IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

# A note on “Self-adaptive General Variable Neighborhood Search algorithm for parallel machine scheduling with unrelated servers”

**This is a pre print version of the following article:**

*Original Citation:*

*Availability:*

This version is available http://hdl.handle.net/2318/2068943 since 2025-04-23T16:39:49Z

*Published version:*

DOI:10.1016/j.cor.2025.107055

*Terms of use:*

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

09 November 2026

# A note on "Self-adaptive General Variable Neighborhood Search algorithm for parallel machine scheduling with unrelated servers"

Andrea Grosso[1] Fabio Salassa[2]
[1]Dipartimento di Matematica
Università di Torino
[2]Dipartimento di Ingegneria Gestionale e della Produzione
Politecnico di Torino

**Abstract**

In a recent paper a two-machines two-servers scheduling problem with identical machines but unrelated servers and makespan objective is studied and solved via sophisticated Variable-Neighbourhood Search procedures, for instances up to 120 jobs in size. We show that the same problem can be transformed to a pretty standard problem with unrelated machines that can be efficiently solved to optimality, up to much larger instances.

## 1 Introduction

In a very recent paper Krimi and Benmansour [2] present a variant of a parallel machines scheduling problem with unrelated servers. In Section 2.1 of [2], the authors define the problem as "*The problem of scheduling two identical parallel machines with two unrelated servers* $P2, S2|s_{j1}, s_{j2}|C_{\max}$ *can be formally defined as follows. We have a set* $N$ *of* $n$ *non-preemptive jobs that must be executed on two identical parallel machines, denoted as* $M = \{1, 2\}$*. Each job* $i$ *is composed of two operations: the first one is the setup operation (anesthesia), and the second one is the processing operation (surgery). An anesthetist conducts the setup operation, while the second one requires the presence of a surgeon who will conduct the surgical operation. The anesthetist should be in the operating room during the surgery to manage the patient's vital functions. Hence the processing operation requires the two resources simultaneously.*". Authors also claim that this problem has unique characteristics, namely the fact that "*In this problem, machines may not necessarily be occupied during the setup time phase. However, during the processing time phase, the machine and the server engaged in job execution should be occupied*" [2]. In [2], two MILP models and a VNS-based heuristic are proposed in order to solve problem instances up to 25 jobs with MILP models (though not all instances to optimality) and 120 jobs

thanks to the approximate approach. The VNS method is especially appealing since it is coupled with AI-based procedures, but it is unfortunate that the tackled problem can be reduced to a simpler problem. It is worth, in any case, to investigate the use of such hybrid approaches to other scheduling problems.

Finally, authors declare that, to the best of their knowledge, this problem has received no attention in the literature: "*To the best of our knowledge, no attention has been given to a more generic problem in which several servers can be used to load/prepare jobs to be scheduled on two parallel machines.*" [2].

With this note we prove that the problem of [2] reduces to the very well-known $R2||C_{\max}$ problem and, more generally, all problems $Pm, Sm|s_{jk}|C_{\max}$ (where $k$ indexes servers), where both the server and the machine are occupied during job execution, reduce to $Rm||C_{\max}$. Also with a standard MILP model, derived from [3] we were able to solve to optimality problems instances like those of [2] with up to 100000 jobs in approximately 1 second.

## 2 Reduction of the problem

In [2] the focus is kept on the problem with two identical machines and unrelated servers. The following proposition offers a dominance property that applies to any number $m$ of machines and servers, so we deal with the slightly more general problem $Pm, Sm|s_{jk}|C_{\max}$.

**Proposition 1.** *For problem $Pm, Sm|s_{jk}|C_{\max}$ there exists an optimal schedule where each server $k$ works exclusively with machine $k$, $k = 1, 2, \ldots, m$.*

*Proof.* We prove that for any feasible schedule $\sigma$, there exists a schedule $\sigma'$ satisfying the thesis with objective $C_{\max}(\sigma') \leq C_{\max}(\sigma)$. Any feasible schedule $\sigma$ induces a partition of jobs into disjoint sets $B_1, B_2, \ldots, B_m$ such that $B_k$ contains all jobs processed by server $k$. Observe that, since the server is kept busy also during the job processing, no server $k$ can finish its work before time $\bar{p}(B_k) = \sum_{j \in B_k}(s_{jk} + p_j)$, hence $C_{\max}(\sigma) \geq \max\{\bar{p}(B_1), \ldots, \bar{p}(B_m)\}$. Build a new schedule $\sigma'$ assigning to each machine $k$ exactly the jobs in $B_k$ without changing the assignments of jobs to servers; it is easily seen that in such a schedule each server can work continuously on its own set of jobs and

$$C_{\max}(\sigma') = \max\{\bar{p}(B_1), \ldots, \bar{p}(B_m)\} \leq C_{\max}(\sigma).$$

This proves the proposition. □

As an example, consider the following 2-machines, 10-jobs instance proposed in [2].

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 21 | 12 | 28 | 26 | 24 | 10 | 28 | 25 | 10 | 15 |
| $s_{i1}$ | 2 | 7 | 7 | 2 | 4 | 3 | 6 | 4 | 5 | 8 |
| $s_{i2}$ | 3 | 8 | 2 | 10 | 8 | 9 | 6 | 10 | 7 | 6 |

Referring to Figure 1, schedule (a) is the solution proposed in [2]. By applying the principle of the above proof we see that $B_1 = \{1, 4, 5, 6, 8\}$ and $B_2 = \{2, 3, 7, 9, 10\}$ and the schedule can be reorganized as in (b) without worsening the makespan. The net effect of the reorganization is that machine 1 (resp. 2) is completely paired with server 1 (resp. 2), so the execution of jobs like 4 and 5, that are loaded by server 1 but processed on different machines, are now moved on the same machine without worsening the solution. Note that the machine-server pairs $(M_1, S_1)$ and $(M_2, S_2)$ evidently work as a couple of unrelated parallel machines where each job is given the equivalent processing time $\bar{p}_{ik} = (s_{ik} + p_i)$. Note also that the ordering of the jobs on each machine-server pair becomes immaterial.

In view of the above observations, the following holds.

**Corollary 1.** *Problem* $Pm, Sm|s_{jk}|C_{\max}$ *reduces to* $Rm||C_{\max}$.

# 3 MILP models

In [2] sophisticated heuristics are developed for solving $P2, S2|s_{j1}, s_{j2}|C_{\max}$. Because of Proposition 1 and its corollary, we suggest that for this specific case things can be simpler, saving the more sophisticated heuristics for more general cases, for example with unrelated machines (not only servers). Each instance of $Pm, Sm|s_{jk}|C_{\max}$ can be easily modeled as an instance of $Rm||C_{\max}$, taking full advantage of the structure of such problem. Let $\bar{p}_{ik} = s_{ik} + p_i$ the processing time required by job $i$ in order to be processed by the machine-server pair $(M_k, S_k)$; then the optimal solution is computed by solving the following (well-known) MILP model — see for example [3]. Binary variables $x_{ik}$ are set to 1 iff job $i$ is processed by unrelated machine $k$, i.e. machine-server pair $(M_k, S_k)$ handles job $i$.

$$\text{minimize } C_{\max} \tag{1}$$

$$\text{subject to } \sum_{k=1}^{m} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{2}$$

$$\sum_{i=1}^{n} \bar{p}_{ik} x_{ik} \leq C_{\max} \qquad k = 1, \ldots, m \tag{3}$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, \ldots, n,\ k = 1, \ldots, m \tag{4}$$

$$C_{\max} \geq 0. \tag{5}$$

Constraints (2) force each job to be assigned to exactly one unrelated parallel machine, bottleneck constraints (3) bound the variable $C_{\max}$ to the makespan.

For the special case with two machines only, model (1)–(5) specializes to the following, where binary variables $x_i$ are set to 1 iff machine 1 handles job $i$, and $x_i = 0$ if the same job is handled by machine 2.

$$\text{minimize } C_{\max} \tag{6}$$

$$\text{subject to } \sum_{i=1}^{n} \bar{p}_{i1} x_i \leq C_{\max} \tag{7}$$

$$\sum_{i=1}^{n} \bar{p}_{i2}(1 - x_i) \leq C_{\max} \tag{8}$$

$$x_i \in \{0, 1\} \qquad i = 1, ..., n \tag{9}$$

$$C_{\max} \geq 0 \tag{10}$$

The literature about unrelated parallel machines is rich. Aside from the above models it is known (see [1]) that $Rm||C_{\max}$ is solvable in pseudopolynomial time $O(nmT^{m-1})$ via dynamic programming, where $T = \sum_{i=1}^{n} \min_k \bar{p}_{jk}$.

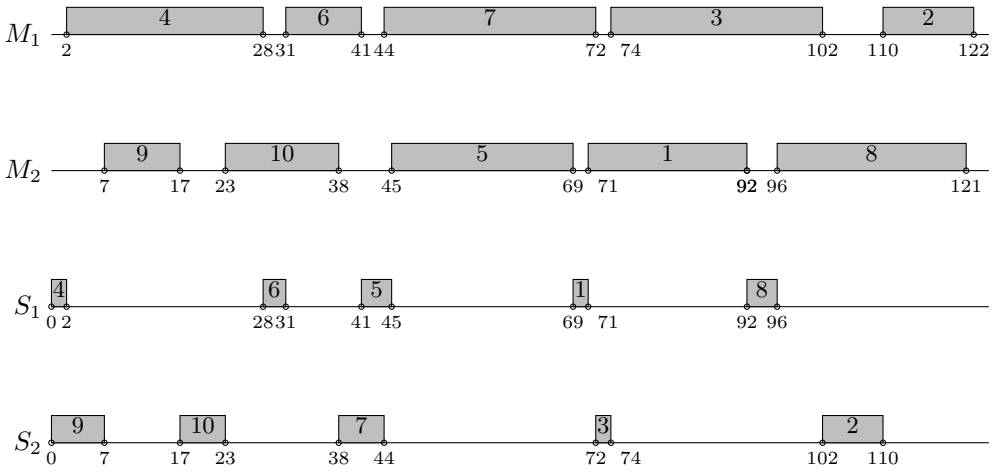
From a practical point of view, as far as the processing and setup times distributions considered in [2] are concerned, the two-machines problem is well solvable. Model (6)–(10) is able to solve very large instances up to 100000 jobs of $P2, S_2|s_{j1}, s_{j2}|\ C_{\max}$ on a laptop Intel Core i7 @ 2.20 GHz machine with 16 GB of RAM using GUROBI 10.0.3 as MILP solver with all default settings except the optimality tolerance that was set to 0.0. The test we performed were done according to the instance generation scheme of [2] where $p_i$ are extracted from a uniform distribution in $[10, 40]$ and $s_{ik}$ are extracted again from a uniform distribution in $[2, 10]$. On a batch of 20 randomly generated instances all with 100000 jobs, GUROBI never exceeded 1 second of CPU time for delivering the optimal solution.

# References

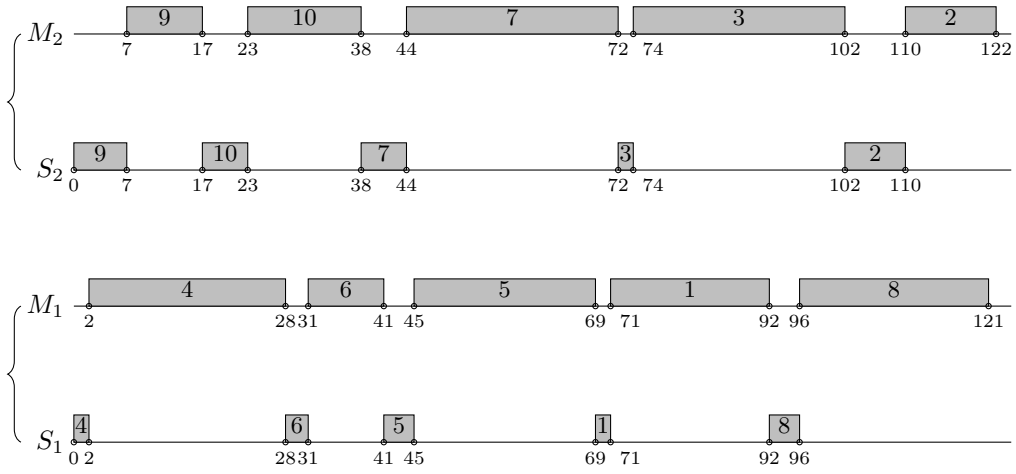

[1] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23:317–327, 1976.

[2] I. Krimi and R. Benmansour. Self-adaptive general variable neighborhood search algorithm for parallel machine scheduling with unrelated servers. *Computers & Operations Research*, 163:106480, 2024.

[3] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.




(a)




(b)

Figure 1: Illustration of Proposition 1 and Corollary 1. Two machines $M_1, M_2$ work with two servers $S_1, S_2$. The schedule (a) can be transformed into schedule (b) where server $S_1$ is paired with machine $M_1$ and server $S_2$ is paired with machine $M_2$. Schedule (b) is not worse than schedule (a), and the two combined pairs $(S_1, M_1)$, $(S_2, M_2)$ work like two parallel unrelated machines.