Expert Systems with Applications 40 (2013) 3328–3340

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

# Scheduling flow lines with buffers by ant colony digraph

Andrea Rossi, Michele Lanzetta *

Department of Civil and Industrial Engineering, University of Pisa, Italy

ARTICLE INFO

*Keywords:*
Flexible manufacturing systems (FMS)
Native non-permutation flowshop scheduling (NPFS)
Metaheuristics
Swarm systems
Ant colony system (ACS)
Benchmark problems

ABSTRACT

This work starts from modeling the scheduling of $n$ jobs on $m$ machines/stages as flowshop with buffers in manufacturing. A mixed-integer linear programing model is presented, showing that buffers of size $n-2$ allow permuting sequences of jobs between stages. This model is addressed in the literature as non-permutation flowshop scheduling (NPFS) and is described in this article by a disjunctive graph (digraph) with the purpose of designing specialized heuristic and metaheuristics algorithms for the NPFS problem. Ant colony optimization (ACO) with the biologically inspired mechanisms of learned desirability and pheromone rule is shown to produce *natively* eligible schedules, as opposed to most metaheuristics approaches, which improve permutation solutions found by other heuristics. The proposed ACO has been critically compared and assessed by computation experiments over existing native approaches. Most makespan upper bounds of the established benchmark problems from Taillard (1993) and Demirkol, Mehta, and Uzsoy (1998) with up to 500 jobs on 20 machines have been improved by the proposed ACO.

## 1. Introduction

A flow line is a conventional manufacturing system where all jobs must be processed on all machines with the same operation sequence (Fig. 1). Examples of flow lines include transfer lines, assembly lines, chemical plants, logistics, and many more (Rossi, Puppato, & Lanzetta, 2012). The flowshop scheduling problem occurs whenever it is necessary to schedule a set of $n$ jobs on $m$ machines so that each job visits all machines in the same order to optimize one or more objective functions. The job sequence of each machine remains unchanged in a permutation flowshop (PFS) scheduling, while in a non-permutation flowshop (NPFS) the sequence of jobs can be different on subsequent machines.

The impact of buffers, along with machines, products and operations, on reconfiguration and performance evaluation is examined by Colledani and Tolio (2005).

In permutation flowshop no buffers are present. To achieve a feasible PFS schedule either the *blocking* or the *no-wait* condition should be applied. In the former case, a job completed on one machine may block that machine until the next downstream machine is free while in the latter the next machine must be available before a job leaves the previous one.

Examples of flow line, which restrict feasible schedules to PFS schedules are factories with conveyors between machines for material transfer and assembly lines performing the final assembly of bulky products or rigid transfer lines.

In non-permutation flowshop scheduling systems, interoperational or intermediate buffers between two stages are necessary and job passing is allowed (Fig. 1). Buffers can be shared in the form of an automatic warehouse or an open space. In most manufacturing applications buffers are present, consequently NPFS has a wider interest.

PFS scheduling is an NP-complete combinatorial optimization problem already for $m = 3$ machines (Garey, Johnson, & Sethi 1976). The computation complexity of permutation approaches is much lower compared with its non-permutation counterpart: there are $n!$ different schedules for ordering jobs on machines in PFS, whilst the number of NPFS schedules increases to $(n!)^m$. To reduce the computation time, permutation flowshop approaches are often preferred by the shop-floor manager, often applying heuristic methods to generate good solutions for practical purposes. PFS scheduler is easier to implement but unfortunately, this simplicity is bought at the price of drastically inferior schedules.

Using the Graham's notation described by a triplet $\alpha|\beta|\gamma$, where field $\alpha$ denotes the system layout and the production flow type, field β indicates the operation characteristics and field $\gamma$ denotes the adopted performance indices. The problem considered here is $F_m|B_i = n-2|C_{\max}$, where $F_m$ stands for flowshop with $m$ machines, $B_i$ denotes the presence of buffers; a buffer of unlimited capacity ($B_i = +\infty$) allows permutations, and consequently implies that the NPFS model is applied; a limited capacity $B_i = n-2$ is sufficient to fulfill the previous requirement as it will be proven in Section 3 along with a mixed-integer linear programing model; $C_{\max}$ denotes the makespan minimization as the optimization criterion. Lower makespan implies other benefits, such as lower idle time, higher machine utilization and higher efficiency.

* Corresponding author.
*E-mail address:* lanzetta@unipi.it (M. Lanzetta).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.041

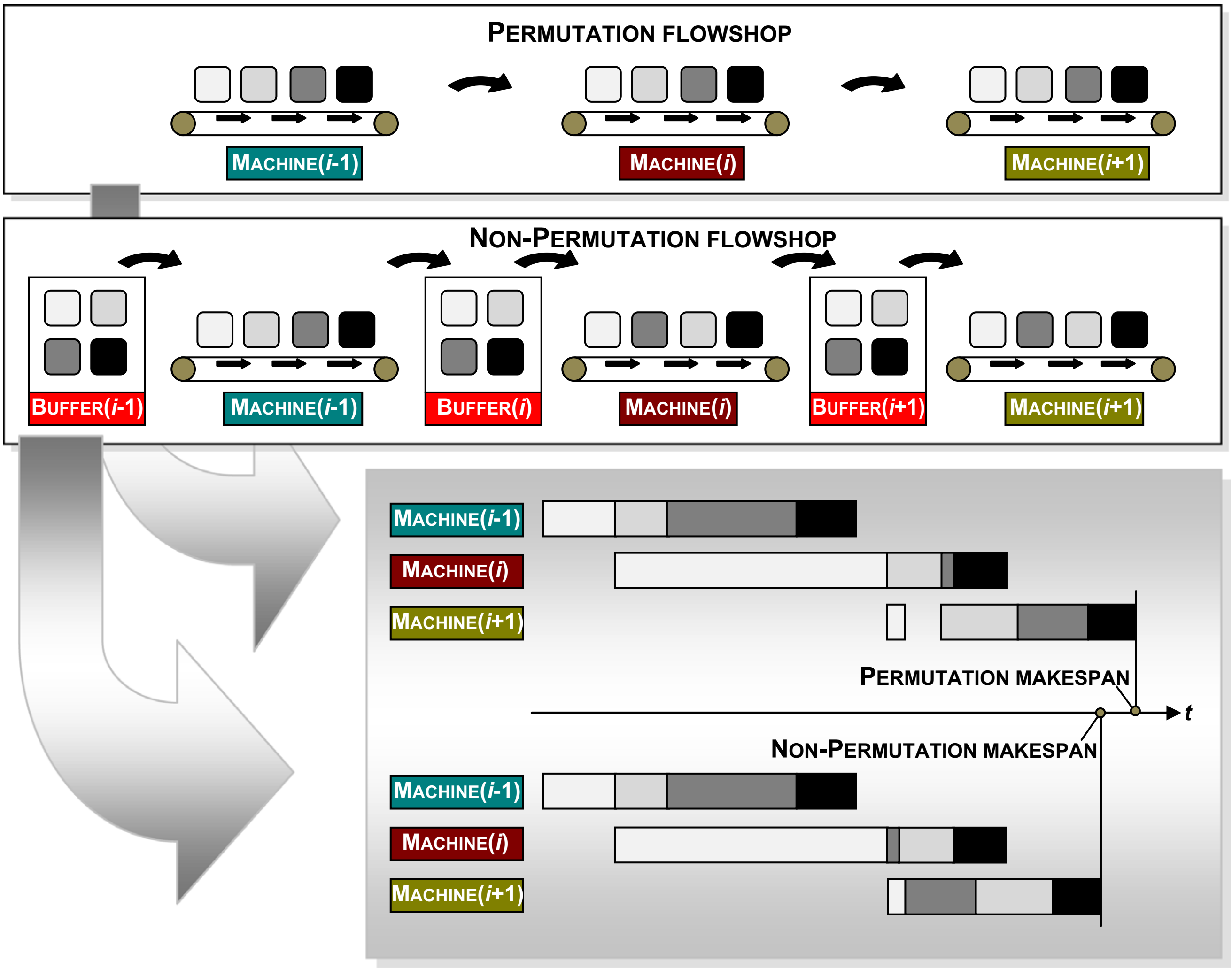

**Fig. 1.** Job Flowing: permutation flowshop (PFS) compared to non-permutation flowshop (NPFS) on a physical flow line and on a Gantt diagram.

It is well-known that the elimination of buffers has the benefit of reducing the work-in-process (WIP) and improving the line efficiency according to the lean manufacturing philosophy, however as shown in the Gantt diagram in Fig. 1, changing the sequence of jobs on different machines can have the benefit of shorter schedules. When buffers between machines are present, non-permutation schedules are likely to outperform their permutation counterparts (Tandon, Cummings, & LeVan, 1991). Permutation schedules are dominant up to the three-machine case. In the general case with $m$ (>3) machines, a permutation schedule is not necessarily optimal anymore (Potts, Shmoys, & Williamson, 1991). The analysis of theoretical upper bounds of permutation and non-permutation schedules of the well-known 120 instance benchmarks proposed by Taillard (1993), show that system performance are improved by NPFS scheduling (Vaessens, 1996). Simulations by Weng (2000) show that up to 100 jobs and 20 machines, a buffer sizes $B_i \geqslant 4$ gives a slight gap in performance with respect to unlimited buffers ($B_i = +\infty$). A possible interpretation is that efficient permutated schedules include permutations of jobs that are at a distance of 4 positions. Permutations of jobs that are very distant in the sequence produce less efficient schedules (e.g. higher makespan). It has also been found that the buffer size should be proportional to the number of jobs. Consequently it can be observed that non-permutation flowshop is more and more beneficial over permutation with higher job number.

In Fig. 2 the logical flow of current work is summarized with alternatives from the literature, reviewed in the next Section 2. It can be noticed that, as anticipated, scheduling on a flowline without buffers can be approached only as permutation flowshop (green path, left); permutation solutions can be used as input for non-permutation algorithms and produce non-permutation schedules (blue path, center). In Section 4 the non-permutation approach (formalized in Section 3) is described by a disjunctive graph (digraph) to show how to select or design new heuristics or metaheuristics able to build *natively* non-permutation solutions (red path, right) as opposed to improving permutation solutions found by other heuristics or metaheuristics. The proposed native-NPFS approach (Section 5) is based on ant colony optimization (ACO) from Bonabeau, Dorigo, and Theraulaz (2000), as discussed in detail in Section 6. Native and non-native approaches are experimentally compared in Section 7. Considering the benchmarks from Taillard (1990) and Demirkol et al. (1998).

## 2. Literature

A new class of computation techniques is leading considerable attention: metaheuristic algorithms. Examples of metaheuristics in flowshop scheduling include taboo search (Nowicki & Smutnicki, 1996), simulated annealing (Lin & Ying, 2009), immune algorithm