Solving the Tool Switching Problem with Memetic Algorithms

Jhon Edgar Amaya, Carlos Cotta, Antonio J. Fernández-Leiva

Jhon Edgar Amaya

Laboratorio de Computación de Alto Rendimiento. Universidad Nacional Experimental del Táchira. Av. Universidad. Paramillo. San Cristóbal. Venezuela. Tel.: +58-276-3552791. Fax: +58-276-3530266. E-mail: jedgar@unet.edu.ve

Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071. Málaga. Spain. Tel.: +34-952-13 7158. Fax: +34-952-13 1397. E-mail: ccottap@lcc.uma.es

Antonio J. Fernández-Leiva

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071. Málaga. Spain. Tel.: +34-952-13 3315. Fax: +34-952-13 1397. E-mail: afdez@lcc.uma.es

Short title: Solving the ToSP with Memetic Algorithms

List of the number of manuscript pages = 1,2,3,..., 50.

List of the number of tables = 1, 2, 3, 4, 5, 6.

List of the number of figures = 1, 2.

Solving the Tool Switching Problem with Memetic Algorithms

Abstract

The tool switching problem (ToSP) is well known in the domain of flexible manufacturing systems. Given a reconfigurable machine, the ToSP amounts to scheduling a collection of jobs on this machine (each of them requiring a different set of tools to be completed), as well as the tools to be loaded/unloaded at each step to process these jobs, such that the total number of tool switches is minimized. Different exact and heuristic methods have been defined to deal with this problem. In this work, we focus on memetic approaches to this problem. To this end, we have considered a number of variants of three different local-search techniques (namely hill climbing, tabu search and simulated annealing), and embedded them in a permutational evolutionary algorithm. It is shown that the memetic algorithm endowed with steepest-ascent hill climbing search yields the best results, performing synergistically better than its stand-alone constituents, and providing better results than the rest of the algorithms (including those returned by an effective ad-hoc beam search heuristic defined in the literature for this problem).

Keywords: Flexible Manufacturing System, Tool Switching Problem, Evolutionary Algorithm, Local Search, Memetic Algorithm.

1 Introduction

Flexible manufacturing systems (FMSs) have the capability to be adjusted for generating different products and/or for changing the order of product generation. Thus, they incorporate versatility and efficiency in the production process. This is precisely the reason that has motivated an increasing interest on this kind of systems; for some time now, the manufacturing industry is more and more often demanding flexible manufacturing systems as an alternative to traditional rigid production systems.

In the setting dealt in this work, we consider a simple machine that has several slots into which different tools can be loaded. Each slot just admits one tool, and each job executed on that machine requires a particular set of tools to be completed. Jobs are sequentially executed, and therefore each time a job is to be processed, the corresponding tools must be loaded in the machine magazine. The number of slots available in this magazine is obviously limited. Since in general the total number of tools required to process all jobs is also larger than the number of slots in the magazine, it may be required at some point to perform a tool switch, i.e., removing a tool from the magazine and inserting another one in its place. In this context, tool management is a challenging task that directly influences the efficiency of flexible manufacturing systems: an inadequate schedule of jobs and/or a poor tool switching policy may result in excessive delays for reconfiguring the machine.

Although the order of tools in the magazine is often irrelevant, the need of performing a tool switching does depend on the order in which the jobs are executed. The tool switching problem (ToSP) consists of finding an appropriate job sequence in which jobs will be executed, and an associated sequence of tool loading/unloading operations that minimizes the number of tool switches in the magazine. Clearly, this problem is specifically interesting when the time needed to change a tool is a significant part of the processing time of all jobs, and therefore the tool switching policy will significantly affect the performance of the system. Different examples of the problem can be found in diverse areas such as electronics manufacturing, metalworking industry, computer memory management, and aeronautics, among others (Belady, 1966; Bard, 1988; Tang & Denardo, 1988; Privault & Finke, 1995; Shirazi & Frizelle, 2001).

It must be noted that the ToSP is an extremely hard problem, whose difficulty scales up depending on the number of jobs, tools, and magazine capacity. As later described in Section 2.1, exact methods ranging from integer linear programming (ILP) techniques to heuristic constructive algorithms have been already applied to the problem with moderate success. The reason is clear: the ToSP has been proved to be NP-hard when the magazine capacity is higher than two (which is the usual case) and thus exact methods are inherently limited. In this context the use of alternative techniques that might eventually overcome this limitation has been explored. In particular, the use of metaheuristic techniques (Blum & Roli, 2003) can be considered. These techniques utilize high-lever strategies to combine basic heuristics, and their most distinctive feature is their ability to escape from local optima (or extrema). They thus have global optimization capabilities, although they cannot in general provide optimality proofs for the solutions they obtain. Nevertheless, when adequately crafted, they will likely provide optimal or near-optimal solutions to a wide range of continuous and combinatorial optimization problems.

Recently, Amaya et al. (2008) proposed three metaheuristics to tackle the ToSP: a simple local search (LS) scheme based on hill climbing, a genetic algorithm, and a memetic algorithm (Moscato & Cotta, 2003; Krasnogor & Smith, 2005; Moscato & Cotta, 2007) (MA), based on the hybridization of the two latter methods. This memetic algorithm produced very good results compared with a very efficient method –i.e., a beam search heuristic (Zhou et al., 2005)– that generated high quality results on a number of ToSP instances. That seminal work paves the way for considering other memetic approaches to the ToSP, based on the use of other recombination approaches, other local search techniques, partial Lamarckianism, as well as the utilization of alternative neighborhood structures. This has been done here, providing also an extensive empirical evaluation that includes a meticulous statistical comparison among 27 algorithms. Our analysis highlights the appropriateness of attacking the ToSP via metaheuristics – in particular memetic approaches– and yields a sound ranking of techniques for the problem, providing useful insights on its heuristic resolution.

2 Background

Before describing formally the ToSP, let us firstly overview the problem and its variants, and review related work.

2.1 Related Work

The ToSP is a combinatorial optimization problem that involves scheduling a number of jobs on a single machine such that the resulting number of tool switches required is kept to a minimum. We are going to focus here on the uniform case of the ToSP, in which there is one magazine, no job requires more tools than the magazine capacity, and the slot size is constant. To the best of our knowledge, the first reference to the uniform ToSP can be found in the literature as early as in the 1960s (Belady, 1966); since then, the uniform ToSP has been tackled via many different techniques. The late 1980s contributed especially to solve the problem (ElMaraghy, 1985; Kiran & Krason, 1988; Bard, 1988; Tang & Denardo, 1988). This way, Tang & Denardo (1988) proposed an ILP formulation of the problem, and Bard (1988) formulated the ToSP as a non-linear integer program with a dual-based relaxation heuristic. More recently, Laporte et al. (2004) proposed two exact algorithms: a branch-and-bound approach and a linear programming-based branch-and-cut algorithm. This latter one is based on a new ILP formulation with a better linear relaxation than that proposed previously by Tang & Denardo (1988). An alternative definition to the problem was formulated by Ghiani et al. (2007), who demonstrated that the ToSP is a symmetric sequencing problem; under this perspective, the authors enriched the branch-and-bound algorithm proposed by Laporte et al. (2004) with this new formulation, obtaining a significant computational improvement.

Despite the moderate success of exact methods, it must be noted that they are inherently limited, since Oerlemans (1992) and Crama et al. (1994) proved formally that the ToSP is NP-hard for $C > 2$, where C is the magazine capacity, i.e., the number of tools it can accomodate. This limitation was already highlighted by Laporte et al. (2004) who reported that their algorithm was capable of dealing with instances with 9 jobs, but provided very low success ratios for instances with more than 10 jobs. Some ad hoc heuristics have been devised in response to this complexity barrier. We refer to Amaya et al. (2008) for an overview of these. The use of metaheuristics has been also considered recently. In addition to Amaya et al. (2008) mentioned before, local search methods such as tabu search (TS) have been proposed (Hertz & Widmer, 1993; Al-Fawzan & Al-Sultan, 2003). Among these, we find specifically interesting the approach presented by Al-Fawzan & Al-Sultan (2003), due to the quality of the obtained results; they defined three different versions of TS that arose from the inclusion of different algorithmic mechanisms such as long-term memory and oscillation strategies. We will return later to this approach and describe it in more detail since it has been included in our experimental comparison.

A different, and very interesting, approach has been described by Zhou et al. (2005), who proposed a beam search algorithm. Beam search is a derivate of branch-and-bound that uses a breadth-first traversal of the search tree, and incorporates a heuristic choice to keep at each level only the best (according to some *quality* measure) β nodes (the so-called *beam width*). This sacrifices completeness, but provides a very effective heuristic search approach. Actually, this method provided good results, e.g., better than those of Bard's heuristics, and will be also included in the experimental comparison.

Note that the ToSP admits a number of variants. In this work we focus on the uniform ToSP (cf. Section 2.2), but this problem can be augmented if additional constraints are posed on tools or on the magazine. In this case, one refers to the so-called *non-uniform ToSP* (Crama et al., 2007). For example, it might be the case that different tools required slots of different sizes (or more than one slot); this was precisely the case addressed by Tzur & Altman (2004) that considered one magazine with slots of variable size, and pointed out three types of decisions to solve the problem, i.e., how to select the job sequence, which tools to switch before each processing operation, and where to locate each tool in the magazine by means of an integer-programming heuristic. An additional variant of the ToSP consists of having multiple magazines. Several proposals for solving this problem variant can be found in the literature; for instance, Kashyap & Khator (1994) analyzed the control rules for tool selection in a FMS with multiple magazines and used a particular policy to determine tool requirements. Błażewicz & Finke (1994) considered two-level nested scheduling problems (i.e., the part-machine scheduling problem, and the resource allocation and sequencing problem) and described some concrete models and solution procedures. Also, Hong-Bae et al. (1999) described several algorithms, (e.g., greedy search based techniques, as well as tool groupings-based methods) to solve the problem with a number of identical magazines, each of which had a particular capacity. A more general case with parallel machines and different magazine capacities was considered by Keung et al. (2001). From a wider perspective, Hop (2005) presents the ToSP as a hierarchical structure that can be analyzed under four different assumptions: variable size for the tools/slots, jobs requiring more tools than the magazine capacity, partial or complete job splitting, and (non-)concurrent tool changes/job changes. All variations of the problem considered under these assumptions were proven to be NP-complete.

2.2 Formulation of the Uniform Tool Switching Problem

In light of the informal description of the uniform ToSP given before, there are two major elements in the problem: a machine M and a collection of jobs $J = \{J_1, \cdots, J_n\}$ to be processed. Regarding the latter, the relevant information that will drive the optimization process are the tool requirements for each job. We assume that there is a set of tools $T = \{\tau_1, \cdots, \tau_m\}$, and that each job J_i requires a certain subset $T^{(J_i)} \subseteq T$ of tools to be processed. As to the machine, we will just consider one piece of information: the capacity C of the magazine (i.e., the number of available slots).

Given the previous elements, we can formalize the ToSP as follows: let a ToSP instance be represented by a pair, $I = \langle C, A \rangle$ where

- C denotes the magazine capacity,
- A is a $m \times n$ binary matrix that defines the tool requirements to execute each job, i.e., $A_{ij} = 1$ if, and only if, tool τ_i is required to execute job J_j, being 0 otherwise.

We assume that $C < m$; otherwise the problem is trivial. The solution to such an instance is a sequence $\langle J_{i_1}, \cdots, J_{i_n} \rangle$ (where $i_1, \ldots, i_n$ is a permutation of numbers $1, \ldots, n$) determining the order in which the jobs are executed, and a sequence $T_1, \cdots, T_n$ of tool configurations ($T_i \subset T$) determining which tools are loaded in the magazine at a certain time. Note that for this sequence of tool configurations to be feasible, it must hold that $T^{(J_{i_j})} \subseteq T_j$.

Let $\mathbb{N}_h = \{1, \cdots, h\}$ henceforth. We will index jobs (respectively tools) with integers from $\mathbb{N}_n$ (respectively $\mathbb{N}_m$). An ILP formulation for the ToSP is shown below, using two sets of zero-one decision variables:

- $x_{jk} = 1$ if job $j \in \mathbb{N}_n$ is assigned to position $k \in \mathbb{N}_n$ in the sequence, and 0 otherwise – see Eqs. (2) and (3),
- $y_{ik} = 1$ if tool $i \in \mathbb{N}_m$ is in the magazine at instant $k \in \mathbb{N}_n$, and 0 otherwise – see Eq. (4).

Processing each job requires a particular collection of tools loaded in the magazine. It is assumed that no job requires a number of tools higher than the magazine capacity, i.e., $\sum_{i=1}^{m} A_{ij} \leqslant C$ for all $j \in \mathbb{N}_n$. Tool requirements are reflected in Eq. (5). Following (Bard, 1988), we assume the initial condition $y_{i0} = 1$ for all $i \in \mathbb{N}_m$. This initial condition amounts to the fact that the initial loading of the magazine is not considered as part of the cost of the solution (in fact, no actual switching is required for this initial load). The objective function $F(\cdot)$ counts the number of switches that have to be done for a particular job sequence – see Eq. (1). We assume that that the cost of each tool switching is constant and unitary.

$$\min \; F(y) = \sum_{j=1}^{n} \sum_{i=1}^{m} y_{ij}(1 - y_{i,j-1}) \tag{1}$$

$$\forall j \in \mathbb{N}_n : \; \sum_{k=1}^{n} x_{jk} = 1 \tag{2}$$

$$\forall k \in \mathbb{N}_n : \sum_{j=1}^{n} x_{jk} = 1 \tag{3}$$

$$\forall k \in \mathbb{N}_n : \sum_{i=1}^{m} y_{ik} \leqslant C \tag{4}$$

$$\forall j, k \in \mathbb{N}_n \ \forall i \in \mathbb{N}_m : \ A_{ij} x_{jk} \leqslant y_{ik} \tag{5}$$

$$\forall j, k \in \mathbb{N}_n \ \forall i \in \mathbb{N}_m : \ x_{jk}, y_{ij} \in \{0, 1\} \tag{6}$$

Recall that this general definition shown above corresponds to the *uniform ToSP* in which each tool fits in just one slot.

2.3 The ToSP as a machine loading problem

The ToSP can be divided into three subproblems (Tzur & Altman, 2004): the first subproblem is *machine loading* and consists of determining the sequence of jobs; the second subproblem is *tool loading*, consisting of determining which tool to switch (if a switch is needed) before processing a job; finally, the third subproblem is *slot loading*, and consists of deciding where (i.e., in which slot) to place each tool. Since we are considering the uniform ToSP, the third subproblem does not apply (all slots are identical, and the order of tools is irrelevant). Therefore only two subproblems have to be taken into account: machine loading and tool loading. In the following we will show that the tool loading subproblem can be optimally solved if the sequence of jobs is known beforehand. This is very important for optimization purposes, since it means that the search effort can be concentrated on the machine loading stage.

As already mentioned the cost of switching a tool is considered constant (the same for all tools) in the uniform ToSP, the relevant decision being whether the tool is to be loaded in the magazine or not at any given time (were the size of the tools not uniform, the location of the tools in the magazine would be relevant too). Under this assumption, if the job sequence is fixed, the optimal tool switching policy can be determined in polynomial time using a greedy procedure termed *Keep Tool Needed Soonest* (KTNS) (Bard, 1988; Tang & Denardo, 1988)[1]. The functioning of this procedure is as follows:

- At any instant, insert all the tools that are required for the current job.

- If one or more tools are to be inserted and there are no vacant slots on the magazine, keep the tools that are needed soonest. Let $J = \langle J_{i_1}, \cdots, J_{i_n} \rangle$ be the job sequence, and let $T_k \subset \mathbb{N}_m$ be the tool configuration at time k. Let $\Xi_{jk}(J)$ be defined as

  $$\Xi_{jk}(J) = \min \left\{ t \mid (t > k) \wedge \left(A_{jJ_{i_t}} = 1 \right) \right\},$$

  that is, the next instant after time k at which tool τ_j will be needed again given sequence J. If a tool has to be removed, the tool τ_{j^*} maximizing $\Xi_{jk}(J)$ is chosen, i.e., remove the tools whose next usage is furthest in time.

The importance of this policy is that, as mentioned before, given a job sequence KTNS obtains its optimal number of tool switches. Therefore, we can concentrate on the machine

[1] As Błażewicz & Finke (1994) point out, the KTNS property was already known to Belady (1966).

loading subproblem, and use KTNS as a subordinate procedure to solve the subsequent tool loading subproblem. As an aside remark, the tool loading problem is NP-hard in the non-uniform ToSP, even if the job sequence is known and unit loading/unloading costs are assumed (Crama et al., 2007).

2.4 An Illustrative Example

To illustrate the formal definition of the problem given in previous subsections, let us present a small example. Let there be a machine with a magazine capacity $C = 4$, and let there be $n = 10$ jobs requiring a total number of $m = 9$ tools. More precisely, let the requirement matrix be the indicated in Table 1:

<TABLE 1>

Now, let us assume we have a job sequence $\langle 1, 6, 3, 7, 5, 2, 8, 4, 9, 10 \rangle$. The initial loading of the magazine must thus comprise the tools required by job 1, namely $T^{(1)} = \{2, 3, 6\}$. Since there are still free slots in the magazine, these are loaded with tools required by the next job in the sequence (job 6; this means tool 1 is loaded too), see Figure 1.

<FIGURE 1>

Job 1 can thus be executed, and so does job 6 without any tool switch. Next job is number 3, that requires tools $\{2, 6, 7\}$. Tools 2 and 6 are already in the magazine but 7 is not, so a tool must be unloaded to make room for it. Two options are available for this purpose: tools 1 and 3. The KTNS policy determines that tool 3 has to be replaced since the next time it will be required is when serving job 2 at position 6 in the sequence, whereas tool 1 is required again for job 5 in position 5 in the sequence. Job 7 come next and requires tools $\{6, 8\}$. Tool 8 is then loaded replacing tool 7 (required again by job 9 at time step 9; the other candidates for replacement were tool 2 – required by job 2 at time 6 – and tool 1 – required by job 5 at time 5). Now, job 5 requires tools $\{1, 5, 9\}$ and only tool 1 is loaded so a double switch is required. Candidates to be replaced at this point are: tool 2 (required by job 2 at time 6), tool 6 (not required again) and tool 8 (required again by job 8 at time 7). Therefore, tools 6 and 8 are replaced. Job 2 comes next and requires tools $\{2, 3, 5, 9\}$ among which only tool 3 is not loaded. In this case the only possibility is replacing tool 1 by tool 3. Proceeding to job 8, tools $\{5, 8, 9\}$ are needed so tool 8 enters in the magazine replacing tool 3 (not required again in the future; the same holds for tool 2, so it is irrelevant which one of the two is removed). Getting to job 4, tool 4 is required in addition to 9 (already loaded). The former enters the magazine substituting tool 8 (again, not used again, much like tool 2; tool 5 is however required later by job 10 at time 10). The last but one is job 9, needing tools $\{4, 7\}$. Since tool 4 is already in the magazine, only tool 7 has to be loaded, replacing either tool 2 or tool 9 (none of them required again in the future). Finally, job 10 is completed using tools $\{4, 5\}$ already in the magazine, so no new switch is required.

3 Solving the ToSP with Metaheuristics

Let us now describe the metaheuristics considered to tackle the ToSP. To do so, Section 3.1 deals with general issues of representation and neighborhood structure, whereas algorithmdependent issues are described in Sections 3.2 and 3.3.

3.1 Representation and Neighborhood Structure

The use of metaheuristics to solve the ToSP requires determining in each case how solutions will be represented, and which the structure of the underlying search space will be. For the purpose of the techniques considered in this work, these considerations turn out to be general issues that we address here. According to the discussion presented in previous subsection, the role of the metaheuristics will be to determine an optimal (or near optimal) job sequence, such that the total number of switches is minimized. Therefore, a permutational encoding arises as the natural way to represent solutions. Thus, a candidate solution for a specific ToSP instance $I = \langle C, A \rangle$ is simply a permutation $\pi = \langle \pi_1, \cdots, \pi_n \rangle \in \mathbb{P}_n$ where $\pi_i \in \mathbb{N}_n$, and $\mathbb{P}_n$ is the set of all permutations of elements in $\mathbb{N}_n$.

Having defined the representation, we now turn our attention to the neighborhood structure. This will be a central ingredient in the local-search-based metaheuristics considered, both when used as stand-alone techniques or when embedded within other search algorithms. Permutations are amenable to different neighborhood structures. We have focused on the following two ones:

1. The well-known *swap* neighborhood $\mathcal{N}_{swap}(\cdot)$, in which two permutations are neighbors if they just differ in two positions of the sequence, that is, for a permutation $\pi \in \mathbb{P}_n$

   $$\mathcal{N}_{swap}(\pi) = \{\pi' \in \mathbb{P}_n \mid H(\pi, \pi') = 2\}$$

   where $H(\pi, \pi') = n - \sum_{i=1}^{n}[\pi_i = \pi'_i]$ is the Hamming distance between sequences π and

π' (the number of positions in which the sequences differ), and $[\cdot]$ is Iverson bracket (i.e., $[P] = 1$ if P is true, and $[P] = 0$ otherwise). Given the permutational nature of sequences, this implies that the contents of the two differing positions have been swapped.

2. The *block* neighborhood $\mathcal{N}_{block}(\cdot)$, a generalization of the swap neighborhood in which a permutation π' is a neighbor of permutation π if the former can be obtained from the latter via a random block swap. A random block swap is performed as follows:

   (a) A block length $b_l \in \mathbb{N}_{n/2}$ is uniformly selected at random.

   (b) The starting point of the block $b_s \in \mathbb{N}_{n-2b_l}$ is subsequently selected at random.

   (c) Finally, an insertion point b_i is selected, such that $b_s + b_l \leqslant b_i \leqslant n - b_l$, and the segments $\langle \pi_{b_s}, \cdots, \pi_{b_s+b_l-1} \rangle$ and $\langle \pi_{b_i}, \cdots, \pi_{b_i+b_l-1} \rangle$ are swapped.

   Obviously, if the block length $b_l = 1$ then the operation reduces to a simple position swap, but this is not typically the case.

Having defined the neighborhood structures, the next step is deploying local-search-based procedures on them. This is described in next subsection.

3.2 Local Search Metaheuristics for the ToSP

Local search (LS) metaheuristics are based on exploring the neighborhood of a certain "current" solution, using some specific decision-making procedure to determine when and where within this neighborhood the search is to be continued. Thus, local search can be typically modelled as a trajectory in the search space, that is, an ordered sequence of solutions such that neighboring solutions in this sequence differ in some small amount of information. The quality of solutions in this sequence does not have to be monotonically increasing in general. Indeed, the ability of performing "downhill" moves, i.e., moving to a solution of inferior quality than the current one, is a crucial feature of most local search metaheuristics, allowing them to escape from local extrema, and hence endowing them with global optimization capabilities. Even more so, the dynamics of some local search techniques cannot even be modelled as a simple trajectory in search space, since some additional mechanisms can be considered to resume the search from a different point when stagnation is detected.

The first local search technique considered is classical exhaustive steepest-ascent hill climbing (HC). Given a current solution π, its neighborhood $\mathcal{N}(\pi)$ is fully explored, and the best solution found is taken as the new current solution, provided it is better than the current one (ties are randomly broken). If no such neighboring solution exist, the search is considered stagnated, and can be restarted from a different initial point.

The basic HC scheme suffers when confronted with a rugged search landscape, keeping the search trapped in low-quality local optima. In order to escape from these, a mechanism for accepting strictly non-improving moves has to be incorporated. One of the most classical proposals to this end is simulated annealing (SA) (Kirkpatrick et al., 1983). Inspired in the physical process of thermal cooling and residual strain relief in metals, SA uses a probabilistic criterion to accept a neighbor as the current one. This criterion is based on Boltzmann's law, and is parameterized by a so-called temperature value (recall the analogy with thermal cooling). More precisely, let Δf be the fitness difference between the tentative neighbor and the current solution (in this case, a negative value if the neighbor is better than the current solution), and let T be the current temperature. Then, the neighboring configuration is accepted with probability P given by

$$P = \begin{cases} 1, & \text{if } \Delta f > 0 \\ e^{-\frac{\Delta f}{k_B T}}, & \text{otherwise} \end{cases}$$

where k_B is Boltzmann's constant (which can be ignored in practice, by considering an appropriate scaling for the temperature). The current temperature T modulates this acceptance probability (if T is high, higher energy increases are allowed). The temperature is decreased from its initial value T_0 to a final value $T_k < T_0$ via a process termed cooling schedule. Two classical cooling schedules are geometric cooling, i.e., $T_{i+1} = \gamma T_i$ for some $\gamma < 1$, and arithmetic cooling, i.e., $T_{i+1} = T_i - \varepsilon$ for some $\varepsilon > 0$. These are however somewhat simplistic strategies, nowadays superseded by more sophisticated cooling schedules that adaptively modify the temperature in response to the evolution of the search. To be precise, we have also considered an approach based on adaptive cooling and reheating (cf. Elmohamed et al., 1998).

The idea underlying the use of adaptive cooling is keeping the system close to equilibrium by decreasing the temperature according to a search-state-dependant variable termed *specific heat*. This variable measures the variability of the cost of states at a given temperature; higher values indicate it will take longer to reach equilibrium and hence slower cooling is required. Following (Huang et al., 1986), the next temperature is thus calculated as

$$T_{i+1} = T_i e^{-\eta T_i / \bar{\sigma}(T_i)}$$

where η is a tunable parameter and $\bar{\sigma}(T_i)$ is a smoothed version of $\sigma(T_i)$, the standard deviation of cost at temperature T_i, computed as (Otten & van Ginneken, 1989; Diekmann et al., 1993):

$$\bar{\sigma}(T_{i+1}) = (1 - \nu)\sigma(T_{i+1}) + \nu\sigma(T_i)\frac{T_{i+1}}{T_i}$$

Parameter ν tunes the learning rate and is generally set to 0.95. As to reheating, it is invoked whenever the search is deemed stagnated (after n_ι evaluations without improvement, where n_ι is a parameter). In that case, the temperature is reset to

$$T_{i+1} = \kappa f_B + T(C_H^{\max})$$

where κ is a parameter, f_B is the cost of the best-so-far solution, and $T(C_H^{\max})$ is the temperature at which the specific heat $C_H(T) = \sigma^2(T)/T^2$ took its maximum value.

The last local search scheme considered is tabu search (TS) (Glover, 1989a,b). TS is a sophisticated extension of basic HC in which the best neighboring solution is chosen as the next configuration, even if it is worse than the current one. To prevent cycling, that is, the search returning to the same point after a few steps (consider for example that it may be the case that $y \in \mathcal{N}(x)$ is the best neighbor of x and viceversa), a tabu list of movements is kept. Hence, a neighboring solution is accepted only if the corresponding move is not tabu. This tabu status of a move is not permanent: it only lasts for a number of search steps, whose value is termed tabu tenure. This value can be fixed for all moves and/or the search process, or can be different for different moves or in different stages of the search. Furthermore, an aspiration criterion may be defined, so that the tabu status of a move can be overridden if a certain condition holds (e.g., improving the best known solution).

The TS method considered in this work is based on the proposal described by Al-Fawzan & Al-Sultan (2003). Different TS schemes were defined and compared therein, the best one turning out to be a TS algorithm featuring long-term memory and strategic oscillation. The first feature refers to the maintenance of a long term memory, in this case measuring the frequency of application of each move. The basic idea is to diversify the search penalizing neighbors attainable via frequent moves. As to the strategic oscillation mechanism, it refers to a procedure for switching between the two neighborhoods defined in Section 3.1. A deterministic criterion based on switching the neighborhood structure after a fixed number of iterations was reported by Al-Fawzan & Al-Sultan (2003) to perform better than a probabilistic criterion (i.e., choosing the neighborhood structure in each step, according to a certain probability distribution). No aspiration criterion is used in this algorithm.

3.3 A Population-based Attack to the ToSP

Unlike local search methods, population-based techniques maintain a pool of candidate solutions, which are used to generate new candidate solutions, not just by neighborhood search but by using other higher-arity procedures such as recombination, i.e., two or more solutions –appropriately termed parents– are combined to create new solutions (Bäck, 1996). While the relevance of recombination versus neighborhood search has been always debated (Reeves, 1994) –a common criticism being the fact that unless adequately crafted to the problem at hand, recombination may reduce to pure macromutation (Jones, 1995)– it is widely accepted that recombination can play a crucial role in information mixing, as well as in the balance between exploitation and exploration (Prügel-Bennett, 2010).

The first population-based approach considered is a steady-state genetic algorithm (GA): a single solution is generated in each generation, and inserted in the population replacing the worst individual. Selection is done by binary tournament. As to recombination, there are many possibilities defined in the literature – check (Oliver et al., 1987; Starkweather et al., 1991; Cotta & Troya, 1998; Larrañaga et al., 1999) among others. We have opted in this work for using uniform cycle crossover (UCX) (Cotta & Troya, 1998), an operator based on the manipulation of positional information. To be precise, it is a generalization of cycle crossover in which all cycles are firstly identified, and subsequently mixed at random. Notice that this operator ensures the new solution contains no exogenous positional information (each position is taken from one of the parents). As to mutation, we have considered the use of random block swap moves, as described in Section 3.1.

On the basis of this GA, we have defined a number of memetic algorithms (MA). MAs are hybrid methods based on the synergistic combination of ideas from different search techniques, most prominently from local search and population-based search. The term "memetic" stems from the notion of *meme*, a concept coined by Dawkins (1976) to denote an analogous of the gene in the context of cultural evolution. Indeed, information manipulation is much more flexible in MAs, thanks to the usage of algorithmic add-ons such as local search, exact techniques, etc. It must be noted that while the connection to cultural evolution is sometimes overstressed in the literature, it is useful to depart from biologically-constrained thinking that turns out to be very restrictive at times. As a matter of fact, the initial developments in memetic algorithms done by Moscato (1989) did not emanate from a biological metaphor, but from the idea of maintaining a population of cooperating/competing search agents, for which a combination of evolutionary algorithms and local search was just a convenient instantiation (local search for encapsulating search agents, and an evolutionary algorithm for encapsulating cooperation – via recombination – and competition – via selection and replacement). Check (Moscato & Cotta, 2010) for a recent up-to-date overview of these techniques.

The MAs considered in this work have been built by endowing the GA with each of the local search schemes previously defined. To be precise, we have used each of the algorithms (i.e., HC, SA, TS) defined in Section 3.2. While in some early memetic algorithms local search was performed on every generated individual, this is not necessarily the best choice (Sudholt, 2009). Indeed partial Lamarckianism (Houck et al., 1997), namely applying local search only to a fraction of individuals, can result in better performance. These individuals to which local search will be applied can be selected in many different ways (Nguyen et al., 2007). We have considered a simple approach in which local search is applied to any individual with a probability p_{LS}; in case of application, the improvement uses up a number of $LSevals$ evaluations (or in the case of HC until it stagnates, whatever comes first) – see Algorithm 1.

The underlying idea of this memetic algorithm is to combine the intensifying capabilities of the embedded local search method, with the diversifying features of population-based search, i.e., the population will spread over the search space providing starting points for a deeper local exploration. As generations go by, promising regions will start to be spotted, and the search will concentrate on them. Ideally, this combination should be synergistic (this will depend on the particulars of the combination, such as the intensity, frequency and depth of local search and its interplay with the underlying evolutionary dynamics – Sudholt (2009)), providing better results that either the GA or the local search techniques by themselves. Empirical evidence of this fact will be provided in next section.

4 Experimental results

The experiments have been performed considering five different basic algorithms: beam search (BS) presented by Zhou et al. (2005), three local search methods (HC, TS, and SA), and a GA. From these, a wide number of algorithms were devised and tested. For instance, in the case of BS, five different values from 1 up to 5 were considered for the beam width. Finally, memetic approaches based on the combination of the GA with each of the local search methods have been considered.

Regarding local search methods, we consider HC, TS, and three variants of SA with arithmetic cooling (SAA), geometric cooling (SAG) and adaptive cooling and reheating (SAR) respectively. Note also that the exploration of the whole neighborhood becomes more and more costly as the number of jobs increases, e.g., for 50 jobs, the number of *swap* neighbors for a given candidate is 1225, not to mention the even higher number of *block* neighbors. In a fixed computational budget scenario, this implies the allocated computational effort can be quickly consumed. For this reason, we have opted for taking into account also local search versions in which a partial exploration of the neighborhood is done by obtaining a fixed-size random sample. To be precise, the size of this sample has been chosen to be αn, i.e., proportional to the number of jobs (the value $\alpha = 4$ has been used). The notation HCP and HCF (respectively TSP and TSF) is used to indicate the HC variant (respectively TS variant) in which the neighborhood is partially or fully explored respectively (in the case of TS, the full exploration refers just to the swap neighborhood, since the block neighborhood has a huge size). Other details of each particular local search method are as follows. In the case of HC, the search is restarted from a different initial point if stagnation takes place before consuming the allotted number of evaluations. As to SA, the initial temperature T_0 has been chosen so that the initial acceptance rate is approximately 50% (this has been done by obtaining offline a small sample of random solutions to measure the average fitness difference θ, and taking $T_0 = 1.44\theta$). The cooling parameter (either geometric and arithmetic) has been chosen so that a final temperature $T_k = 0.1$ is reached in the number of evaluations allocated to the corresponding instance. As for adaptive cooling and reheating, we use $\eta = 10^{-4}$, $\nu = 0.95$, $\kappa = T_0/f_0$ (where f_0 is the mean cost of random solutions), and $n_\iota = 20$. Finally, regarding TS, the tabu tenure is 5, and the number of iterations on each neighborhood for performing strategic oscillation is 3. This corresponds to the setting used by Al-Fawzan & Al-Sultan (2003).

As to the GA (and subsequently to the MA), an elitist generational model replacing the worst individual of the population ($popsize = 30$, $p_X = 1.0$, $p_M = 1/n$ where n is the number of jobs, i.e., number of genes per individual) with binary tournament selection has been utilized. As mentioned in Section 3.3, mutation is done by applying a random block swap, and recombination uses UCX. Finally, regarding the memetic algorithms, we conducted preliminary experiments considering $P_{LS} \in \{0.001, 0.01, 0.1, 1.0\}$ and $LSevals \in \{100, 200, \cdots, 1000\}$ to analyze parameter sensitivity; the best results were obtained for values of $LSevals$ equal to 200 and 1000, and for $P_{LS} = 0.01$, and thus our memetic algorithms were run considering these values. Overall, this means twelve different versions of memetic algorithms i.e., those resulting from the hybridization of the GA with each of the six local search schemes pointed out above and fixing $LSevals$ to the two values mentioned before. The notation MA$xxyy$ is used, where xx stands for a particular local search technique, and $yy \in \{02, 10\}$ indicate $LSevals = 200$ and $LSevals = 1000$ respectively.

As far as we know, no standard benchmark exists for this problem (at least publicly available). For this reason, we have selected a wide set of problem instances that were considered in the literature (Bard, 1988; Hertz et al., 1998; Al-Fawzan & Al-Sultan, 2003; Zhou et al., 2005); to be precise, 16 instances have been selected, with number of jobs, number of tools, and machine capacity ranging in [10,50], [9,60] and [4,30] respectively. Table 2 shows the different problem instances chosen for the experimental evaluation where a specific instance with n jobs, m tools and machine capacity C is labeled as $C\zeta_n^m$.

<TABLE 2>

Five different datasets[2] (i.e., tool requirement matrices) were generated randomly per instance. Each dataset was generated with the constraint, already imposed in previous works such as (Hertz et al., 1998), that no job is *covered* by any other job in the sense that

[2] All datasets are available at http://www.unet.edu.ve/~jedgar/ToSP/ToSP.htm

for no two different jobs i and j, $T^{(J_i)} \subseteq T^{(J_j)}$. Were this the case, job i could be removed from the problem instance, since scheduling it immediately after job j would result in no tool switching. This consideration has been also taken into account by Bard (1988) and Zhou et al. (2005).

All algorithms –except beam search, see below– have been run 10 times per dataset (i.e., 50 runs per problem instance), for a maximum of $maxevals = \varphi n(m - C)$ evaluations[3] per run (with $\varphi > 0$). Preliminary experiments on the value of φ proved that $\varphi = 100$ is an appropriate value that allows to keep an acceptable relation between solution quality and computational cost. Regarding the BS algorithm, because of its deterministic nature, just one run per dataset (and per value of beam width) has been done. The algorithm was allowed to run till exhaustion of the search tree. Tables 3 and 4 show the obtained results, grouped by problem instance.

<TABLE 3>

<TABLE 4>

A first consideration regarding the results is the fact that TSP performs better than remaining non-hybrid techniques. Also, HCF performs better on average than BS versions in most of the instances (i.e., exactly in 13 out of 16 instances). However, HCP is not as competitive as its full-exploration counterpart. Note for example that the performance of

[3]Observe that the number of evaluations increases with the number of jobs and tools (assumed to be directly related with problem difficulty) and decreases when the magazine capacity increases (thus making the decision problem less tight).

HCP degrades when the instance is larger. This is not surprising, since such larger instances are likely to exhibit a much more rugged multimodal landscape, and basic local search schemes suffer in these scenarios. In this case, BS is capable of adjusting better than HCP to this curse of dimensionality, given its pseudo-population-based functioning (it is not truly population-based in the sense that no set of multiple full solutions is maintained, although it does indeed keep a *population* of constructive paths), which modulates the greediness of the branch selection mechanism. Observe that, in general, BS exhibits a very robust behavior in all its versions and shows a competitive performance with respect to the rest of the techniques (especially in larger instances of the problem i.e., for $C \geqslant 15$). SA with adaptive cooling and reheating significantly improves the performance of SAA and SAG (which do not generally provide competitive results with respect to the rest of techniques). These comparatively better results of SAR with respect to SAA and SAG, as well as the better results of TSP with respect to the remaining local search based techniques and to the GA, highlight the need of adaptive strategies to traverse the search space of the ToSP effectively. As to the GA, it offers a robust performance given the fact that rather standard parameters have been used. It actually provides very good results especially in the smaller instances of the problem (i.e., for $C < 10$), and exhibits a good overall performance (very competitive with respect to HC, SA, and BS, as well as TSF). The GA shows an irregular performance in some instances though; in particular, its performance worsens when the number of jobs increases (i.e., $n \geqslant 40$). This scaling difficulty in the case of the GA reflects the intricacy of the search landscape of the ToSP, and the problem it poses for a pure population-based approach in order to fine-tuning good solutions for larger sizes. While the GA can be good at jumping among different basins of attraction, identifying the corresponding local optima requires stronger intensification of the search. Such an intensification capability can be provided via the integration of a local search method and a population-based technique, using the memetic approaches defined above.

Inspecting the results of the hybrid local/population-based techniques (i.e., the memetic approaches) shown in Table 4, it can be seen that these often provide better results than their constituent parts (with the exception of the MATS* versions). For instance, notice that despite the poor performance of SAA and SAG, MASAA/MASAG variants are still capable of performing better than most local search techniques, although the combination does not reach synergistic value, since the results are comparable to those of the GA alone. A similar consideration can be done with respect to MATS* variants, although in this case its performance drops below that of the constituent parts. This may be due to the fact that the increased computational cost of a potentially larger search trajectory does not pay off (in other words, the TS schema has good diversification characteristics that results in good performance as a stand-alone technique, but does not contribute enough intensification in order to be effective within a memetic algorithm). Finally, observe that the hybridization of GA with HCP (i.e., MAHCP*) provides the best overall results, even better than the combination of GA with HCF, despite the fact that HCF performs much better than HCP as stand-alone technique. The reason may be that, when used as local improvement, the full exploration scheme in hill climbing demands a higher computational cost to produce a move in the search space.

<FIGURE 2>

In order to analyze the significance of the results and obtain a global perspective on how they compare to each other, we have used a rank-based approach. To do so, we have computed the rank r_j^i of each algorithm j on each instance i (rank 1 for the best, and rank k for the worst, where $k = 27$ is the number of algorithms; in case of ties, an average rank is awarded). The distribution of these ranks is shown in Fig. 2. Next, we have used two well-known non-parametric statistical tests (Lehmann & D'Abrera, 1998) to compare ranks:

- Friedman test (Friedman, 1937): we compute Friedman statistic value as

  $$\chi_F^2 = \frac{12N}{k(k+1)} \sum_{j=1}^{k} \left(R_j - \frac{k+1}{2} \right)^2$$

  where R_j is the mean rank of algorithm j across all N instances. The result is compared with the χ^2-distribution with $k-1$ degrees of freedom.

- Iman-Davenport test (Iman & Davenport, 1980): a less conservative test based on Friedman statistic value as follows:

  $$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \ .$$

  In this case the result is compared with the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

The results are shown in Table 5. As seen in the first row, the statistic values obtained are clearly higher than the critical values, and therefore the null hypothesis, namely that all algorithms are equivalent, can be rejected. Since there are algorithms with markedly poor performance, we have repeated the test with the top 7 algorithms (i.e., the MAs incorporating HC and SAR, and TSP), whose performance places them in a separate cluster from the remaining algorithms (cf. Figure 2). Again, it can be seen that the statistical test is passed, thus indicating significant differences in their ranks at $\alpha = 0.01$ level.

<TABLE 5>

Subsequently, we have focused in these top 7 algorithms, and performed Holm's test (Holm, 1979) in order to determine whether there exists significant differences with respect to a control algorithm (in this case MAHCP02, the algorithm with the best mean rank). To do so, we compute the following z-statistic for the ith algorithm:

$$z = (R_i - R_0)/\sqrt{\frac{k(k+1)}{6N}}\ .$$

Then, we determine the corresponding p-value for a normal distribution, and sort the algorithms for increasing p-values. Finally, these p-values are compared with and adjusted critical p-value α/i, where α is the significance level and i is the algorithm's position (1 for the lowest p-value, $k-1$ for the highest p-value; recall that one algorithm is used as control, and hence there are only $k-1$ slots). Tests are sequentially done for increasing p-values until the null hypothesis cannot be rejected at a certain i. In that case, the null hypothesis is retained for every $j \leqslant i$, i.e., algorithms with larger p-values. The results are shown in Table 6. Notice that –with the exception of MAHCP10, for which statistical significance can only be established at 82% level– the test is passed at 99% confidence level for all algorithms with respect to MAHCP02. This is a robust result that indicates a clear trend of superiority of MAHCP02 over the remaining approaches.

<TABLE 6>

5 Conclusions and Future Work

We have tackled the uniform tool switching problem with different techniques, and showed how metaheuristics can be very adequate to solve the problem. To be precise, we have conducted an extensive empirical evaluation of three different local-search heuristics (hill climbing, simulated annealing, tabu search), genetic algorithms, and memetic algorithms. The experimentation has included the beam search method described in (Zhou et al., 2005), since it was demonstrated to be especially effective compared to other techniques previously published. The results show that metaheuristics provide encouraging results, and are capable of improving the results obtained by beam search.

Starting on a general note, one of the main conclusions to be extracted from the results is the versatileness and effectiveness of memetic algorithms as a search paradigm. They constitute a natural framework in which different heuristics can be seamlessly integrated into a single optimization engine. Thus, MAs should not be regarded as competitors for existing approaches; on the contrary, it is much more appropriate to regard them as integrators: whenever single metaheuristics start to reach their limits, the use of memetic algorithms is in order to overcome these limitations.

Focusing now on each of the techniques considered, the experimental results indicate that tabu search is the most effective local search technique among the proposals considered. Its ability to traverse the search space escaping from local optima, and the enhanced exploration capabilities provided by the use of a strategic oscillation mechanism are crucial for this. Regarding the genetic algorithm, the particular recombination operator utilized –uniform cycle crossover– has shown the relevance of processing structural positional information to create new tentative sequences. A similar consideration can be made with respect to the choice of both local search technique to be embedded in the memetic algorithm and its neighborhood exploration policy. Regarding the first issue, the memetic algorithm endowed with HC has yielded the best results, improving both the GA and the remaining local-search techniques as stand-alone techniques, and thus providing evidence of the synergy of the combination. The reason why MAHC* behaves better than MATS* can be found in the better tradeoff between search intensification and computational cost provided by the former. While TS can provide improved solutions with respect to HC, its role when embedded within an memetic algorithm is different, since it has to share exploration duties with the underlying GA. Hence, the savings in computational effort obtained by removing some of this diversification capability from the local-searcher (which can thus focus on intensifying the search in promising regions) results in a net gain for the hybrid approach. This guideline seems generalizable to other related engineering problems –e.g., single machine total weighted tardiness (Maheswaran et al., 2005); see also (Cotta & Fernández, 2007)– in which simple and more intensive local improvement strategies perform adequately.

Regarding the choice of the scheme for exploring the neighborhood in the process of local improvement embedded in an memetic algorithm, the less computationally demanding option considering a sample instead of the whole neighborhood produces better results to solve the ToSP. Again, this is due to the balance between the computational cost of local search and the potentially attainable gain in solution quality. The interplay between the local search and population-based component of the memetic algorithm demands the former is applied at a low rate, and with a moderate intensity.

In connection to this last issue, and as an avenue for further research, it would be interesting to explore in more detail the intensification/diversification balance within the memetic algorithm. In this work we have leaned towards a more explorative combination, by using a blind recombination operator in the GA. It would be worth exploring other models though, e.g., by incorporating an intense exploration of the dynastic potential (i.e., set of possible children) of the solutions being recombined. Ideas from local branching (Fischetti & Lodi, 2003) or from dynastically optimal recombination (Cotta & Troya, 2003; Gallardo et al., 2007) could be used here. We also plan to analyze new instances and variants of the problem (Kashyap & Khator, 1994; Błażewicz & Finke, 1994; Hong-Bae et al., 1999) in the future.

Acknowledgements

We would like to thank the reviewers for their useful comments. The second and third authors are partially supported by Spanish MCINN under project NEMESIS (TIN2008-05941) and Junta de Andalucía under project TIC-6083.

*Bibliography

Al-Fawzan, M. & Al-Sultan, K. (2003). A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers & Industrial Engineering* 44(1), 35–47.

Amaya, J., Cotta, C. & Fernández, A. (2008). A memetic algorithm for the tool switching problem. In *Hybrid Metaheuristics 2008* (M. Blesa et al., Eds.). Vol. 5296 of *Lecture Notes in Computer Science*, pp. 190–202. Málaga: Springer-Verlag.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press. New York.

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

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

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

Blum, C. & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* 35(3), 268–308.

Cotta, C. & Fernández, A. (2007). Memetic algorithms in planning, scheduling and timetabling. In *Evolutionary Scheduling* (K. Dahal, K.-C. Tan & P. Cowling, Eds.), pp. 1–30. Berlin-Heidelberg: Springer-Verlag.

Cotta, C. & Troya, J. (1998). Genetic forma recombination in permutation flowshop problems. *Evolutionary Computation* 6(1), 25–44.

Cotta, C. & Troya, J. (2003). Embedding branch and bound within evolutionary algorithms. *Applied Intelligence* 18(2), 137–153.

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

Crama, Y., Moonen, L., Spieksma, F. & Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research* 182(2), 952–957.

Dawkins, R. (1976). *The Selfish Gene*. Clarendon Press. Oxford.

Diekmann, R., Lüling, R. & Simon, J. (1993). Problem independent distributed simulated annealing and its applications. In *Applied Simulated Annealing* (R. Vidal, Ed.). Vol. 3962 of *Lecture Notes in Economics and Mathematical Systems*, pp. 17-44. Berlin Heidelberg: Springer-Verlag.

ElMaraghy, H. (1985). Automated tool management in flexible manufacturing. *Journal of Manufacturing Systems* 4(1), 1–14.

Elmohamed, M. A. S., Coddington, P. D. & Fox, G. (1998). A comparison of annealing techniques for academic course scheduling. In *Practice and Theory of Automated Timetabling II* (E. Burke & M. Carter, Eds.). Vol. 1498 of *Lecture Notes in Computer Science*, pp. 92–112. Berlin Heidelberg: Springer-Verlag.

Fischetti, M. & Lodi, A. (2003). Local branching. *Mathematical Programming B* 98, 23–47.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association* 32(200), 675–701.

Gallardo, J., Cotta, C. & Fernández, A. (2007). On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 37(1), 77–83.

Ghiani, G., Grieco, A. & Guerriero, E. (2007). An exact solution to the TLP problem in an NC machine. *Robot. Comput.-Integr. Manuf.* 23(6), 645–649.

Glover, F. (1989a). Tabu search – part I. *ORSA Journal of Computing* 1(3), 190–206.

Glover, F. (1989b). Tabu search – part II. *ORSA Journal of Computing* 2(1), 4–31.

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

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

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 65–70.

Hong-Bae, J., Yeong-Dae, K. & Suh, S. H.-W. (1999). Heuristics for a tool provisioning problem in a flexible manufacturing system with an automatic tool transporter. *IEEE Transactions on Robotics and Automation* 15(3), 488–496.

Hop, N. V. (2005). The tool-switching problem with magazine capacity and tool size constraints. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 38(5), 617– 628.

Houck, C., Joines, J., Kay, M. & Wilson, J. (1997). Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation* 5(1), 31–60.

Huang, M., Romeo, F. & Sangiovanni-Vincentelli, A. (1986). An efficient general cooling schedule for simulated annealing. *1986 IEEE International Conference on Computer Aided Design (ICCAD)*, pp. 381–384. Santa Clara, CA: IEEE Press.

Iman, R. & Davenport, J. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics* 9, 571–595.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis. University of New Mexico.

Kashyap, A. & Khator, S. (1994). Modeling of a tool shared flexible manufacturing system. *Proceedings of the 26th Simulation Conference*. International Society for Computer Simulation, pp. 986–993. San Diego, CA.

Keung, K. W., Ip, W. H. & Lee, T. C. (2001). A genetic algorithm approach to the multiple machine tool selection problem. *Journal of Intelligent Manufacturing* 12(4), 331–342.

Kiran, A. & Krason, R. (1988). Automated tooling in a flexible manufacturing system. *Industrial Engineering* 20, 52–57.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983* 220, 4598, 671–680.

Krasnogor, N. & Smith, J. (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation* 9(5), 474–488.

Laporte, G., Salazar-González, J. & Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions* 36(1), 37–45.

Larrañaga, P., Kuijpers, C., Murga, R., Inza, I. & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Articial Intelligence Review* 13, 129–170.

Lehmann, E. & D'Abrera, H. (1998). *Nonparametrics: statistical methods based on ranks.* Prentice-Hall. Englewood Cliffs, NJ.

Maheswaran, R., Ponnambalam, S. & Aranvidan, C. (2005). A meta-heuristic approach to single machine scheduling problems. *International Journal of Advanced Manufacturing Technology* 25, 772–776.

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Technical Report Caltech Concurrent Computation Program, Report. 826*. California Institute of Technology. Pasadena, California, USA.

Moscato, P. & Cotta, C. (2003). A gentle introduction to memetic algorithms. *Handbook of Metaheuristics*, pp. 105–144. Boston MA: Kluwer Academic Press.

Moscato, P. & Cotta, C. (2007). Memetic algorithms. In *Handbook of Approximation Algorithms and Metaheuristics* (T. González, Ed.). Chapman & Hall/CRC Press. chapter 27.

Moscato, P. & Cotta, C. (2010). A modern introduction to memetic algorithms. In *Handbook of Metaheuristics (2nd Edition)* (M. Gendrau & J.-Y. Potvin, Eds.). Vol. 146 of *International Series in Operations Research and Management Science*, pp. 141–183. New York NY: Springer-Verlag.

Nguyen, Q. H., Ong, Y.-S. & Krasnogor, N. (2007). A study on the design issues of memetic algorithm. In *2007 IEEE Congress on Evolutionary Computation* (D. Srinivasan & L. Wang, Eds.), pp. 2390–2397. Singapore: IEEE Press.

Oerlemans, A. (1992). *Production planning for flexible manufacturing systems*. Ph.d. dissertation. University of Limburg. Maastricht, Limburg, Netherlands.

Oliver, I., Smith, D. & Holland, J. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the 2^{nd} International Conference on Genetic Algorithms* (J. Grefenstette, Ed.), pp. 224–230. Hillsdale NJ: Lawrence Erlbaum Associates.

Otten, R. & van Ginneken, L. (1989). *The Annealing Algorithm*. Kluwer Academic Publishers.

Privault, C. & Finke, G. (1995). Modelling a tool switching problem on a single NC-machine. *Journal of Intelligent Manufacturing* 6(2), 87–94.

Prügel-Bennett, A. (2010). Benefits of a population: Five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation* 14(4), 500 – 517.

Reeves, C. (1994). Genetic algorithms and neighbourhood search. In *Evolutionary Computing* (T. Fogarty, Ed.). Vol. 865 of *Lecture Notes in Computer Science*, pp. 115–130. Berlin Heidelberg: Springer-Verlag.

Shirazi, R. & Frizelle, G. (2001). Minimizing the number of tool switches on a flexible machine: an empirical study. *International Journal of Production Research* 39(15), 3547–3560.

Starkweather, T., McDaniel, S., Mathias, K., Whitley, D. & Whitley, C. (1991). A comparison of genetic sequencing operators. In *Proceedings of the 4th International Conference on Genetic Algorithms* (R. Belew & L. Booker, Eds.), pp. 69–76. San Mateo CA: Morgan Kauffman.

Sudholt, D. (2009). The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science* 410(26), 2511–2528.

Tang, C. & Denardo, E. (1988). Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. *Operations Research* 36(5), 767–777.

Tzur, M. & Altman, A. (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions* 36(2), 95–110.

Zhou, B.-H., Xi, L.-F. & Cao, Y.-S. (2005). A beam-search-based algorithm for the tool switching problem on a flexible machine. *The International Journal of Advanced Manufacturing Technology* 25(9-10), 876–882.

Biographies

Jhon Edgar Amaya, received the degree of Electronic Engineer from the Universidad Nacional Experimental del Tachira, Venezuela, in 1997, the M.Sc. degree in Computation from the Universidad de Los Andes, Venezuela, in 2003 and currently is a Computer Science PhD student in the University of Málaga (Spain) under the supervision of Dr. Carlos Cotta and Dr. Antonio J. Fernández. He is also a member of the High Performance Computing Laboratory (LCAR) of the Universidad Nacional Experimental del Tachira (UNET). His research interests are evolutionary computation and applications of artificial intelligence to practical problems.

Carlos Cotta, received the MS and PhD degrees in Computer Science from the University of Málaga in 1994 and 1998 respectively. He is currently an associate professor at the School of Computer Science in the University of Málaga. His area of research comprises areas such as metaheuristics (in particular evolutionary computation), combinatorial optimization, and bioinformatics. He is involved in the technical organization and program committees of the major conferences in the field of evolutionary computation, and has co-edited several books on knowledge-driven computing, adaptive metaheuristics, and evolutionary combinatorial optimization.

Antonio J. Fernández-Leiva, received the BS and MS degrees in Computer Science from the University of Málaga in 1991 and 1995 respectively. In 2002 he obtained his

PhD degree in the same University under the supervision of Dr. Patricia M. Hill from Leeds University where he spent long periods of time during four years. He is currently an associate professor at the School of Computer Science in the University of Málaga. His area of research comprises areas such as the implementation of constraint programming languages, and the attainment of hybrid optimization techniques that involve evolutionary algorithms.

Algorithm 1: Pseudocode of a basic MA based on a local search LS

```
for i ∈ ℕ_μ do
    pop[i] ← RANDOM-SOLUTION();
    LOCAL-IMPROVEMENT (pop[i]);
end for
i ← 0;
while i < MaxEvals do
    RANK-POPULATION (pop); // sort population according to fitness
    parent_1 ← SELECT (pop);
    if Rand[0,1] < p_X then // recombination is done
        parent_2 ← SELECT (pop);
        child ← RECOMBINE (parent_1, parent_2);
    else
        child ← parent_1;
    end if
    child ← MUTATE (child, p_M); // p_M is the mutation probability per gene
    if Rand[0,1] < p_LS then // LS is applied
        LOCAL-IMPROVEMENT (child); // Local Improvement
    end if
    pop[μ] ← child; // replace worst
end while
return best solution in pop;
```

Table 1: Example of tool requirement matrix. Each cell A_{ij} identifies if a particular job j requires ($\bullet$) tool i or not ($\circ$).

	jobs									
tools	1	2	3	4	5	6	7	8	9	10
1	○	○	○	○	●	●	○	○	○	○
2	●	●	●	○	○	○	○	○	○	○
3	●	●	○	○	○	●	○	○	○	○
4	○	○	○	●	○	○	○	○	●	●
5	○	●	○	○	●	○	○	●	○	●
6	●	○	●	○	○	●	●	○	○	○
7	○	○	●	○	○	○	○	○	●	○
8	○	○	○	○	○	○	●	●	○	○
9	○	●	○	●	●	○	○	●	○	○

Table 2: Problem Instances considered in the experimental evaluation. The minimum and maximum of tools required for all the jobs is indicated, as well as the work(s) from which the problem instance was obtained: [1] (Al-Fawzan & Al-Sultan, 2003), [2] (Bard, 1988), [3] (Hertz et al., 1998), [4] (Zhou et al., 2005).

	$4\zeta_{10}^{9}$	$4\zeta_{10}^{10}$	$6\zeta_{10}^{15}$	$6\zeta_{15}^{12}$	$6\zeta_{15}^{20}$	$8\zeta_{20}^{15}$	$8\zeta_{20}^{16}$	$10\zeta_{20}^{20}$
Min.	9	9	11	4	6	6	7	9
Max.	24	24	30	10	15	15	20	20
Source	[2,4]	[1,3]	[4]	[2,4]	[3]	[1]	[2,4]	[2,4]

	$10\zeta_{30}^{25}$	$15\zeta_{30}^{40}$	$15\zeta_{40}^{30}$	$20\zeta_{40}^{60}$	$24\zeta_{20}^{30}$	$24\zeta_{20}^{36}$	$25\zeta_{50}^{40}$	$30\zeta_{20}^{40}$
Min.	4	6	6	7	9	9	9	11
Max.	10	15	15	20	24	24	20	30
Source	[1]	[3]	[1]	[3]	[2,4]	[2,4]	[1]	[4]

Table 3: Results of GA, BSβ considering several values ($1 \leqslant i \leqslant 5$) for the beam width β, and different versions of HC, TS, and SA. Best results (in terms of the best solution average) are underlined and marked in boldface. Av = solution average; SD = standard deviation.

		GA	HCP	HCF	SAA	SAG	SAR	TSP	TSF	BS1	BS2	BS3	BS4	BS5
$4\zeta_{10}^{9}$	Av	**<u>7.98</u>**	8.40	8.40	10.46	10.14	8.58	8.08	8.12	8.40	8.40	8.40	8.40	8.40
	SD	0.71	1.00	1.11	2.19	1.72	1.13	0.74	0.77	0.49	0.49	0.49	0.49	0.49
$4\zeta_{10}^{10}$	Av	**<u>8.72</u>**	9.60	9.34	11.54	11.74	9.38	8.80	9.06	10.00	9.80	9.60	9.60	9.60
	SD	1.58	1.57	1.56	2.17	2.08	1.66	1.61	1.58	2.10	1.83	2.06	2.06	2.06
$6\zeta_{10}^{15}$	Av	13.78	14.7	14.38	17.6	16.96	14.32	**<u>13.68</u>**	13.82	15.20	14.80	14.80	14.80	14.80
	SD	2.02	2.25	2.22	2.43	2.55	2.09	2.10	2.00	1.47	1.47	1.47	1.47	1.47
$6\zeta_{15}^{12}$	Av	**<u>16.18</u>**	20.10	18.32	23.32	24.06	19.12	16.46	17.08	18.20	17.60	17.60	17.40	17.40
	SD	1.92	2.05	1.71	2.92	3.24	2.76	1.93	2.09	0.75	1.02	1.02	1.20	1.20
$6\zeta_{15}^{20}$	Av	23.36	26.54	24.76	30.54	30.56	23.78	**<u>23.02</u>**	23.40	26.20	25.80	25.20	25.20	25.20
	SD	2.06	2.40	2.35	3.18	3.33	2.27	2.00	1.97	2.32	2.14	1.60	1.60	1.60
$8\zeta_{20}^{15}$	Av	24.3	28.9	24.46	34.66	34.30	28.72	**<u>23.62</u>**	24.3	27.0	26.00	25.60	25.20	25.20
	SD	3.51	4.17	3.29	5.05	4.36	4.60	3.63	3.79	3.95	4.05	4.27	4.12	4.12
$8\zeta_{20}^{16}$	Av	28.58	33.78	28.76	39.28	40.54	32.22	**<u>27.92</u>**	28.76	29.4	29.40	29.40	29.40	29.40
	SD	2.14	2.48	1.49	3.34	4.45	3.96	2.13	2.07	1.62	1.62	1.62	1.62	1.62
$10\zeta_{20}^{20}$	Av	31.86	37.46	34.40	43.72	44.16	36.32	**<u>30.72</u>**	31.78	34.20	33.60	33.40	33.40	33.40
	SD	2.53	2.88	1.64	3.98	3.47	4.61	2.50	2.46	3.19	2.87	2.80	2.80	2.80
$10\zeta_{30}^{25}$	Av	74.94	85.46	69.6	96.20	96.48	80.86	**<u>67.72</u>**	85.34	73.6	70.80	70.8	70.80	70.60
	SD	2.70	2.97	1.02	4.51	3.92	8.34	1.52	12.72	1.02	1.47	1.47	1.47	1.50
$15\zeta_{30}^{40}$	Av	111.06	121.2	103.00	134.54	134.48	114.58	**<u>101.72</u>**	104.28	111.60	110.00	109.20	107.80	107.80
	SD	13.41	14.61	14.03	16.69	15.89	13.99	13.07	13.44	15.19	13.55	13.41	13.26	13.26
$15\zeta_{40}^{30}$	Av	114.54	127.54	**<u>100.34</u>**	142.46	142.12	126.08	101.9	130.5	105.20	103.20	102.80	102.80	102.40
	SD	8.96	9.51	8.84	10.91	10.59	11.94	8.14	17.39	8.52	9.58	9.74	9.74	9.97
$20\zeta_{40}^{60}$	Av	233.64	248.7	214.42	269.76	269.86	240.48	**<u>213.74</u>**	255.80	221.80	220.00	218.80	218.60	218.60
	SD	8.84	10.17	9.88	12.80	12.25	21.04	8.38	39.84	7.03	6.16	5.71	5.82	5.82
$24\zeta_{20}^{30}$	Av	25.54	30.24	25.80	34.18	35.14	29.82	**<u>25.04</u>**	25.72	33.00	32.60	32.40	32.40	32.40
	SD	3.23	3.69	2.71	3.90	4.23	4.70	3.02	3.33	4.43	4.50	4.63	4.63	4.63
$24\zeta_{20}^{36}$	Av	47.14	53.24	48.48	59.40	60.60	50.36	**<u>45.9</u>**	47.04	54.00	54.00	53.80	53.80	53.40
	SD	8.62	9.33	8.69	11.11	11.33	9.79	8.98	8.84	8.20	8.20	8.33	8.33	7.61
$25\zeta_{50}^{40}$	Av	174.64	191.02	**<u>150.88</u>**	210.12	209.22	190.78	153.58	196.28	167.2	164.00	162.80	162.80	161.80
	SD	13.64	13.36	15.29	16.44	14.02	18.12	12.89	31.63	12.91	12.55	12.34	12.34	12.16
$30\zeta_{20}^{40}$	Av	42.82	49.12	44.40	55.68	56.1	46.50	**<u>42.12</u>**	42.82	52.20	50.20	50.20	50.20	50.20
	SD	4.81	5.29	5.78	6.65	7.11	5.85	4.34	4.68	6.24	7.03	7.03	7.03	7.03

Table 4: Results of the variants of MA considered. Best results (in terms of the best solution average) are underlined and marked in boldface. Av = solution average; SD = standard deviation.

		MAHCP02	MAHCP10	MAHCF02	MAHCF10	MASAA02	MASAA10	MASAG02	MASAG10	MASAR02	MASAR10	MATSP02	MATSP10	MATSF02	MATSF10
$4\zeta_{10}^{9}$	Av	**7.84**	7.88	7.88	7.88	7.98	8.32	8.04	8.42	8.10	8.12	8.08	8.46	8.08	8.40
	SD	0.73	0.71	0.71	0.71	0.73	1.03	0.69	1.10	0.78	0.86	0.74	1.00	0.74	1.00
$4\zeta_{10}^{10}$	Av	**8.60**	8.62	**8.60**	8.62	8.74	9.24	8.76	9.24	8.96	8.96	8.78	9.20	8.78	9.50
	SD	1.62	1.62	1.62	1.65	1.65	1.63	1.67	1.68	1.66	1.60	1.69	1.70	1.57	1.63
$6\zeta_{10}^{15}$	Av	13.62	13.66	**13.6**	13.62	13.8	14.18	13.78	14.20	13.92	13.98	13.78	14.12	13.78	14.30
	SD	2.14	2.12	2.15	2.14	2.10	2.13	2.06	2.20	2.06	2.07	2.03	2.09	2.11	2.04
$6\zeta_{15}^{12}$	Av	15.64	**15.50**	15.86	15.52	16.40	18.30	16.36	18.24	16.32	16.50	17.04	18.86	17.72	19.04
	SD	1.89	1.80	1.89	1.69	1.92	2.36	1.95	2.07	1.93	2.06	2.28	2.45	2.16	2.62
$6\zeta_{15}^{20}$	Av	**22.16**	**22.16**	22.40	22.22	22.98	24.06	23.12	23.76	23.38	23.26	23.28	23.92	23.40	24.28
	SD	1.82	1.75	1.92	1.87	2.08	2.21	2.04	2.29	2.16	1.98	1.98	2.19	2.15	2.27
$8\zeta_{20}^{15}$	Av	22.36	**22.20**	23.06	22.8	24.24	26.94	24.10	27.12	22.94	23.32	25.3	27.26	27.24	27.24
	SD	3.51	3.49	3.91	3.64	3.50	3.57	3.21	4.02	3.61	3.55	3.56	4.12	3.59	3.72
$8\zeta_{20}^{16}$	Av	26.70	**26.58**	27.52	26.96	28.34	31.30	28.12	31.24	26.96	27.30	29.58	31.16	31.54	32.22
	SD	2.06	1.98	2.44	1.98	2.30	2.23	2.21	2.74	2.13	1.96	2.56	2.39	2.63	2.48
$10\zeta_{20}^{20}$	Av	29.50	**29.24**	29.94	29.88	31.34	33.78	31.44	33.44	30.38	30.80	31.98	33.70	33.64	34.16
	SD	2.59	2.51	2.50	2.60	2.71	3.14	2.79	3.35	2.43	2.41	2.96	3.04	3.14	3.14
$10\zeta_{30}^{25}$	Av	**63.70**	64.96	67.76	71.84	71.14	76.42	71.14	76.70	64.20	65.30	74.44	78.84	82.00	82.36
	SD	2.11	1.95	2.86	3.21	2.87	3.14	2.96	3.01	2.43	2.41	2.92	3.00	2.85	2.43
$15\zeta_{30}^{40}$	Av	**97.38**	97.62	99.50	102.02	104.7	107.80	104.50	109.14	99.14	99.22	107.3	110.4	114.06	113.68
	SD	12.59	13.19	13.23	14.45	13.25	13.75	13.83	13.69	13.08	13.11	12.95	15.09	14.1	15.16
$15\zeta_{40}^{30}$	Av	**95.18**	100.50	104.12	114.66	108.42	114.18	108.90	115.18	95.94	96.72	115.12	114.72	127.04	125.98
	SD	7.51	9.28	10.50	10.57	9.03	10.12	8.30	9.72	7.97	7.52	9.67	9.76	9.54	9.55
$20\zeta_{40}^{60}$	Av	**203.24**	207.08	207.30	209.04	218.50	226.18	219.10	225.34	203.88	205.40	225.12	226.52	238.70	240.04
	SD	8.32	9.52	8.40	8.56	9.55	9.15	10.01	9.66	8.23	8.39	10.93	9.57	8.74	11.33
$24\zeta_{20}^{30}$	Av	24.10	**24.00**	24.88	24.38	25.48	28.62	25.48	28.5	24.62	24.98	26.68	28.66	28.52	28.60
	SD	3.22	3.03	3.65	3.02	3.21	3.70	3.34	3.84	3.21	3.10	3.50	3.54	4.18	3.65
$24\zeta_{20}^{36}$	Av	43.74	**43.62**	44.46	44.44	46.58	49.08	46.46	49.12	45.48	45.94	47.06	49.14	49.50	49.44
	SD	8.27	8.36	8.30	8.34	9.01	9.23	8.84	9.13	8.36	8.27	9.09	9.58	9.58	9.69
$25\zeta_{50}^{40}$	Av	146.00	160.44	171.58	174.44	165.52	176.56	165.76	175.14	**144.04**	146.38	176.18	176.98	190.66	191.0
	SD	12.66	14.65	15.64	16.76	14.32	14.87	13.16	13.97	12.65	12.31	13.25	13.93	14.46	14.12
$30\zeta_{20}^{40}$	Av	**39.98**	40.02	40.68	40.76	42.40	45.64	42.60	45.0	41.16	41.82	43.24	45.16	45.68	45.34
	SD	4.37	4.32	4.60	4.57	4.68	5.12	4.71	5.00	4.26	4.67	4.79	4.92	5.20	5.06

Table 5: Results of Friedman and Iman-Davenport tests for $\alpha = 0.01$.

	Friedman value	critical χ^2 value	Iman-Davenport value	critical F_F value
all	320.30	45.64	50.20	1.80
top 7	47.10	16.81	14.45	3.01

Table 6: Results of Holm's test using MAHCP02 as control algorithm ($\alpha = 0.01$).

i	algorithm	z-statistic	p-value	α/i
1	MAHCP10	0.941065	0.173335	0.010000
2	MAHCF10	3.314184	0.000397	0.005000
3	MASAR02	3.355100	0.000460	0.003333
4	MAHCF02	3.436932	0.000294	0.002500
5	MASAR10	4.991734	< 0.000001	0.002000
6	TSP	5.441809	< 0.000001	0.001667

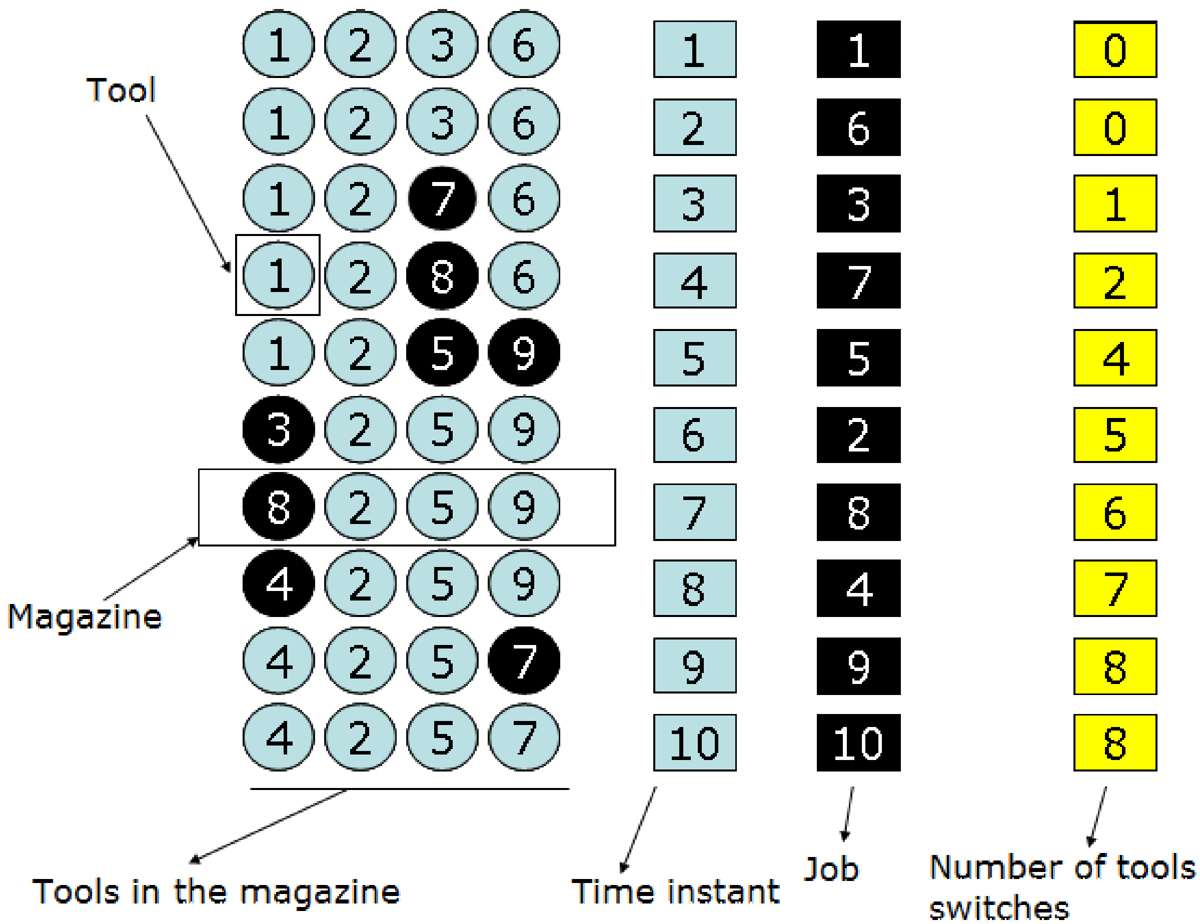

Figure 1: Example of the application of the KTNS policy. The tool requirements for each job are those indicated in Table 1. Slots in the magazine are denoted by circles (each row depicting the state of the magazine at a give time step). Black circles denote a tool switch. Finally the sequence of jobs is given by the dark squares, and the cumulative number of switches is indicated in the right side of the figure.

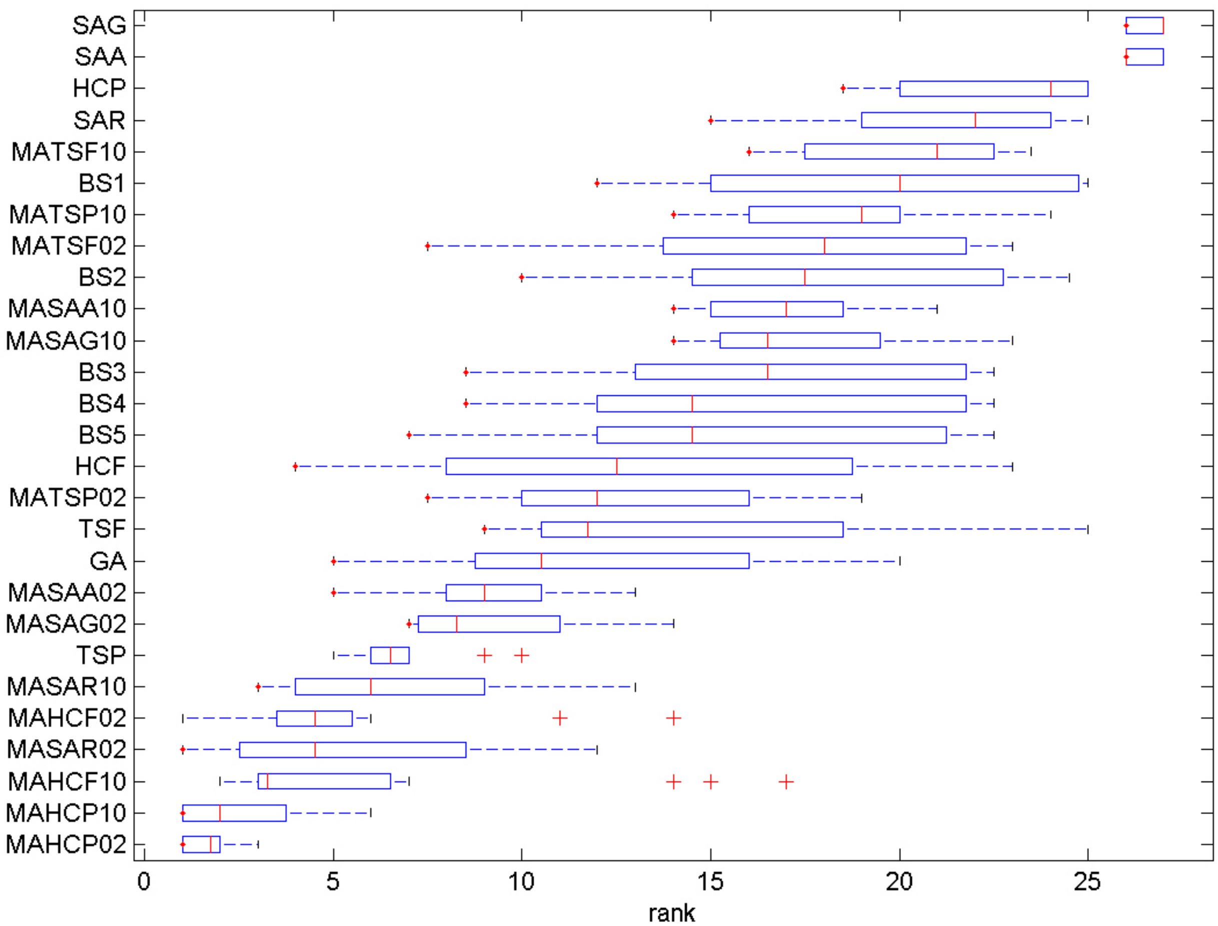

Figure 2: Rank distribution of each algorithm across all instances. As usual, each box comprises the second and third quartiles of the distribution, the median is marked with a vertical line, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign.