A Memetic Cooperative Optimization Schema and its Application to the Tool Switching Problem

Jhon Edgar Amaya[1], Carlos Cotta[2], and Antonio J. Fernández Leiva[2]

[1] Universidad Nacional Experimental del Táchira (UNET)
Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela
jedgar@unet.edu.ve
[2] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain.
{ccottap,afdez}@lcc.uma.es

Abstract. This paper describes a generic (meta-)cooperative optimization schema in which several agents endowed with an optimization technique (whose nature is not initially restricted) cooperate to solve an optimization problem. These agents can use a wide set of optimization techniques, including local search, population-based methods, and hybrids thereof, hence featuring multilevel hybridization. This optimization approach is here deployed on the Tool Switching Problem (ToSP), a hard combinatorial optimization problem in the area of flexible manufacturing. We have conducted an ample experimental analysis involving a comparison of a wide number of algorithms or a large number of instances. This analysis indicates that some meta-cooperative instances perform significantly better than the rest of the algorithms, including a memetic algorithm that was the previously incumbent for this problem.

1 Introduction and related work

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.

Different schemes have been proposed for cooperating metaheuristics. For example, Toulouse *et al.* [1] considered using multiple instances of tabu search (TS) running in parallel, eventually exchanging some of the attributes stored in tabu memory. Crainic and Gendreau [2] presented a cooperative parallel TS method that was shown to outperform independent search strategies. Crainic *et al.* [3] also proposed a method for asynchronous cooperative multi-search using variable neighborhood search (VNS). Pelta *et al.* [4] presented a cooperative

multi-thread 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. Milano and Roli [5] developed a multi-agent system called MAGMA (MultiAGent Metaheuristic Architecture) in which metaheuristics are used at different levels (creating solutions, improving them, defining the search strategy, and coordinating lower-level agents). Recently, Amaya et al. [6] have proposed agent topologies equipped with local search (LS) methods based on simple structures of communication similar to those used in the computer networks. More specifically, [6] proposed four different cooperative models (i.e., RING,BROADCAST, RANDOM, and a so-called Ring SDI model) to handle the uniform tool switching problem (ToSP).

In this paper we go a step beyond and have generalized the first model (i.e., RING) described in [6]; the result is a generic schema whose instances produce meta-cooperative architectures in which one or more agents can also be loaded with a cooperative optimization technique. This schema is not specific for a particular optimization problem and thus can be applied to many combinatorial optimization problems. To demonstrate both the adequacy of the schema and the goodness of its instances (as meta-cooperative algorithms) we have also conducted an empirical evaluation on the ToSP.

2 Ring-Based (Meta-)Cooperative Model

Let us denote $\mathbb{N}_n^+ = \{1, \cdots, n\}$. The optimization architecture proposed is shown in Algorithm 1. As it can be seen, it features an architecture R with n agents connected in form of a ring; each agent a_i $(0 \leqslant i \leqslant n-1)$ in R consists of an optimization method (e.g., any metaheuristic, such as a local searcher, a population-based method, or even a hybrid thereof). Observe that there exists a circular list of agents in which each node only sends (resp. receives) information to its successor (resp. from its predecessor). The agents in the architecture engage in periods 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 set of candidates solutions managed by agent a_i; note that the nature of S_i is variable (e.g., if a_i is a population-based method this means that S_i is its corresponding population whereas if a_i is a loaded with a trajectory-based method, then S_i just contains one candidate).

Firstly all the agents are initialized with a set of initial candidate solutions (lines 1-3, function GENERATECANDIDATESET). The size of this set depends on the technique endowed in the agent (e.g., it might be a population or just one single candidate). Then, the algorithm is run for a maximum number of iterations cycles (lines 5-16) where, in each cycle, an optimization phase of the specific candidate set kept in each agent is done (lines 6-9) and the best solution obtained in each agent (line 8, function BESTCANDIDATEIN) is sent to its successor agent if this solution is better than the best one obtained in the successor agent (lines 10-

Algorithm 1: RING-BASED (META-)COOPERATIVE MODEL$_n$

```
1  for i ∈ ℕ_n^+ do
2  |  S_i ← GenerateCandidateSet();
3  end for
4  cycles ← 1;
5  while cycles ⩽ cycles_max do
6  |  for i ∈ ℕ_n^+ do
7  |  |  S_i ← a_i(S_i);
8  |  |  b_i ← BestCandidateIn(S_i);
9  |  end for
10 |  for (i, j) ∈ {(i, i(n) + 1) | i ∈ ℕ_n^+ and i(n) denotes i modulo n} do
11 |  |  if fitness(b_i) < fitness(b_j) then
12 |  |  |  Replace worst candidate in S_j with b_i;
13 |  |  end if
14 |  end for
15 |  cycles ← cycles + 1
16 end while
17 return max^{-1}{fitness(BestCandidateIn(S_i)) | i ∈ ℕ_n^+};
```

14). Note that an agent only accepts an incoming solution if it is better than the best one kept in its candidate set (lines 11-13) . 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 optimization method of any agent takes E_{cycle}/n evaluations at most.

Multiple variants of this cooperative schema can be devised from the general schema shown above as no specific mention is done about the type of the agents involved in the architecture.

3 Experimental Results

To test the feasibility of the proposed architecture we consider the uniform tool switching problem (ToSP) as a benchmark. This section is thus devoted to present the problem and provide a formal description of it. Then, an experimental analysis that includes a wide number of optimization techniques is shown.

3.1 The ToSP

The uniform tool switching problem (ToSP) is a hard combinatorial optimization problem that can be found in flexible manufacturing systems (FMSs) and diverse areas such as electronics manufacturing, metalworking industry, computer memory management, and aeronautics, among others [7–10]. This problem occurs 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 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. Therefore management tool directly affects the efficiency of FMS. The ToSP has been tackled through different methods such as exact methods [8], LS methods, population-based optimization methods [11], and even cooperative models [6]. [12] and [13] proved formally that the ToSP is NP-hard for $C > 2$ and thus exact methods are inherently limited.

Following the previous description of the uniform ToSP, 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 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 $T^{(J_{i_j})} \subseteq T_j$.

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 [8], 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:

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

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

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

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

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

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

This general definition shown above corresponds to the *uniform ToSP* in which each tool fits in just one slot. The uniform ToSP considered the cost of switching a tool constant (the same for all tools) and computing the cost of a job sequence by means of a greedy procedure termed *Keep Tool Needed Soonest* (KTNS) [8, 9]. The importance of this policy is that given a job sequence KTNS obtains its optimal number of tool switches in polynomial time. Therefore, we can concentrate on determining the sequence of jobs, and use KTNS as a subordinate procedure to decide where (i.e., in which slot) to place each tool.

3.2 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 [8, 14–16]; 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]	[8]		[8]			[8]	[8]	[8]	[8]						
	[15]	[16]	[16]	[16]	[14]	[15]	[16]	[16]	[16]	[16]	[16]	[15]	[14]	[15]	[14]	[15]

Five different datasets[3] (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

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

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 [8] and [16].

The experiments have been performed using a wide set of different algorithms. In particular we have considered deterministic methods, local search (LS) techniques, cooperative methods, and a number of meta-cooperative algorithms devised from the schema shown in Algorithm 1. From these, a number of variants have also been considered. For instance, as deterministic method we have considered the beam search (BS) algorithm presented in [16]; this algorithms admits a parameter β (termed beam-widht) for which we have considered five different values $\beta \in \mathbb{N}_5^+$. As to LS methods, we have considered two of them: (1) The tabu search versions (TSP and TSF) specialized for the ToSP and described in [6]; here *P and *F is used to indicate the algorithmic variant in which the neighborhood is partially or fully explored respectively (the interested reader is referred to [6] for more details), and (2) the steepest-ascent Hill Climbing (HC) method presented in [11]; from this we have also devised two versions HCP and HCF following the same principles of partial/full exploration mentioned previously.

As cooperative techniques we have considered the memetic algorithm (MA) presented in [11] (denoted as MAHCP because it is a combination of a genetic algorithm (GA) and the method HCP mentioned previously). In [6] we shown that this MA was a killer approach for the ToSP (beating to a number of cooperatives models in which all agents where loaded with LS techniques). We have also included in the comparison a new MA denoted as MATSP because it is a combination of a GA and the TSP method mentioned previously (the parameters were the same as those indicated in [11] for the MAHCP). In these two MAs the LS techniques were always applied to each offspring generated after the mutation step. Other parameters are: $popsize = 30$, $p_X = 1.0$, and $p_M = 1/n$ where n is the number of jobs, with binary tournament selection; alternating position crossover (APX) is used [17], and mutation is done by applying the random block swap as operator (see [11] for more details).

Regarding the meta-cooperative model, we have devised 10 different instances from the schema shown in Algorithm 1 where $n = 3$ and $cycles_{\max} \in \{4, 5\}$; GenerateCandidateSet represents a random initialization, and fitness is defined as the KTNS method described in Section 3.1. In all the instances, at least one agent has been loaded with a cooperative optimization technique, in particular with one of the two MAs mentioned above (i.e., MAHCP or MATSP). In the rest of the paper we have used the notation U(*dd*,*ee*,*ff*,*xx*) to represent an instance of three agents loaded with techniques *dd*, *ee*, *ff* and where *xx* is the number of cycles considered. All algorithms were run 10 times (per instance and dataset) and a maximum of $E_{\max} = \varphi n(m - C)$ evaluations, cf. [6]. 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 $k = 21$ is the number of algorithms; in case of ties, an average rank is awarded). The distribution of these ranks is shown in Fig. 1.

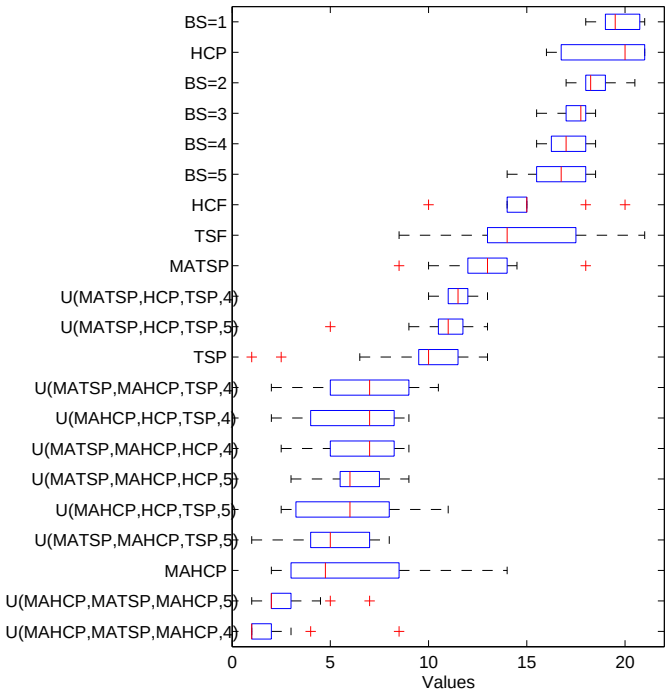

Fig. 1. Rank distribution of each algorithm across all instances. As usual, each box comprises the second and third quartiles of the distribution, the median is marked with a vertical line, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign.

Next, we have used two well-known non-parametric statistical tests [18] to compare ranks, namely Friedman test [19] and Iman-Davenport test [20]. 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 4 algorithms (i.e., U(MAHCP,MATSP,MAHCP,4), U(MAHCP,MATSP,MAHCP,5), MAHCP,, and U(MATSP,MAHCP,TSP,5)). Again, it can be seen that the sta-

tistical test is passed, thus indicating significant differences in their ranks at the standard $\alpha = 0.05$ level.

Table 2. Results of Friedman and Iman-Davenport tests.

	Friedman value	critical χ^2 value	Iman-Davenport value	critical F_F value
all	277.30	31.41	97.41	1.61
top 4	41.44	7.81	94.71	2.81

Subsequently, we have focused in these top 4 algorithms, and performed Holm's test [21] in order to determine whether there exists significant differences with respect to a control algorithm (in this case U(MAHCP,MATSP,MAHCP,4), the algorithm with the best mean rank). The results are shown in Table 3. The test indicates there exists a significant difference between the control algorithm and both MAHCP and U(MATSP,MAHCP,TSP,5), but not with respect to U(MAHCP,MATSP,MAHCP,5) (at the 0.05 level; the p-value is only slightly above this value though).

Table 3. Results of Holm's test using U(MAHCP,MATSP,MAHCP,4) as control algorithm.

i	algorithm	z-statistic	p-value	$\alpha/(k-i)$
1	U(MAHCP,MATSP,MAHCP,5)	1.369	0.0855	0.017
2	MAHCP	3.834	$6.3e-5$	0.025
3	U(MATSP,MAHCP,TSP,5)	4.108	$1.9e-5$	0.050

Table 4. Computational results. Best results (in terms of the best solution average) are underlined and in boldface. U_1 = U(MAHCP,MATSP,MAHCP,4), U_2 = U(MAHCP,MATSP,MAHCP,5), U_3 = U(MATSP,MAHCP,TSP,5) and MA = MAHCP. $\overline{x}$, σ and b denotes the mean, standard deviation and best values respectively.

		$4\varsigma_{10}^{10}$	$4\varsigma_{10}^{9}$	$6\varsigma_{10}^{15}$	$6\varsigma_{15}^{12}$	$6\varsigma_{15}^{20}$	$8\varsigma_{20}^{15}$	$8\varsigma_{20}^{16}$	$10\varsigma_{20}^{20}$	$24\varsigma_{20}^{30}$	$24\varsigma_{20}^{36}$	$30\varsigma_{20}^{40}$	$10\varsigma_{30}^{25}$	$15\varsigma_{30}^{40}$	$15\varsigma_{40}^{30}$	$20\varsigma_{40}^{60}$	$25\varsigma_{50}^{40}$
U1	$\overline{x}$	**8.78**	**7.9**	13.82	**15.92**	22.98	**22.66**	26.96	**30.18**	24.42	**44.82**	**40.6**	**64.2**	**99.1**	**95.08**	**205.78**	**143.22**
	σ	1.72	0.81	2.01	1.86	1.92	3.22	2.07	2.16	3.48	8.46	4.34	2.23	12.39	8.09	7.82	11.46
	b	7	7	11	13	20	17	22	26	19	35	32	60	80	82	194	125
U2	$\overline{x}$	8.86	7.98	13.76	16.12	22.84	22.9	**26.78**	30.26	**24.34**	44.92	41.04	64.3	98.64	95.46	206.0	144.72
	σ	1.71	0.79	2.11	1.82	2.04	3.37	1.96	2.38	3.1	8.02	4.66	1.93	12.41	7.62	7.92	11.67
	b	7	7	11	12	20	18	23	25	21	35	31	60	79	83	192	128
MA	$\overline{x}$	8.94	8.1	13.89	16.26	23.18	22.86	27.24	30.53	24.78	44.87	41.3	64.32	99.7	95.86	206.3	144.18
	σ	1.62	0.75	1.99	1.79	1.96	3.41	2.22	2.49	3.29	7.55	4.41	2.4	12.82	7.52	8.81	11.94
	b	7	7	11	12	20	17	22	26	20	35	31	59	80	80	193	122
U3	$\overline{x}$	8.86	7.98	**13.7**	16.28	**22.82**	23.02	27.08	30.48	24.84	45.2	41.52	65.52	100.06	97.1	207.38	145.48
	σ	1.69	0.73	2.06	1.77	2.17	3.72	2.12	2.74	3.13	8.49	4.69	2.86	12.77	7.73	9.89	11.72
	b	7	7	11	13	20	17	22	25	21	33	31	59	81	85	191	127

Also, analyzing the obtained results, grouped by problem instances (see Table 4 for the results of these top 4 algorithms), one can observe that the two best meta-cooperative models (i.e., U_1 and U_2) outperform MAHCP (the previous incumbent for this problem) in all the problem instances.

4 Conclusions

In this work we have proposed a memetic cooperative architecture where several agents endowed with MAs and other techniques cooperate in solving a certain optimization problem. This model takes advantage of maintaining a high diversity of possible solutions as well as providing a certain degree of independence in the exploration of different regions of the search space as in island model-based evolutionary systems (although the former is much more flexible since it does not depend solely on population-based algorithms, and tries to exploit the synergy between different search techniques).

The results obtained show the effectiveness of the model on the ToSP, a very hard combinatorial problem related to flexible manufacturing. As expected, the experimentation indicates the choice of heuristic combinations, as well as the number of cycles used in the meta-cooperative model, are crucial parameters. Combinations including several memetic algorithms endowed with both TS and HC have been shown to provide the best results, with statistical significance with respect to other models (including a single MA that was the previous incumbent for this problem). Determining the proper values of some of the parameters (such as the number of agents, number of cycles for communication, the probability of acceptance of solutions, communication topology) in the ToSP and other problems is a line of future work.

Acknowledgements.

We thank the reviewers for their valuable comments and suggestions. The second and third authors were partially supported by Spanish MICINN project under contract TIN2008-05941 (NEMESIS project).

References

1. Toulouse, M., Crainic, T.G., Sanso, B., Thulasiraman, K.: Self-organization in cooperative tabu search algorithms. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Volume 3. (11–14 Oct. 1998) 2379–2384
2. Crainic, T.G., Gendreau, M.: Cooperative parallel tabu search for capacitated network design. Journal of Heuristics **8**(6) (2002) 601–627
3. Crainic, T.G., Gendreau, M., Hansen, P., Mladenović, N.: Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics **10**(3) (2004) 293–314

4. Pelta, D., Cruz, C., Sancho-Royo, A., Verdegay, J.: Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. Information Sciences **176** (2006) 1849–1868
5. Milano, M., Roli, A.: Magma: a multiagent architecture for metaheuristics. IEEE Transactions on Systems, Man, and Cybernetics, Part B **34**(2) (April 2004) 925–941
6. Amaya, J.E., Cotta, C., Fernández, A.J.: Hybrid cooperation models for the tool switching problem. In et al., D.P., ed.: International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Series on Studies in Computational Intelligence, Granada, Spain, Springer (2010) In Press.
7. Belady, L.: A study of replacement algorithms for virtual storage computers. IBM Systems Journal **5** (1966) 78–101
8. Bard, J.F.: A heuristic for minimizing the number of tool switches on a flexible machine. IIE Transactions **20**(4) (1988) 382–391
9. Tang, C., Denardo, E.: Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. Operations Research **36**(5) (1988) 767–777
10. Shirazi, R., Frizelle, G.: Minimizing the number of tool switches on a flexible machine: an empirical study. International Journal of Production Research **39**(15) (2001) 3547–3560
11. Amaya, J., Cotta, C., Fernández, A.: A memetic algorithm for the tool switching problem. In Blesa, M., et al., eds.: Hybrid Metaheuristics 2008. Volume 5296 of Lecture Notes in Computer Science., Málaga, Spain, Springer-Verlag (2008) 190–202
12. Oerlemans, A.: Production planning for flexible manufacturing systems. Ph.d. dissertation, University of Limburg, Maastricht, Limburg, Netherlands (October 1992)
13. Crama, Y., Kolen, A., Oerlemans, A., Spieksma, F.: Minimizing the number of tool switches on a flexible machine. International Journal of Flexible Manufacturing Systems **6** (1994) 33–54
14. Hertz, A., Laporte, G., Mittaz, M., Stecke, K.: Heuristics for minimizing tool switches when scheduling part types on a flexible machine. IIE Transactions **30** (1998) 689–694
15. Al-Fawzan, M., Al-Sultan, K.: A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. Computers & Industrial Engineering **44**(1) (2003) 35–47
16. Zhou, B.H., Xi, L.F., Cao, Y.S.: A beam-search-based algorithm for the tool switching problem on a flexible machine. The International Journal of Advanced Manufacturing Technology **25**(9-10) (May 2005) 876–882
17. Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Articial Intelligence Review **13** (1999) 129–170
18. Lehmann, E., D'Abrera, H.: Nonparametrics: statistical methods based on ranks. Prentice-Hall, Englewood Cliffs, NJ (1998)
19. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association **32**(200) (1937) 675–701
20. Iman, R., Davenport, J.: Approximations of the critical region of the Friedman statistic. Communications in Statistics **9** (1980) 571–595
21. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6** (1979) 6570