 1988.
- [6] J. Mecler, A. Subramanian, and T. Vidal, "A simple and effective hybrid genetic search for the job sequencing and tool switching problem," 2019. arXiv: 1910.10021. [Online]. Available: <http://arxiv.org/abs/1910.10021>.
- [7] G. Ghiani, A. Grieco, and E. Guerriero, "An exact solution to the TLP problem in an NC machine," *Robotics and Computer-Integrated Manufacturing*, volume 23, number 6, pages 645–649, 2007, ISSN: 07365845. DOI: 10.1016/j.rcim.2007.02.011.
- [8] D. Catanzaro, L. Gouveia, and M. Labbé, "Improved integer linear programming formulations for the job Sequencing and tool Switching Problem," *European Journal of Operational Research*, volume 244, number 3, pages 766–777, Aug. 2015. DOI: 10.1016/j.ejor.2015.02.018.
- [9] J. Christiaens and G. Vanden Berghe, "Slack Induction by String Removals for Vehicle Routing Problems," *Transportation Science*, number April, 2020, ISSN: 0041-1655. DOI: 10.1287/trsc.2019.0914.

N	M	C	i	SMRI $\mathcal{S}_{best}$	SMRI $\mathcal{S}_{avg}$	SMRI t	ILS $\mathcal{S}_{best}$	ILS $\mathcal{S}_{avg}$	ILS t	HGS $\mathcal{S}_{best}$	HGS $\mathcal{S}_{avg}$	HGS t	DIF
10	10	7	10	3.0	3.0	3.48	3.0	3.0	0.04	3.0	3.0	0.05	0.0
10	10	6	10	4.1	4.1	4.75	4.1	4.1	0.05	4.1	4.1	0.06	0.0
10	10	5	10	5.8	5.8	5.34	5.8	5.8	0.09	5.8	5.8	0.06	0.0
15	20	12	10	7.2	7.2	15.9	7.2	7.2	0.69	7.2	7.2	0.6	0.0
10	10	4	10	8.5	8.5	6.59	8.5	8.5	0.09	8.5	8.5	0.07	0.0
15	20	10	10	9.7	9.7	29.65	9.7	9.7	1.09	9.7	9.7	0.69	0.0
15	20	8	10	13.7	13.7	35.43	13.7	13.7	1.34	13.7	13.7	0.79	0.0
15	20	6	10	20.5	20.5	33.6	20.5	20.5	1.06	20.5	20.5	0.88	0.0
30	40	25	10	27.8	27.83	300.47	26.3	26.47	133.27	26.2	26.32	48.01	-6.11
30	40	20	10	48.5	48.6	285.79	46.6	46.78	167.41	46.6	46.66	60.58	-4.08
30	40	17	10	66.8	67.03	319.76	65.5	65.82	132.97	65.4	65.43	71.77	-2.09
30	40	15	10	85.1	85.1	291.06	83.9	84.09	96.49	83.5	83.5	74.71	-1.92
40	60	30	10	91.7	91.93	250.37	85.2	85.42	1473.24	84.3	84.66	484.14	-8.32
40	60	25	10	127.5	127.77	300.5	121.2	121.68	1037.2	120.3	120.61	466.12	-5.71
40	60	22	10	156.6	156.7	311.87	151.6	152.05	684.95	150.3	150.45	437.85	-4.09
40	60	20	10	181.6	181.9	333.92	178.0	178.36	474.0	176.5	176.71	501.21	-2.77
10				53.6	53.7	158.03	51.9	52.1	262.75	51.6	51.7	134.22	-4.39

Table 3: Results for the Catanzaro dataset