Hybrid cooperation models for the Tool Switching Problem

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

Abstract The Tool Switching Problem (ToSP) is a hard combinatorial optimization problem of relevance in the field of flexible manufacturing systems (FMS), that has been tackled in the literature using both complete and heuristic methods, including local-search metaheuristics, population-based methods and hybrids thereof (e.g., memetic algorithms). This work approaches the ToSP using several hybrid cooperative models where spatially-structured agents are endowed with specific local-search/population-based strategies. Issues such as the intervening techniques and the communication topology are analyzed via an extensive empirical evaluation. It is shown that the cooperative models provide better results than their constituent parts. Furthermore, they not only provide solutions of similar quality to those returned by the memetic approach but raise interest prospects with respect to its scalability.

1 Introduction

The uniform tool switching problem (ToSP) is a hard combinatorial optimization problem that appears in Flexible Manufacturing Systems (FMSs), an alternative to rigid production systems that has the capability to be adjusted for generating different products and/or for changing the order of product generation. This problem arises in a single 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

Jhon Edgar Amaya
Universidad Nacional Experimental del Táchira (UNET), Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela, e-mail: jedgar@unet.edu.ve

Carlos Cotta and Antonio J. Fernández Leiva
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain. e-mail: {ccottap,afdez}@lcc.uma.es

in the machine magazine. The 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. In this context, tool management is a challenging task that directly influences the efficiency of flexible manufacturing systems. Different examples of the problem can be found in diverse areas such as electronics manufacturing, metalworking industry, computer memory management, and aeronautics, among others [3, 4, 30, 32].

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 can be considered. In this line of work, [2] recently proposed three methods to tackle the ToSP: a simple local search (LS) scheme based on hill climbing, a genetic algorithm (which, as far as we know, constituted the first population-based approach to solve the uniform ToSP[1]), and a memetic algorithm [19, 26] (MA), based on the hybridization of the two latter methods. Related to this latter approach, this work proceeds along the cooperative side of hybridization by considering composite models in which different search techniques cooperate for solving the ToSP. These models can be arranged in a plethora of ways, and as a first step we have focused on the use of local-search metaheuristics as basic search strategies, and more precisely on how they can synergistically interact and the effect of the communication topology.

2 Background

Before proceeding, let us firstly describe more in depth the ToSP. Then, we will review previous related work.

2.1 The 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 for the optimization process is 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

[1] Note that genetic algorithms (GAs) have been applied to other variants of the problem –e.g., [17]– though.

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, and 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.

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 (resp. tools) with integers from $\mathbb{N}_n^+$ (resp. $\mathbb{N}_m^+$). An ILP formulation for the ToSP is shown below, using two sets of zero-one decision variables – x_{jk} ($j \in \mathbb{N}_n^+$, $k \in \mathbb{N}_n^+$), and y_{ik} ($i \in \mathbb{N}_m^+$, $k \in \mathbb{N}_n^+$) – that respectively indicate whether a job j is executed at time k or not, or whether a tool i is in the magazine at time k or not. Notice that since each job makes exclusive use of the machine, time-step k can be assimilated to the time at which the kth job is executed.

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 [3], 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:

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

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

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

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

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

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

This general definition shown above corresponds to the *uniform ToSP* in which each tool fits in just one slot. The ToSP can be divided into three subproblems [35]: 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). Moreover, and without loss

of generality, the cost of switching a tool is considered constant (the same for all tools) in the uniform ToSP. Under this assumption, the tool loading subproblem can also be obviated because 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) [3, 32]. The importance of this policy is that given a job sequence KTNS obtains its optimal number of tool switches. Therefore, we can concentrate on the machine loading subproblem, and use KTNS as a subordinate procedure to solve the subsequent tool loading subproblem.

2.2 Related Work on the ToSP

This paper focuses 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 1960's [4]; since then, the uniform ToSP has been tackled via many different techniques. The late 1980's contributed specially to solve the problem [3, 11, 18, 32]. This way, [32] proposed an ILP formulation of the problem, and [3] formulated the ToSP as a non-linear integer program with a dual-based relaxation heuristic. More recently, [20] proposed two exact algorithms: a branch-and-bound approach and a linear programming-based branch-and-cut algorithm.

Despite the moderate success of exact methods, it must be noted that they are inherently limited, since [27] and [8] proved formally that the ToSP is NP-hard for $C > 2$. This limitation was already highlighted by Laporte et al. [20] 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 (e.g., [10, 14, 30]).

The use of metaheuristics has been also considered recently. For instance, local search methods such as tabu search (TS) have been proposed [1, 13]. Among these, we find specifically interesting the approach presented by [1], 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 [36], who proposed a beam search algorithm. Beam search (BS) 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.

2.3 Background on Cooperative Models

Different schemes have been proposed for cooperating metaheuristics. For example, Toulouse *et al.* [33] considered using multiple instances of tabu search running in parallel, eventually exchanging some of the attributes stored in tabu memory. Later on, Toulouse *et al.* [34] proposed a a hierarchical cooperative model based on problem decomposition. Crainic and Gendreau [6] presented a cooperative parallel tabu search method for capacitated network design problem that was shown to outperform independent search strategies. Crainic *et al.* [7] also proposed a method for asynchronous cooperative multi-search using variable neighborhood search with application to the p-median problem. Pelta *et al.* [28] presented a cooperative multithread search-based optimization strategy, in which several solvers were controlled by a higher-level coordination algorithm which collected information on their search performance and altered the behavior of the solvers accordingly (see also [9]).

More recently, Lu *et al.* [23] presented a hybrid cooperative version of quantum particle swam optimization aimed to improving the diversity of the swarms. Another approach for the implementation of cooperative mechanisms with metaheuristics is multi-agent systems. Milano and Roli [25] developed a multi-agent system called MAGMA (multiagent metaheuristic architecture) allowing the use of metaheuristics at different levels (creating solutions, improving them, defining the search strategy, and coordinating lower-level agents). Malek [24] introduced a multi-agent system like MAGMA which considered particular metaheuristics implemented by individual agents and the exchange of solutions between these.

To the best of our knowledge, no cooperative scheme has been applied to tackle the ToSP, perhaps with the exception of our memetic proposal described in [2] that can be catalogued as an integrative cooperation according to the classification described in [29] (note at any rate that none of the techniques involved in the MA is a complete algorithm). In any case, no classical cooperation model in the sense of "search algorithms working in parallel with a varying level of communication" [5] has been tried. This paper presents the first cooperative models according to this mentioned schema for solving the ToSP.

3 Hybrid Cooperative Models

We have considered four collaborative architectures. In three of them, agents are attached to a certain spatial structure endowed with a LS mechanism. These architectures are defined on the basis of the particular LS methods used, and on their interaction topology. Therefore, these two aspects are defined separately in Sections 3.1 and 3.2 respectively. A fourth architecture, also described in Section 3.2, is defined on the basis of a model based in heterogeneous techniques for executing search, diversification and intensification.

3.1 Local searchers

LS metaheuristics are based on exploring the neighborhood of a certain "current" solution. It is thus convenient to address firstly the representation of solutions and the structure of this neighborhood, and subsequently of the underlying search space. 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 $\mathbb{N}_n^+$. The KTNS algorithm is used to obtain the actual tool configuration of the machine for the corresponding job sequence.

Having defined the representation, we now turn our attention to the neighborhood structure. In general, we have considered 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. For some specific applications (named when necessary), we have also considered a specific neighborhood called *block* neighborhood $\mathcal{N}_{block}(\cdot)$. This is a generalization of the swap neighborhood in which two non-overlapping blocks (i.e., subsequences of adjacent positions) of a randomly chosen length $b_l \in \mathbb{N}_{n/2}^+$ are selected at random within a permutation, and swapped.

These neighborhoods are exploited within two different LS frameworks. The first one is steepest-ascent Hill Climbing (HC), in which given a current solution π, its neighborhood $\mathcal{N}(\pi)$ is 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 second LS technique considered is a Tabu Search (TS) method along the lines of the proposal in [1]. This TS method is based on a strategic oscillation mechanism which switches between the two neighborhoods defined before. A deterministic criterion based on switching the neighborhood structure after a fixed number of iterations was reported by [1] to perform better than a probabilistic criterion (i.e., choosing the neighborhood structure in each step, according to a certain probability distribution). We implement a long term memory scheme using a frequency based memory structure with a mechanism based in swapping to select new candidate solutions [1]. No aspiration criterion is used in this referred algorithm.

3.2 Interaction Topology

Let R be an architecture with n agents; each agent a_i ($0 \leqslant i \leqslant n-1$) in R consists of one of the metaheuristics described in Sect. 3.1. These agents engage in peri-

Algorithm 1: COOPERATIVE-MODEL$_n$

```
1  for i ∈ ℕ_n^+ do
2  |  S_i ← GenerateInitialSolution();
3  endfor
4  cycles ← 1;
5  while cycles ⩽ cycles_max do
6  |  for i ∈ ℕ_n^+ do
7  |  |  S_i ← a_i(S_i);
8  |  endfor
9  |  for (i, j) ∈ T_R do
10 |  |  if KTNS(S_i) < KTNS(S_j) then
11 |  |  |  S_j ← S_i;
12 |  |  endif
13 |  endfor
14 |  cycles ← cycles + 1
15 endw
16 return max^{-1}{KTNS(S_i) | i ∈ ℕ_n^+};
```

ods of isolated exploration followed by synchronous communication. We denote as $cycles_{\max}$ the maximum number of such exploration/communication cycles in a certain cooperative model. Also, let S_i be the best solution found by agent a_i, and let $\mathbf{T}_R \subseteq \mathbb{N}_n^+ \times \mathbb{N}_n^+$ be the communication topology over R (i.e., if $(i,j) \in \mathbf{T}_R$ then a_i can send information to agent a_j). The general architecture of the model is then described in Algorithm 1. Firstly all the agents are initialized with random initial solution (lines 1-3). Then, the algorithm is executed for a maximum number of iterations cycles (lines 5-15) where, in each cycle, a local improvement of the solution kept in each agent is done (lines 6-8), and solutions are fed from an agent to another according to the topology considered (lines 9-13). Note that an agent only accepts an incoming solution if it is better than its incumbent. Observe also that, for a maximum number of evaluations $E_{\max}$ and for a specific number of cycles $cycles_{\max}$, each cycle in our cooperative algorithms spends $E_{cycle} = E_{\max}/cycles_{\max}$ evaluations, and the specific LS method of any agent takes E_{cycle}/n evaluations at most.

Three strategies based on different interaction topologies are considered:

- RING: $\mathbf{T}_R = \{(i, i(n)+1) \mid i \in \mathbb{N}_n^+ \text{ and } i(n) \text{ denotes } i \text{ modulo } n\}$. Thus, there exists a circular list of agents in which each node only sends (resp. receives) information to its successor (resp. from its predecessor).
- BROADCAST: $\mathbf{T}_R = \mathbb{N}_n^+ \times \mathbb{N}_n^+$, i.e., a *go with the winners*-like topology in which the best overall solution at each synchronization point is transmitted to all agents. This means all agents executes intensification over the same local region of the search space at the beginning of each cycle.
- RANDOM: $\mathbf{T}_R$ is composed by n pairs (i,j) that are randomly sampled from $\mathbb{N}_n^+ \times \mathbb{N}_n^+$. This sampling is done each time communication takes place, and hence any two agents might eventually communicate in any step.

In addition to these strategies we have considered a so-called Ring SDI model, based on an interesting proposal described in [31]. SDI stands for Search, Diversification and Intensification, and hence the SDI architecture consists of three agents dedicated to distinct purposes: the first one to local search, the second one to diversification and the third one to intensification. As described in next section, within this SDI model the intervening techniques are not just local searchers, but other techniques can be used for intensification/diversification purposes.

4 Computational Results

As far as we know, no standard data instance exists for this problem (at least publicly available) so that we have selected a wide set of problem instances that were attacked in [1, 3, 14, 36]; more specifically, 16 instances were chosen with values for the number of jobs, number of tools, and machine capacity ranging in [10,50], [9,60] and [4,25] respectively. Table 1 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 1 Problem Instances considered in the experimental evaluation. The minimum and maximum of tools required for all the jobs is indicated in second and third rows respectively. Fourth row shows the work from which the problem instance was obtained.

	$4\zeta_{10}^{10}$	$4\zeta_{10}^{9}$	$6\zeta_{10}^{15}$	$6\zeta_{15}^{12}$	$6\zeta_{15}^{20}$	$8\zeta_{20}^{15}$	$8\zeta_{20}^{16}$	$10\zeta_{20}^{20}$	$24\zeta_{20}^{30}$	$24\zeta_{20}^{36}$	$30\zeta_{20}^{40}$	$10\zeta_{30}^{25}$	$15\zeta_{30}^{40}$	$15\zeta_{40}^{30}$	$20\zeta_{40}^{60}$	$25\zeta_{50}^{40}$
Min.	2	2	3	3	3	3	3	4	9	9	11	4	6	6	7	9
Max.	4	4	6	6	6	8	8	10	24	24	30	10	15	15	20	20
Source	[14]	[3]		[3]			[3]	[3]	[3]	[3]						
	[1]	[36]	[36]	[36]	[14]	[1]	[36]	[36]	[36]	[36]	[36]	[1]	[14]	[1]	[14]	[1]

Five different datasets[2] (i.e., incident matrixes or relations among tools and jobs) were generated randomly per instance. Each dataset was generated with the restriction, already imposed in previous works such as [14], that no job is *covered* by any other job in the sense that $\forall i, j \in \mathbb{N}_n^+, i \neq j, T^{(J_i)} \not\subseteq T^{(J_j)}$. The reason to enforce this constraint is to avoid the simplification of the problem by preprocessing techniques as done for instance in [3] and [36].

The experiments have been performed using a wide set of different algorithms: the beam search (BS) presented in [36], three LS methods, a GA, the memetic approach (denoted as MaHC) presented in [2], and the four cooperative algorithms described in this paper. From these, a wide number of algorithms were devised and tested. For instance, in the case of BS, five different values $\beta \in \mathbb{N}_5^+$ were considered for the beam width. Regarding LS methods, we consider the TS proposed in [1], and

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

HC as described previously. Moreover, we have taken into account also LS versions in which a partial exploration of the neighborhood is done by obtaining a fixed-size random sample; in particular, 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 (resp. TSP and TSF) is used to indicate the HC variant (resp. TS variant) in which the neighborhood is partially or fully explored respectively. Also, 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. Regarding TS, the tabu tenure is 5, and the number of iterations on each neighborhood for performing strategic oscillation is 3. In both cases, this corresponds to the setting used by [1]. The GA is a steady-state genetic algorithm whose parameters are exactly as those described in [2], that is to say, $popsize = 30$, $p_X = 1.0$, and $p_M = 1/n$ where n is the number of jobs, with binary tournament selection; alternating position crossover (APX) is used [21], and mutation is done by applying the random block swap as operator. The MaHC consists of a combination of this GA with HCP where HCP was always applied to each offspring generated after the mutation step. The election of the parameter values (including the value for α) was done after an extensive phase of experimentation with many different values. The best combinations of the values were finally selected.

Regarding the cooperative models, we have used $cycles_{\max} \in \{3,4,5\}$, and have focused on models with 3 agents to make easier the comparison with the SDI model. In this latter RINGSDI model we connect HCP for LS, GA for diversification, and for intensification we plug in the *KickOperator* that was also used in [31]. In our rendition of this operator it acts as a first-ascent HC on the swap neighborhood. As to the basic RING, BROADCAST and RANDOM topologies, their three agents were loaded with HCF, HCP and TSP techniques respectively.

All algorithms were run 10 times (per instance and dataset) and a maximum of $E_{\max} = \varphi n(m-C)$ 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 execution per dataset (and per value of beam width) was run and the algorithm was allowed to be executed until exhaustion (i.e., until completing the search).

Due to space limitations we will not present all the obtained results for each of the instances and for all the algorithms involved in the comparison, and will use a rank-based approach in order to analyze the significance of the results. 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 $k = 23$ is the number of algorithms; in case of ties, an average rank is awarded). The distribution of these ranks is shown in Fig. 1. Here one can extract important conclusions: the most important is that

[3] Observe that the number of evaluations depends directly on the number of jobs although it seems evident that the problem difficulty lies in the relation between number of tools and magazine capacity. In this sense, the number of evaluations increases with the number of tools (assumed to be directly related with problem difficulty) and decreases when the magazine capacity increases (that, in some sense, it is also inversely related to the problem difficulty).

in general, the cooperative models behaves better than its constituent parts. This is an important fact as the cooperative models have not been optimized exhaustively (due to the high number of possible metaheuristics combinations to be loaded in the agents). Also, the fact that RINGSDI is better than RING might indicate the need for a diversification algorithm to increase the area of exploration in the search landscape.

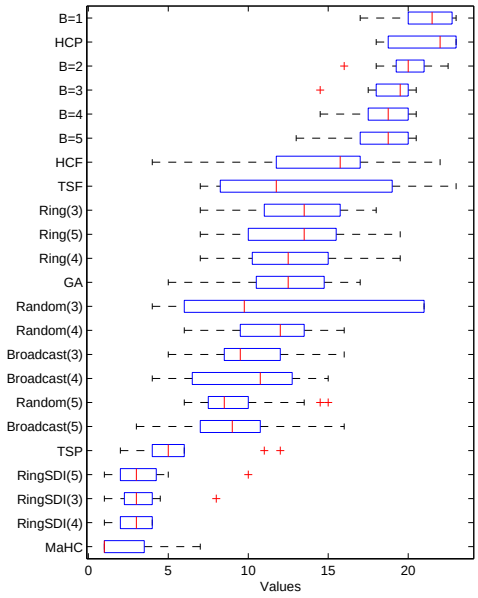

Fig. 1 Rank distribution of each algorithm across all instances. As usual, each box comprises the second and third quartiles of the distribution, the median is marked with a vertical line, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign. The numbers in parentheses indicate the number of cycles of execution (i.e., $cycles_{max}$).

Next, we have used two well-known non-parametric statistical tests [22] to compare ranks, namely Friedman test [12] and Iman-Davenport test [16]. The results are shown in Table 2. 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 5 algorithms (i.e., the MaHC, all RINGSDI and TS), whose performance places them in a separate cluster from the remaining algorithms. Again, it can be seen that the statistical test is passed, thus indicating significant differences in their ranks at the standard $\alpha = 0.05$ level.

Subsequently, we have focused in these top 5 algorithms, and performed Holm's test [15] in order to determine whether there exists significant differences with re-

Table 2 Results of Friedman and Iman-Davenport tests.

	Friedman value	critical χ^2 value	Iman-Davenport value	critical F_F value
all	269.80	33.92	49.23	1.57
top 5	19.04	9.49	6.35	2.53

spect to a control algorithm (in this case MaHC, the algorithm with the best mean rank). The results are shown in Table 3. Notice that the test is passed for all algorithms with respect to MaHC and that there is no statistical difference between MaHC, RINGSDI(4) and RINGSDI(3).

Table 3 Results of Holm's test using *MaHC* as control algorithm.

i	algorithm	z-statistic	p-value	$\alpha/(k-i)$
1	RingSDI(4)	1.286	0.09926	0.013
2	RingSDI(3)	1.957	0.02520	0.016
3	RingSDI(5)	2.012	0.02209	0.025
4	TSP	4.249	< 0.00001	0.050

Table 4 Computational results. Best results (in terms of the best solution average) are underlined and in boldface.

		$4\zeta_{10}^{10}$	$6\zeta_{10}^{15}$	$4\zeta_{10}^{9}$	$6\zeta_{15}^{12}$	$6\zeta_{15}^{20}$	$8\zeta_{20}^{15}$	$8\zeta_{20}^{16}$	$10\zeta_{20}^{20}$	$24\zeta_{20}^{30}$	$24\zeta_{20}^{36}$	$30\zeta_{20}^{40}$	$10\zeta_{30}^{25}$	$15\zeta_{30}^{40}$	$15\zeta_{40}^{30}$	$20\zeta_{40}^{60}$	$25\zeta_{50}^{40}$
TSP	mean	8.8	13.68	8.08	16.46	23.02	23.62	27.92	30.72	25.04	45.9	42.12	67.72	101.72	101.9	213.74	153.58
	σ	1.61	2.1	0.74	1.93	2.0	3.63	2.13	2.5	3.02	8.98	4.34	1.52	13.07	8.14	8.38	12.89
	best	7	11	7	13	20	18	23	26	21	34	33	65	82	89	199	130
MaHC	mean	**8.68**	13.7	**7.86**	**15.5**	**22.38**	**22.36**	**26.66**	**29.92**	24.9	46.54	**41.04**	**64.92**	**100.86**	**97.96**	211.88	153.36
	σ	1.62	2.09	0.721	1.982	1.938	3.576	1.986	2.357	3.28	8.81	4.54	1.573	12.9	7.887	7.812	13.52
	best	7	11	7	12	20	17	23	26	20	36	31	62	81	86	201	132
RingSDI (3)	mean	8.72	13.74	7.9	16.14	23.0	23.34	27.6	30.72	24.76	**45.26**	41.3	66.98	102.66	99.52	210.68	148.14
	σ	1.64	2.04	0.68	1.85	2.21	3.36	2.21	2.31	3.17	8.46	4.43	2.64	13.03	7.71	7.79	11.75
	best	7	11	7	13	20	19	23	26	20	35	32	62	82	87	197	129
RingSDI (4)	mean	8.72	**13.66**	7.9	16.16	22.96	23.06	27.44	30.72	**24.46**	45.84	41.8	67.32	102.28	99.16	211.08	**146.34**
	σ	1.65	2.12	0.73	1.9	2.14	3.34	2.05	2.41	3.51	7.91	4.79	2.69	12.37	7.64	9.34	12.28
	best	7	11	7	13	20	17	23	25	19	36	32	61	79	88	198	126
RingSDI (5)	mean	8.7	13.74	7.92	16.1	23.1	23.2	27.28	30.74	**24.46**	45.64	41.58	68.1	102.2	100.24	**210.44**	146.98
	σ	1.62	2.07	0.69	2.02	2.08	3.42	2.3	2.38	3.13	8.55	4.5	2.76	12.78	8.07	9.17	12.12
	best	7	11	7	12	20	18	22	25	20	33	32	63	81	87	195	124

Also, Table 4 shows the obtained results, grouped by problem instances, for these top 5 algorithms. One can observe that all RINGSDI algorithms perform better than MaHC in several instances, particularly in the largest one (i.e., last column), in which Wilcoxon's ranksum test indicates that RingSDI[4] significantly outperforms (at the standard 0.05 level) the MA in all five datasets generated for this parameter combination. This raises interest prospects for the scalability of these models, thus hinting the need for experiments at a larger scale to confirm this.

[4] All tables are available in *http://www.unet.edu.ve/∼jedgar/ToSP/Wilcoxon.htm*

5 Conclusions

Collaborative optimization models constitute a very appropriate framework for integrating different search techniques. Each of these techniques has a different view of the search landscape, and by combining the corresponding different exploration patterns, the search can benefit from an increased capability for escaping from local optima. Of course, this capability is more useful whenever the problem tackled poses a challenging optimization task to the individual search algorithms. Otherwise, computational power is diversified in unproductive explorations.

We have tackled here the tool switching problem and have proposed four cooperative methods to attack it. An empirical evaluation was executed in order to prove the validity and performance of the proposed techniques. One topology based on heterogenous intervening techniques (RINGSDI) provides better computational results than well-known algorithms for solving the ToSP, i.e., beam search and tabu search, and does not perform worse than a memetic algorithm. Indeed, some results with larger instances lead us to hypothesize that this model might have better scalability properties than the MA. This issue will be analyzed in future work.

We believe that there is room for improvement. For instance, it would be interesting to test other alternatives to LS. More precisely, the MA is a killer approach for the ToSP, so it may be interesting to include this technique in the cooperative model. In this case, it would be necessary to re-balance the intensification/exploration ratio, since MAs perform a much more intensified search than other techniques, and thus may require a more explorative counterweight. This line of research is in progress.

Acknowledgements.

The authors wish to thank the anonymous reviewers for their constructive comments and suggestions, which have improved the readability of the paper. The second and third authors were partially supported by Spanish MICINN project under contract TIN2008-05941 (NEMESIS project).

References

[1] 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

[2] Amaya J, Cotta C, Fernández A (2008) A memetic algorithm for the tool switching problem. In: Blesa M, et al (eds) Hybrid Metaheuristics 2008, Springer-Verlag, Málaga, Spain, Lecture Notes in Computer Science, vol 5296, pp 190–202

[3] Bard JF (1988) A heuristic for minimizing the number of tool switches on a flexible machine. IIE Transactions 20(4):382–391

[4] Belady L (1966) A study of replacement algorithms for virtual storage computers. IBM Systems Journal 5:78–101
[5] Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv 35(3):268–308
[6] Crainic TG, Gendreau M (2002) Cooperative parallel tabu search for capacitated network design. Journal of Heuristics 8(6):601–627
[7] Crainic TG, Gendreau M, Hansen P, Mladenović N (2004) Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10(3):293–314
[8] 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
[9] Cruz C, Pelta DA (2009) Soft computing and cooperative strategies for optimization. Applied Soft Computing 9(1):30–38
[10] Djellab H, Djellab K, Gourgand M (2000) A new heuristic based on a hypergraph representation for the tool switching problem. International Journal of Production Economics 64(1-3):165–176
[11] ElMaraghy H (1985) Automated tool management in flexible manufacturing. Journal of Manufacturing Systems 4(1):1–14
[12] 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
[13] 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
[14] 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
[15] Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6:6570
[16] Iman R, Davenport J (1980) Approximations of the critical region of the Friedman statistic. Communications in Statistics 9:571–595
[17] Keung KW, Ip WH, Lee TC (2001) A genetic algorithm approach to the multiple machine tool selection problem. Journal of Intelligent Manufacturing 12(4):331–342
[18] Kiran A, Krason R (1988) Automated tooling in a flexible manufacturing system. Industrial Engineering 20:52–57
[19] 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
[20] Laporte G, Salazar-González J, Semet F (2004) Exact algorithms for the job sequencing and tool switching problem. IIE Transactions 36(1):37–45
[21] 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

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

[23] Lu S, Sun C (2008) Coevolutionary quantum-behaved particle swarm optimization with hybrid cooperative search. In: Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Applications PACIIA'08, vol 1, pp 109–113

[24] Malek R (2009) Collaboration of metaheuristic algorithms through a multi-agent system. In: HoloMAS'09: Proceedings of the 4th International Conference on Industrial Applic. of Holonic and Multi-Agent Systems, Springer-Verlag, Berlin, Heidelberg, pp 72–81

[25] Milano M, Roli A (2004) Magma: a multiagent architecture for metaheuristics. IEEE Transactions on Systems, Man, and Cybernetics, Part B 34(2):925–941

[26] Moscato P, Cotta C (2007) Memetic algorithms. In: González T (ed) Handbook of Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC Press, chap 27

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

[28] Pelta D, Cruz C, Sancho-Royo A, Verdegay J (2006) Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. Information Sciences 176:1849–1868

[29] Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: Mira J, Álvarez JR (eds) IWINAC 2005, Springer, Las Palmas, Canary Islands, Spain, Lecture Notes in Computer Science, vol 3562, pp 41–53

[30] 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

[31] Talbi EG, Bachelet V (2006) Cosearch: A parallel cooperative metaheuristic. Journal of Mathematical Modelling and Algorithms 5(1):5–22

[32] 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

[33] Toulouse M, Crainic TG, Sanso B, Thulasiraman K (1998) Self-organization in cooperative tabu search algorithms. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol 3, pp 2379–2384

[34] Toulouse M, Thulasiraman K, Glover F (1999) Multi-level cooperative search: A new paradigm for combinatorial optimization and an application to graph partitioning. In: Amestoy P, et al (eds) 5th International Euro-Par Conference, Toulouse, Springer, Lecture Notes in Computer Science, vol 1685, pp 533–542

[35] 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

[36] Zhou BH, Xi LF, Cao YS (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