Cross entropy-based memetic algorithms: An application study over the tool switching problem

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

[1] Universidad Nacional Experimental del Táchira (UNET), Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela,

E-mail: jedgar@unet.edu.ve

[2] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, Málaga, 29071, Spain.

E-mail: {ccottap,afdez}@lcc.uma.es

Abstract

This paper presents a parameterized schema for building memetic algorithms based on cross-entropy (CE) methods. This novel schema is general in nature, and features multiple probability mass functions and Lamarckian learning. The applicability of the approach is assessed by considering the Tool Switching Problem, a complex combinatorial problem in the field of Flexible Manufacturing Systems. An exhaustive evaluation (including techniques ranging from local search and evolutionary algorithms to constructive methods) provides evidence of the effectiveness of CE-based memetic algorithms.

Keywords: Cross-entropy method, memetic algorithms, hybridization, tool switching problem

1. Introduction

Cross Entropy (CE) is a Monte Carlo approach to optimization proposed by Rubistein[1]. It can be described as a population-based iterative optimization technique based on the use of probability distributions by minimizing the cross-entropy distance (also called the Kullback-Leibler divergence) to a target distribution. It can thus be regarded as an Estimation of Distribution Algorithm (EDA)[2,3]. The CE method has been applied to tackle many optimization problems in telecommunications[4], engineering[5,6], and combinatorial optimization[7,8], just to cite a few examples.

Traditionally, EDAs are considered as evolutionary algorithms despite the fact they do not use the classical crossover and mutation operators. Instead the information is extracted from the candidate solutions via probability mass functions (pmfs). The particular form these probability distributions are modeled in actually leads to different EDA versions[9,10].

CE basically works as follows: starting from a initial pmf a population is built; then, in each iteration a sample from the population —the so-called elite sample— is selected to update the probability mass function. The so-updated pmf is then used to generate a new population. This process is iteratively repeated until eventually the distribution converges to a local optimum (ideally the global optimum). A complete mathematical description of the CE method is given by de Boer et al.[7]. Among other features, CE requires a large number of samples to achieve high quality solutions, thus leading to a potentially slow optimization process[7]. Some solutions have been proposed to cope with these problems such as, e.g., the use of several pmfs (instead of a simple one) along with their associated updating mechanisms as an alternative for exploring the search space more effectively[11]. Parallel versions of CE have been also proposed[12,13], and Laguna et al.[14] have recently described a hybrid CE version that follows the idea of memetic algorithms (MA)[15,16,17,18,19] by endowing CE with local search (LS).

Now this paper presents a parameterized schema upon which a number of many different CE versions can be devised. Some specific instantiations of this schema correspond to memetic models already proposed in the literature[14], classical CE algorithms and CE with multiple pmfs. In addition, novel algorithms (that, as far as we know, have not been proposed before) can be derived as direct instances of our schema as, for example CE-based memetic algorithms with multiple pmfs. In the simplest cases we can produce simple (i.e., non-memetic) CE algorithms using only one pmf function; however the schema also represents a model to hybridize EDAs (in a general sense, and CEs from a more particular perspective) with local search (LS) techniques. The hybridization is done in an integrative way[20] where the local searcher is employed as a component that can be activated during the execution of the primary optimization technique, in this case an EDA/CE. In this context, LS helps to intensify the search whereas the primary technique provides both a population-based search and the use of multiple pmfs that contribute to diversify the optimization in the search space of the problem under consideration. The hybrid algorithms devised from this model should be synergistic, providing better results than each of its constituent parts (i.e., underlying algorithms).

To show the adequacy of this schema as well as the effectiveness of some algorithms devised from it, we have considered the Tool Switching Problem (ToSP) as application domain. The ToSP has its origin in Flexible Manufacturing Systems (FMSs). To be precise, we consider the case of a reconfigurable production machine endowed with a *magazine* that has several slots into which different tools can be loaded. By an appropriate scheduling of tools to be loaded/unloaded to/from this magazine, the machine can handle a sequence of jobs, each of them with their own tool requirements. In this context, the ToSP amounts to finding an appropriate job sequence as well as the schedule of loading/unloading operations that minimizes the total number of tool switches within the *magazine*. Examples of the ToSP may be found in various areas such as electronics industry, metal industry, management of memory computers, aerospace and manufacturing in general[21,22]. The problem will be described in more detail in Section 4.

In the context of this problem, we describe here a number of specific instantiations of our generic schema to solve it; to the best of our knowledge all of them are novel algorithms for tackling the ToSP as no CE-based algorithm was previously used for it. In addition, some of these instantiations outperform a cooperative (i.e., agent-based) algorithm that represented so far the state of the art for the ToSP and thus they supersede the latter as the best-known approach to solve this problem.

The rest of the article is organized as follows. Section 2 provides a general overview of EDAs and CE, and Section 3 describes our parameterized generic schema for producing CE-based memetic algorithms. Section 4 is devoted to overview the ToSP and describing two standard EDAs for tackling the ToSP. Section 5 shows the results of a comprehensive experimental study that involves a wide set of algorithms to solve the ToSP. The paper ends on some conclusions and alternative lines of future research.

2. Background on EDAs and CE

EDAs were proposed by Mühlenbein et al.[23] and departed from traditional evolutionary algorithms in that the generation of new solutions depended on a probabilistic mechanism, rather than on the use of

Algorithm 1: Schema of a standard EDA

```
1  $M_0 \leftarrow$ GenerateInitialModel();
2  $P_0 \leftarrow$ GeneratePopulationFrom($N$,$M_0$); // $N$ solution candidates
3  $i \leftarrow 0$ ;
4  while $\neg$ Termination($i$,$P_i$) do
5      $i \leftarrow i+1$;
6      $Q_i \leftarrow$ Select($\kappa$,$P_{i-1}$); // Select $\kappa \leqslant N$ individuals from $P_{i-1}$
7      $M_i \leftarrow$ BuildModel($Q_i$); // Build probabilistic model $M_i$ from $Q_i$
8      $P_i \leftarrow$ GeneratePopulationFrom($N$,$M_i$);
9  end while
10 return Best solution in $P_i$
```

a set of genetic operators. Relationships and dependencies among the variables that define a solution to the problem under consideration are explicitly expressed in EDAs via probability distributions. The latter are built from a subset of individuals selected from a population generated from the model obtained in a previous iteration[24,25]. Thus, EDAs require neither crossover nor mutation operators.

In general, the most simple schema of an EDA is as described in Algorithm 1. An initial probabilistic model is built, typically describing a uniform distribution over the search space (although some heuristic initialization can also be done if problem-knowledge is available). This model is subsequently sampled to obtain a population of solutions from which an elite sample will be extracted and used to rebuild the model (lines 6-7). The new model is then resampled and the whole process is repeated until a certain termination condition (usually reaching a predetermined number of iterations, or finding a good enough solution) is fulfilled. How to address the functions Select/2, BuildModel/1 and GeneratePopulationFrom/1 is something for which numerous alternatives exist.

Larrañaga et al.[10] and Pelikan et al.[9] presented a classification of EDAs for combinatorial optimization based on the structure of the probabilistic model used to capture the interdependencies among variables from selected individuals. In this classification there are three major classes of EDAs:

- Univariate distributions: the probabilistic model comprises marginal probabilities for each variable, as in, e.g., UMDA (*Univariate Marginal Distribution Algorithm*)[23].
- Bivariate distributions: each variable depends at most on another variable. The probabilistic model is thus a tree or a collection of trees, whose nodes represent a variable and comprises conditional probabilities for its value (conditional to the value of the parent variable in the tree). Examples include MIMIC (*Mutual Information Maximization for Input Clustering*)[26] or COMIT (*Combining Optimizers with Mutual Information Trees*)[27].
- Multiple dependencies: each variable may depend on many other variables, so the probabilistic model is typically represented as a Bayesian network, as in BOA (*Bayesian Optimization Algorithm*)[28].

The CE algorithm considered in this work falls within the bivariate class and will be described in detail later on. Let us firstly focus on UMDA and PBIL (*Population Based Incremental Learning*), two algorithms that will be used later in the empirical comparison along with other methods defined in the literature.

As already mentioned, UMDA is an EDA based on univariate probabilistic models. It was proposed by Mühlenbein and Paaß[23] and models the joint distribution of variable values as the product of independent univariate marginal distributions. Let $\mathscr{X}$ be the solution space, and let us assume that solutions $x \in \mathscr{X}$ are represented by n variables $v_1, \cdots, v_n$, where v_i takes values from a domain $D_i =$

$\{d^i_1, \cdots, d^i_{m_i}\}$. Then, the probabilistic model at step t consists of an array of probability mass functions (pmfs) $\mathbf{f}^t = (f^t_1, \ldots, f^t_n)$, where $f^t_i(k)$ (for $1 \leqslant i \leqslant n$ and for $1 \leqslant k \leqslant m_i$) is the probability of v_i taking value d^i_k. At each step, the model is sampled to obtain a new population P, and a new model $\mathbf{f}^{t+1}$ is obtained from the best (according to a guiding function $\mu : \mathscr{X} \to \mathbb{R}$) κ individuals in this population by

$$f^{t+1}_i(k) = \frac{1}{\kappa} \sum_{z=1}^{\kappa} [Q_{zi} = d^i_k] \ , \qquad (1)$$

where Q is the set of κ elite samples, Q_{zi} is the value of the i-variable of the z-th solution in Q, $[\cdot]$ is an indicator function ($[\texttt{TRUE}] = 1$ and 0 otherwise).

PBIL was proposed by Baluja[29], and generalizes the functioning of UMDA by considering the following Hebbian rule for updating the probabilistic model:

$$f^{t+1}_i(k) = (1-\alpha) f^t_i(k) + \alpha \frac{1}{\kappa} \sum_{z=1}^{\kappa} [Q_{zi} = d^i_k] \ , \qquad (2)$$

where $\alpha \in (0,1]$ is the *learning* parameter. A proof of the algorithm convergence is shown by Hohfeld and Rudolph[30]. Note that PBIL is equivalent to UMDA when $\alpha = 1$.

The Cross Entropy method (CE)[1] is an adaptive technique initially used to estimate the probability of rare events based on the minimization of the Kullback-Leibler (KL) divergence. Roughly speaking, the KL divergence ($\mathscr{D}_{KL}$) is a measure of the "distance" between two pmfs a and b, i.e.,

$$\mathscr{D}_{KL}(a,b) = \sum_{X \in \mathscr{X}} a(X) ln \frac{a(X)}{b(X)} \qquad (3)$$

The idea in rare event simulation is finding the probability ε that $\mu(x) \geqslant \gamma$, where γ is a real number (a threshold) and $x \in \mathscr{X}$ is a solution sampled from $\mathscr{X}$ using a certain pmf $f(\cdot)$. To do so, a pmf g is sought such that it minimizes the KL divergence with an "ideal" pmf $g^*(x) = [\mu(x) \geqslant \gamma] f(x)/\varepsilon$ (i.e., $g^*(x) > 0$ iff $\mu(x) \geqslant \gamma$). Of course, this ideal pmf is unknown but a solution to the mentioned problem can be estimated (for not too small values of ε) as

$$\arg\max_g = \sum_{j=1}^{M} [f(x^j) \geqslant \gamma] \ln g(x^j) \qquad (4)$$

where the sample $x^1, x^2, ..., x^M$ is obtained from $\mathscr{X}$ using pmf $f(\cdot)$. Note that the estimation problem in Eq. (4) can be solved analytically under certain conditions, e.g., discrete solutions following a n-dimensional Bernoulli distribution[7,5] leading to a solution analogous to Eq. (1).

In order to deploy CE on combinatorial optimization problems, the method is transformed in an iterative approach in which different rare-event problems are solved at each stage. Such rare-events are defined as follows: at step i a pmf $f_i(\cdot)$ is considered; initially $f_1(\cdot)$ is a uniform distribution, and later $f_{i+i}(\cdot) = g_i(\cdot)$, where $g_i(\cdot)$ is the optimal solution to the i-th estimation problem. This problem is defined as the rare-event $\mu(x) \geqslant \gamma_i$, where x follows the pmf $f_i(\cdot)$ and γ_i is the k-th worst value of $\mu(x)$ in a sample of M solutions, i.e., γ_i if the cut-off fitness value in the elite sample. This procedure is repeated for a suitable number of iterations.

Regarding hybridization of EDAs in general (and CE in particular) with other optimization methods, we can cite for instance the work of Ortiz-García and Pérez-Bellido[31], where an integration mechanism between CE and a neural network was used to solve the frequency assignment problem. Campelo et al.[32] describe a hybrid approach of EDAs with approximations to the most promising solutions obtained via local search-based methods, and apply it to the design of electromagnetic devices. More recently, Laguna et al.[14] utilized local search mechanisms to optimize the whole elite sample, and test this approach on the maximum cut problem. This proposal was shown to be especially effective and can be considered as an instance of our algorithmic schema shown in the next section. Santana et al.[33] combine a variable neighborhood search (VNS) technique with UMDA with the aim of inferring the structure of a protein from its aminoacid sequence. They propose and analyze three alternatives for the combination: incorporating VNS within the UMDA, using probabilistic models within VNS, and alternating VNS and UMDA. Zhang et al.[34] proposed the hybridization of PBIL with a 2-opt local search algorithm. This proposal was applied successfully to the quadratic assignment problem. Peña et al.[35] described a hybrid scheme between a steady state genetic algorithm (GA) and a UMDA and where each algorithm is responsible, according to certain *participation function*, for the generation of part of the total population. Also, Zhou et al.[36] propose a model resulting from the hybridization of particle swarm optimization (PSO) and EDAs. Recently, Ahn et al.[37] have presented a hybridization between binary particle swarm optimization (BPSO) and a multivariate version of an EDA; this proposal is based on a process of selecting the best solutions found by the BPSO through the EDAs with the aim of promoting the exploitation of candidate solutions.

3. A general cross entropy-based memetic schema

The Cross Entropy (CE) method can be considered as an EDA in the field of combinatorial optimization problems and basically follows the schema shown in Algorithm 1. More specifically, CE consists of two main phases: the first one is devoted to obtaining samples (i.e., the elite samples) from a specific pmf and the second one to modify the parameters of the probability distribution from the actual elite sample, in order to produce better samples at the next iteration[7]. Following this general model, we propose here the schema shown in Algorithm 2 from which many different variants of CE can be devised. When referring to CE, just one pmf is traditionally considered; note that our schema also considers the use of multiple pmfs though, as well as the integration of a local search (LS) mechanism. Note however that this schema, in its current form, might also be used to devise other EDA variants as no specific detail for CE are explicitly incorporated to increase both its legibility and understanding. In any case, in the rest of the paper we center specifically on showing how this schema can be used to generate CE-based (possibly memetic) algorithms so we focus on specific features of CE.

Let $I_{\mathscr{P}}$ be a specific instance of a particular optimization problem $\mathscr{P}$ to be tackled. Let us assume, without lost of generality that $\mathscr{P}$ is a minimization problem. Several parameters are considered in this schema to solve $I_{\mathscr{P}}$:

- An ordered collection $\mathbb{P}$ (i.e., the population) of N candidate solutions.
- $\mathbb{F}_w = \{f^1, \ldots, f^w\}$, a set of w pmfs.
- ρ, the *cutoff point*, that will be used to select the size of the elite sample. The cutoff point ρ determines the percentage of the population that will be considered as the elite sample. For instance, suppose that $\rho = 0.05$ and $N = 100$ then the elite sample would contain $\rho N = 5$ solution candidates. As already mentioned, the elite sample is used to update the probability function(s).
- θ, the probability of local search (LS).

Firstly, all the pmfs are defined (i.e., initialized in line 2 as uniform probability mass functions). Then, iteratively the algorithm executes the following steps in sequence: it creates a pool $\mathbb{P}$ of N candidate solutions (line 6) via the pmfs in $\mathbb{F}_w$, evaluates each of the components in $\mathbb{P}$ via an $I_{\mathscr{P}}$-specific fitness function (line 7), and sorts the pool in ascending (descending, in the case of a maximization problem) order according to fitness values (line 10); then, the algorithm incorporates an improvement phase in which the best ρN elements of this ordered population $\mathbb{P}$ (i.e., the elite sample) are locally improved (line 13); this step is executed with a certain probability θ; observe that while $\theta = 0.0$ in simple versions of CE, strictly positive values of this parameter lead to memetic versions of the algorithm. Finally, the pmfs are updated from information extracted of these ρN (possibly improved) elite elements (line 15). To do so, this elite subset is sorted by decreasing fitness values and divided in w interleaved tiers; then, each pmf is updated from one of these tiers, i.e., f^i uses elite solutions indexed as $i + jw$, $j \geqslant 0$. This cycle is repeated until a certain termination condition is held (e.g., reaching a maximum number of evaluations *maxEvals*). The algorithm returns the best solution that can be generated from $\mathbb{F}_w$ (line 17).

Observe that if $w = 1$ and $\theta = 0.0$ we have the canonical version of CE. Note also that the procedures INITIALIZEFUNCT/1, GENERATEPOPULATION/1, UPDATEPMFS/2, and BESTSOLUTION/1,

Algorithm 2: Schema of memetic CE/EDA with w pmfs for ToSP

```
1  for $k \in \mathbb{N}_w$ do // $\mathbb{N}_w = \{1, \cdots, w\}$
2      InitializeFunct($f^k$);
3  end for
4  while Not Termination Condition do
5      // Generate a new population of size $N$ from $w$ pmfs.
6      $\mathbb{P} \leftarrow$ GeneratePopulation($\mathbb{F}_w$); // return $\mathbb{P} = \{P^1, \ldots, P^N\}$
7      EvaluatePopulation($\mathbb{P}$);
8      // sort population in ascending order according to fitness, to subsequently
9      // select the elite sample.
10     RankPopulation($\mathbb{P}$);
11     // process of EliteSampleImprovement:  LS stochastically applied to the
12     // best $\rho N$ solutions with probability $\theta$ in each case.
13     LocalImprovement($\mathbb{P}, \rho, \theta$);
14     // update $f^1, \ldots, f^w$ from an elite sample of size $\rho N$ extracted from $\mathbb{P}$.
15     UpdatePMFs($\mathbb{F}_w, \mathbb{P}$);
16 end while
17 return BestSolution($\mathbb{F}_w$);
```

must be adapted specifically to the problem $\mathscr{P}$ (although these can be also devised from standard proposals for classical CE as shown in our specific application).

As stated before, assigning $\theta > 0.0$ means that our schema can generate a number of memetic algorithms by endowing CE with a local search operator. Thus, the hybrid algorithm described by Laguna et al.[14] can be viewed as an instance of our schema, with $w = 1$ and $\theta = 1.0$.[*] This setting means performing local search on each generated candidate, but this is not necessarily the best choice in standard memetic algorithms[38]. Indeed partial Lamarckianism[39], namely applying local search only to a fraction of individuals ($0 < \theta < 1$), can result in a better performance. Also, the individuals to which local search will be applied can be selected in many different ways[40]. We have considered a simple approach in which local search is applied, individually, to the best ρN candidates in the population (i.e., an elite sample $\subseteq \mathbb{P}$) with a probability θ; in case of application, the improvement uses up to a number of *LSevals* evaluations or, in the case of a specific local search such as Hill climbing (HC) until it stagnates, whichever comes first.

The idea behind CE is to find the pmf to generate the best possible solutions. However, one of the ways to include a broader exploration of search space is to include multiple pmfs in the CE method (i.e., assigning $w > 1$). The selection of samples for upgrading pmfs and subsequently generating populations leads to different versions of the algorithm, for example, the well-known Model Reference Adaptive Search[11] that uses multiple pmfs but establishes specific mechanisms for updating the values of the pmfs.

The underlying idea of our hybrid CE+LS is to combine the intensifying capabilities of the embedded LS method, with the diversifying features of both a population-based search and the use of multiple pmfs i.e., the population, generated from different pmfs, 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. Moreover, an improved population can devise an im-

[*]Laguna et al.[14] introduce a parameter δ to control which fraction of the population undergoes local search. The equivalence mentioned only holds when $\delta = 1$.

proved set of pmfs, and ideally, this combination should be synergistic, providing better results that either the CE or the LS techniques by themselves. Empirical evidence of this fact will be provided in the following sections.

4. A case study: the tool switching problem

To assess the schema presented in the previous section, we have chosen the tool switching problem as a benchmark. The ToSP is a combinatorial optimization problem that involves scheduling a number of jobs on a single reconfigurable machine so that the resulting number of tool switches required is kept to a minimum. This section is thus devoted to an overview the problem, and describes the deployment of the optimization method on it.

4.1. Formulation of the uniform tool switching problem

In light of the informal description of the uniform ToSP given in Section 1, there are two major elements in the problem: a machine M and a collection of jobs $JOBS = \{Job_1, \cdots, Job_n\}$ 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 Job_i requires a certain subset $T^{(Job_i)} \subseteq T$ 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., $A_{ij} = [\tau_i \in T^{(Job_j)}]$, thus being 1 if —and only if— tool i (i.e., τ_i) is required to execute job j (i.e. Job_j), being 0 otherwise.

We assume that $C < m$; otherwise the problem is trivial. The solution to such an instance is a sequence $\langle J_1, \cdots, J_n \rangle$ that consists of a permutation of $JOBS$ (i.e., $J_i \in \{Job_1, \cdots, Job_n\}$ for $1 \leqslant i \leqslant n$, and $J_i \neq J_j$ for all $i, j \in \{1, \ldots, n\}$ and $i \neq j$) 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 $T^{(J_j)} \subseteq T_j$.

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 integer linear programming (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. (6) and (7),
- $y_{ik} = 1$ if tool $i \in \mathbb{N}_m$ is in the magazine at instant $k \in \mathbb{N}_n$, and 0 otherwise —see Eq. (8).

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. (9). Following the work by Bard[21], 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. (5). Without loss of generality we assume that that the cost of each tool switching is unitary.

$$\min\ F(y) = \sum_{j=1}^{n} \sum_{i=1}^{m} y_{ij}(1 - y_{i,j-1}) \tag{5}$$

$$\forall j \in \mathbb{N}_n : \sum_{k=1}^{n} x_{jk} = 1 \tag{6}$$

$$\forall k \in \mathbb{N}_n : \sum_{j=1}^{n} x_{jk} = 1 \tag{7}$$

$$\forall k \in \mathbb{N}_n : \sum_{i=1}^{m} y_{ik} \leqslant C \tag{8}$$

$$\forall j, k \in \mathbb{N}_n\ \forall i \in \mathbb{N}_m : A_{ij} x_{jk} \leqslant y_{ik} \tag{9}$$

$$\forall j,k \in \mathbb{N}_n\ \forall i \in \mathbb{N}_m:\ x_{jk}, y_{ij} \in \{0,1\} \qquad (10)$$

From a conceptual point of view, the ToSP can be divided into three subproblems[41]: 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. The latter can however 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 —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, 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)[42,21,22,43]. 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 inserted and there are no vacant slots on the magazine, keep the tools that are needed soonest. Let $J = \langle J_1, \cdots, J_n\rangle$ be the job sequence, and let $T_k \subset \mathbb{N}_m$ be the tool configuration at time k (for $1 \leqslant k \leqslant n$). Let $\Xi_{jk}(J)$ be defined as

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

  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 away 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 loading subproblem, and use KTNS as a subordinate procedure to solve the subsequent tool loading subproblem.

In the rest of the paper, $KTNS_A(J)$ will denote the number of tool switches obtained from applying the KTNS method to the job sequence J and considering A as the tool requirement matrix associated to the problem instance, and write $KTNS(J)$ whenever A is implicit in the context.

The ToSP is NP-hard for $C > 2$[44,45] and hence exact methods are inherently limited. We refer to previous works[46,47] for a brief survey of related work on exact methods and heuristics for the ToSP. Among these techniques, it is worth mentioning beam search, a derivate of branch-and-bound that uses a breadth-first traversal of the search tree, heuristically keeping at each level just the best β nodes. This method was shown to be effective for the ToSP[48] and will be included in our experimental comparison. In addition, we will also consider metaheuristics such as the tabu search (TS) approach presented by Al-Fawzan and Al-Sultan[49], as well as the hill climbers and memetic algorithms (MA) presented by Amaya et al.[46]. It must be noted that the latter MA is the current incumbent for solving the ToSP.

4.2. Solving the ToSP with EDAs

Let $I = \langle C, A\rangle$ be a specific instance of the problem with m tools and n jobs. Then, following the schema shown in Algorithm 2 we consider,

- $\mathbb{P} = \{J^1, \cdots, J^N\}$ where each $J^h = \langle J^h_1, \cdots, J^h_n\rangle$ (for $h = 1, \cdots, N$) is a job sequence, that is to say, J^h is a permutation of the elements in $JOBS$ (i.e., $\{Job_1, \ldots, Job_n\}$) determining the order in which the jobs are executed.
- For any $k \in \{1, \ldots, w\}$, $f^k \in \mathbb{F}_w$, and $J_i, J_j \in JOBS$, $f^k_{J_i,J_j}$ represents the probability of executing job J_j just after having executed job J_i. Initially (line 2 of Algorithm 2) all pmfs are uniform, i.e., the probability of executing any job after a specific

Algorithm 3: Method GENERATEPOPULATION($\{f^1,\ldots,f^w\}$). Generate a population of N job sequences from a set of w pmfs

```
1  $\mathbb{P} \leftarrow \emptyset$;
2  for $i \in \mathbb{N}_w$ do
3      $j \leftarrow 1$;
4      while $j \leqslant N/w$ do
5          $J \leftarrow$ GENERATEJOBSEQUENCE($f^i$); // Construct sequence from $f^i$
6          if $J \notin \mathbb{P}$ then
7              $\mathbb{P} \leftarrow \mathbb{P} \cup \{J\}$;
8              $j \leftarrow j+1$;
9          end if
10     end while
11 end for
12 return $\mathbb{P}$
```

job is the same, that is to say, for any $f^k \in \mathbb{F}_w$, $f^k_{J_i,J_j} = \frac{1}{n-1}$ if $J_i \neq J_j$ and 0 otherwise.

- The fitness function FITNESS(P^i) is defined as $KTNS_A(J^i)$

Also, for the LS/1 method shown in line 8 of Algorithm 2 we have used Hill Climbing and Tabu Search. The former is a steepest ascent method working on the swap-neighborhood of permutations[46]. Note that the exploration of the whole neighborhood becomes more and more costly as the number of jobs increases (its size is $\Theta(n^2)$ where n is the number of jobs). In a fixed computational budget scenario, this implies the allocated computational effort can be quickly consumed. For this reason, we have opted for also taking into account two variants of neighborhood exploration: one in which the swap neighborhood is fully explored and another one in which a partial exploration is done by obtaining a fixed-size random sample (to be precise, the size of this sample has been chosen to be λn, i.e., proportional to the number of jobs). As for Tabu Search, it is based on the proposal described by Al-Fawzan and Al-Sultan[49], consisting in performing strategic oscillation between the said swap neighborhood and the block neighborhood (swapping larger blocks of elements —see also the work by Amaya et al.[47]). As indicated, we have considered a simple approach in which local search is applied to the best ρN candidates in the population (i.e., an elite sample $\{J^1,\ldots,J^{\rho N}\} \subseteq \mathbb{P}$) with a probability θ; in case of application, the improvement uses up to a number of *LSevals* evaluations (or in the case of HC until it stagnates, whichever comes first).

Algorithm 3 shows the pseudocode for the procedure GENERATEPOPULATION/1. Our proposal is a generational algorithm in which the population $\mathbb{P}$ is renewed in each iteration. The idea is that each pmf is used to create N/w job sequences; this means that each pmf has the same weight on the overall population. The procedure for generating a job sequence from a pmf is shown in Algorithm 4 and is based on a method proposed by Rubinstein[1] to solve the *traveling salesman problem*. The idea is to construct a sequence by selecting a new job to be added to the sequence considering the probabilities of moving from the job previously placed to this new job. In fact, this method bears some similarities with roulette-wheel selection in genetic algorithms. The first job in the sequence is randomly selected (lines 2 and 3); let $J_{current}$ (for $current \in \{1,\ldots,n-1\}$) be the last job placed in the (partial) sequence, then the new job to place is selected from the set of jobs not yet placed in the sequence, assigning to each of them a portion of the wheel proportional to the probability of moving from this last job $J_{current}$ to any other job.

Algorithm 4: Method GENERATEJOBSEQUENCE(f)

```
1  $processed \leftarrow \emptyset$; // Jobs already processed
2  $current \leftarrow random(n)$; // Current job randomly drawn from [1,n]
3  $J_1 \leftarrow Job_{current}$; // first element in job sequence
4  i $\leftarrow 2$;
5  while $i \leqslant n$ do
6      $processed \leftarrow processed \cup \{current\}$;
7      $\mu \leftarrow URand01() * \left(\sum_{another \notin processed} f_{Job_{current}, Job_{another}}\right)$;
8      // For $another \in \mathbb{N}_n$
9      $next \leftarrow 0$; $sum \leftarrow 0$;
10     while $sum < \mu$ do
11         $next \leftarrow next + 1$;
12         if $next \notin processed$ then
13             sum $\leftarrow$ sum + $f_{Job_{current}, Job_{next}}$;
14         end if
15     end while
16     $J_i \leftarrow Job_{next}$; $current \leftarrow next$; $i \leftarrow i + 1$;
17 end while
18 return $\langle J_1, \cdots, J_n \rangle$
```

Algorithm 5: Method UPDATEPMFS($\{f^1, \ldots, f^w\}, \{J^1, \ldots, J^N\}$).

```
1  for $k \in \mathbb{N}_w$ do
2      $f^k \leftarrow$ allzeros$(n,n)$; // $f^k_{Job_i, Job_j} = 0.0$ for any $Job_i$ and $Job_j$
3      $h \leftarrow k$; $g \leftarrow 1$;
4      while $(g \leqslant \rho N)$ and $(h \leqslant N)$ do
5          for $i \in \mathbb{N}_{n-1}$ do
6              $f^k_{J^h_i, J^h_{i+1}} \leftarrow f^k_{J^h_i, J^h_{i+1}} + \frac{1}{\rho N}$;
7              $f^k_{J^h_n, J^h_i} \leftarrow f^k_{J^h_n, J^h_i} + \frac{1}{\rho N(n-1)}$;
8          end for
9          $h \leftarrow h + w$; $g \leftarrow g + 1$;
10     end while
11 end for
```

An schema for updating the pmfs is shown in Algorithm 5. Here, the elite sample $\{J^1, \ldots, J^{\rho N}\} \subseteq \mathbb{P}$ (i.e., the best ρN candidates) is used to update each function $f^k \in \mathbb{F}$ as follows:

$$\forall h \in \mathbb{N}_{\rho N}, \forall i \in \mathbb{N}_{n-1} \; : \; f^k_{J^h_i, J^h_{i+1}} = f^k_{J^h_i, J^h_{i+1}} + \frac{1}{\rho N} \quad (11)$$

The probability of moving from a job to the next one in the sequence is augmented by a certain factor depending on the size of the elite sample. There is a special case for the last job in the sequence that consists of taking into account the possibility of moving from this last job to any of the other possible jobs (i.e., note that there are $n-1$ possibilities and thus the probability is increased by a factor $\frac{1}{\rho N(n-1)}$).

Finally Algorithm 6 shows our proposal for the algorithm BESTSOLUTION/1 adjusted to the ToSP optimization. The aim is to produce the best solution from the (best produced) set of pmfs. Observe that for each pmf f^k in $\mathbb{F}_w$, this method creates n jobs sequences, each one starting with a different initial job. Then, the rest of the sequence is constructed greedily as follows (cf. Algorithm 7): assume than J_t is the last job added to the sequence, the job $Job_v \in JOBS$ whose probability, according to f^k, of moving from J_t to Job_v is highest is chosen for being added next (where Job_v is a job that was not added previously to the sequence). Ties are broken randomly in case there are several jobs that maximize this probability value. Note that although this method BESTSOLUTION/1 is specific to the ToSP it is easy to adapt it to other optimization problems by basically redefining the greedy construction. This redefinition can also lead to different versions of the method for the same problem.

We have also implemented the algorithms PBIL and UMDA as described in Section 2 for tackling ToSP. Both of these algorithms correspond to instances of our schema shown in Algorithm 2 where $\mathbb{P}$, FITNESS/1 and methods INITIALIZEFUNCT/1, GENERATEPOPULATION/1 and BESTSOLUTION/1 are as in CE versions described above, and also $w = 1$ and $\theta = 0.0$, i.e., they are standard EDAs considering only one pmf and with no local improvement phase. However, the method UPDATEPMFS is different to that of our CE-based versions since the pmf just captures the probability of a certain task occurring at a certain position (of course, sampling is later done avoiding already considered tasks, and renormalizing probabilities). In the following these algorithms are denoted as UMDA(ρ) and PBIL (ρ, α) according to different values of ρ and α (note that in UMDA $\alpha = 1.0$).

5. Computational results

This section describes an empirical analysis that has been executed to handle the ToSP via a number of distinct algorithms. Some of these algorithms are instances of our schema shown in Algorithm 2 as explained in Section 4.2.

5.1. General issues

As far as we know, no standard benchmark exists for this problem (at least no publicly available). For this reason, we have selected a wide set of problem instances that have been considered in the literature[21,50,49,48]; to be precise, 16 instances have been selected, with number of jobs $n \in [10, 50]$, number of tools $m \in [9, 60]$, and machine capacity $C \in [4, 30]$. 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 $C\zeta^m_n$.

Five different datasets[†](i.e., tool incidence matrices) were generated randomly per instance. Each dataset was generated with the constraint, already imposed in previous works, e.g., by Hertz et al.[50], that no job is *covered* by any other job in the sense that for no two different jobs i and j, $T^{(Job_i)} \subseteq T^{(Job_j)}$. 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[21] and Zhou et al.[48].

All the algorithms/variants that have been considered in the performance analysis (see Section 5.2) have been executed 10 times by *dataset* for a total

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

Algorithm 6: Method BESTSOLUTION($\{f^1, \ldots, f^w\}$).

```
1  $max \leftarrow \infty$;
2  for $k \in \mathbb{N}_w$ do
3      // Each function generates $n$ solutions
4      for $i \in \mathbb{N}_n$ do
5          // Construct a job sequence starting with job $Job_i$ from function $f^k$
6          $J \leftarrow$ GREEDYSEQUENCECONSTRUCTION($f^k, i$);
7          if $KTNS(J) < max$ then
8              $J^* \leftarrow J$;
9              $max \leftarrow KTNS(J)$;
10         end if
11     end for
12 end for
13 return $J^*$
```

Algorithm 7: Method GREEDYSEQUENCECONSTRUCTION(f, i)).

```
1  $J_1 \leftarrow Job_i$; // Sequence starts with job $Job_i$
2  $t \leftarrow 1$;
3  while $t < n$ do
4      $next \leftarrow max\{v \in \mathbb{N}_n \mid f_{J_t,Job_v}$ is maximal and $Job_v \notin \{J_1, \ldots, J_t\}\}$;
5      // Ties for different values of $v$ are randomly broken
6      $J_{t+1} \leftarrow Job_{next}$;
7      $t \leftarrow t + 1$;
8  end while
9  return $\langle J_1, \ldots, J_n \rangle$
```

of 50 executions for instance. The number of evaluations was given by $\varphi n(m - C)$ with $\varphi = 100$ as described by Amaya et al. [46].

5.2. *Algorithms and parametrization*

The experiments were carried out using a wide set of different algorithms; among these we include a Beam Search (BS) as presented by Zhou et al.[48], two LS methods: Hill Climbing (HC)[46] and Tabu Search (TS)[47], a genetic algorithm (GA) and a memetic algorithm (MAHC, a GA hybridized with Hill Climbing) as described in [46]; note that we do not include here the cooperative models proposed by Amaya et al.[47] as in that approach it was shown that the MAHC mentioned outperformed these collaborative approaches; in addition we also include a cooperative (agent-based) algorithm, termed as 4Ri(MAHCP,MATSP,MAHCP), that can be considered the state-of-the-art for solving the ToSP as shown in [51]; this algorithm consists of a ring topology with three agents loaded with a memetic algorithm that cooperate during the search process to handle the ToSP. Also, as novel applications to solve the ToSP we include the following algorithms: PBIL and UMDA as described in Section 4.2, and a number of different CE-based algorithms that stem from our generic schema described in Section 3. In particular we considered a simple cross entropy algorithm (CE, with $w = 1$ and $\theta = 0.0$), two memetic versions (CELS(θ), where the local search LS $\in$ {TS, HC} as defined previously, $w = 1$, $\theta \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$,

Table 1: 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 (see bibliography section). Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

	$4\zeta_{10}^{9}$	$4\zeta_{10}^{10}$	$6\zeta_{10}^{15}$	$6\zeta_{15}^{12}$	$6\zeta_{15}^{20}$	$8\zeta_{20}^{15}$	$8\zeta_{20}^{16}$	$10\zeta_{20}^{20}$
Min.	2	2	3	3	3	3	3	4
Max.	4	4	6	6	6	8	8	10
Source	21,48	49,50	48	21,48	50	49	21,48	21,48
	$10\zeta_{30}^{25}$	$15\zeta_{30}^{40}$	$15\zeta_{40}^{30}$	$20\zeta_{40}^{60}$	$24\zeta_{20}^{30}$	$24\zeta_{20}^{36}$	$25\zeta_{50}^{40}$	$30\zeta_{20}^{40}$
Min.	4	6	6	7	9	9	9	11
Max.	10	15	15	20	24	24	20	30
Source	49	50	49	50	21,48	21,48	49	48

and $LSevals = 100$), and the corresponding versions of both CE and CELS(θ) with $w > 1$ pmfs, namely CEM and CEMLS(θ). In the experiments, we set the value $w = 4$ as preliminary experiments suggested that this value is reasonably good for obtaining a correct performance.

As for local search methods, the notation HCP and HCF (resp. TSP and TSF) is used to indicate the HC variant (resp. TS variant) in which the swap neighborhood is partially or fully explored respectively (in the case of partial exploration we consider $\lambda = 4$, i.e., $4n$ neighbors explored [46]). Other details of each particular LS 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. 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 Al-Fawzan and Al-Sultan[49].

In [7], de Boer et al. suggest that the size of the samples should be a function of n^2 (where n is the number of variables), and thus for the CE method and its different variants (i.e., CE, CEM and their corresponding hybridized versions with LS methods), and also for UMDA and PBIL we set N to be n^2 (where n is the number of jobs) and $\rho = 0.01$. In the case of UMDA we also consider $\rho = 0.2$ (experiments with this value of ρ were also done with PBIL, but performance was very poor so it is not included in the tables for the sake of simplicity).

In summary, 19 types of different algorithms (i.e., HCP, HCF, TSP, TSF, BS, MAHCP, 4Ri(MAHCP,MATSP,MAHCP), UMDA, PBIL, CE, CEM, CEHCP, CEHCF, CETSP, CETSF, CEMHCP, CEMHCF, CEMTSP, and CEMTSF) and 103 variants of these for different values of its parameters have been considered in the experiments.

5.3. *Performance results and statistical comparisons*

The performance results obtained by all algorithms (and their different versions up to a number of 103 algorithms/variants) mentioned previously are shown in Table 6 (LS methods, Beam Search, MAHCP, 4Ri(MAHCP,MATSP,MAHCP), and non-memetic CE versions), Tables 7 and 8 (CELS variants for different values of θ), Table 9 (UMDA and PBIL for different values of α), and Tables 10 and 11 (CEMLS versions for different values of θ).

In all these tables $\bar{x}$ indicates the average number of the tool switchings obtained considering all the executions (i.e., the 50 executions per instance) and $\overline{\sigma}$ shows the mean standard deviations across the 5 datasets in the instance under consideration. Best results in terms of average are marked in bold.

As seen in Table 6, non-hybrid methods such as HC and TS are outperformed by hybrid variants ($4Ri(MAHCP, MATSP, MAHCP)$ and $MAHCP$ and by CE and CEM in 15 out of 16 instances (TS provides the best results in $6\zeta_{10}^{15}$). Additionally, we can

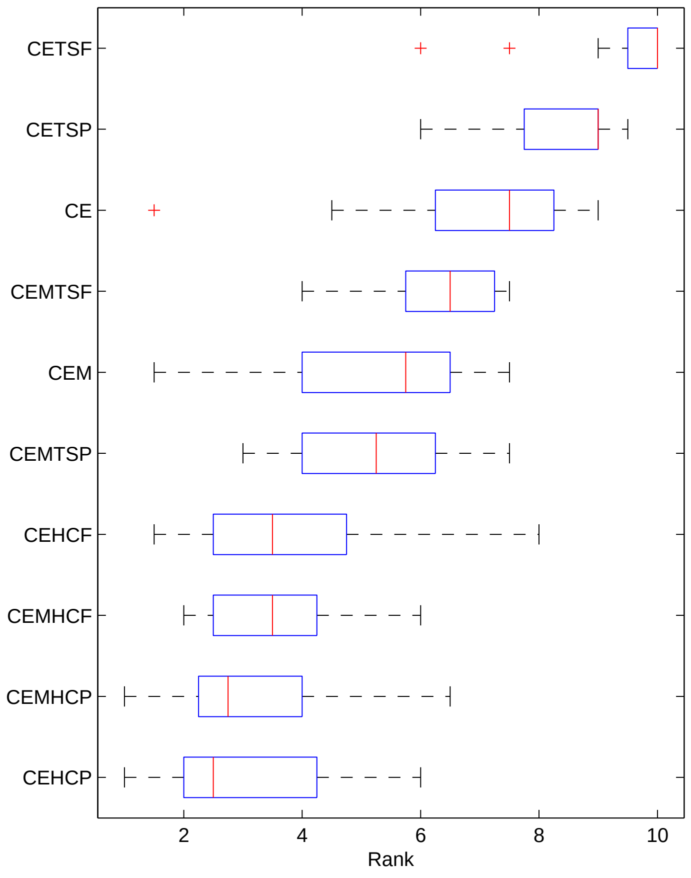

Figure 1: Rank distribution of CE-based algorithms considering all instances. As usual, each box comprises from the first to the third quartile of the distribution, the median (2nd quartile) is marked with a vertical line, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign.

Table 2: Results of Friedman and Iman-Davenport tests for the techniques in Figure 1.

	Friedman value	X_F^2 value	Iman-Davenport value	F_F value
CE-based algorithms	92.40	16.92	26.86	1.95

observe that CE and CEM perform comparably to the best algorithm reported for the resolution of the ToSP, i.e. $4Ri(MAHCP, MATSP, MAHCP)$; it is particularly interesting to note that CEM outperforms $4Ri(MAHCP, MATSP, MAHCP)$ in 12 out of 16 instances. This said, due to the high number of algorithm variants it is not easy to anatomize the performance of each of them compared with the rest by simply inspecting the numerical tables, so we have opted for a rank-based approach. To be precise, we have firstly focused on the performance of the family of CE-based algorithms (i.e., 10 different algorithms) and secondly on a global analysis comparing the best versions of each family of algo-

Table 3: Results of Holm's test using the CEHCP family as control algorithm ($\alpha = 0.05$).

i	algorithm	z-statistic	p-value	α/i
1	CEMHCP	0.1459	0.4419	0.05
2	CEMHCF	0.4379	0.3307	0.025
3	CEHCF	0.8466	0.1986	0.0167
4	CEM	1.9559	0.0252	0.0125
5	CEMTSP	2.0727	0.0191	0.01
6	CEMTSF	3.0653	0.0011	0.0083
7	CE	3.7659	< 0.0001	0.0071
8	CETSP	5.0213	< 0.0001	0.0063
9	CETSF	6.0431	< 0.0001	0.0056

rithms. In both cases, we have computed the rank r_j^i of each algorithm j on each instance i. In the first case we have used for ranking purposes the median value provided by each family on each dataset (i.e., the median of the mean values provided by each algorithm in the family), to account for the fact that some families have more members than others. In the second case the comparison involves individual algorithms run for the same number of times (10) on each dataset (for each of the 5 datasets in an instance). Hence we consider the mean of the solutions found on each dataset. The best algorithm receives rank 1 and the worst one receives rank k, where k is the number of algorithms involved in the ranking.

Specifically for CE-based algorithms, we have $k = 10$ different algorithms/versions namely, CE, CEM, CETSF, CETSP, CEHCF, CEHCP, CEMTSF, CEMTSP, CEMHCF, and CEHCP. Figure 1 shows the rank distribution for the CE-based algorithms; note that for the memetic versions we have used the version that provides the least average error considering all the instances according to Tables 7, 8, 10 and 11 that show the results obtained by the memetic versions for different values of parameter θ.

Next, we have used two well-known non-parametric statistical tests[52] to compare ranks, namely Friedman test[53] and Iman-Davenport test[54]. The results, at the standard level of $\alpha = 0.05$, are shown in Table 2. 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 (i.e., the statistical test is passed, thus indicating significant differences in their ranks at the standard $\alpha = 0.05$ level).

Subsequently, we have performed Holm's test[55] in order to determine whether there exist significant differences among these variants of CE with respect to a control algorithm (in this case the best version of the algorithm CEHCP, which presented the best average rank as shown in Figure 1). To do so, we compute the following z-statistic for the i-th algorithm as $z = (R_i - R_0)/\sqrt{\frac{k(k+1)}{6T}}$, where R_i is the average of the i-th algorithm, R_0 is the average of the control algorithm, $k = 10$ is the number of algorithms and T is the number of instances. Table 3 shows the results of this test. Note that the test is only passed for 4 of the algorithms in relation to the mentioned control algorithm and that no significant difference exists among the algorithms CEMHCP, CEMHCF, CEHCF, CEM, CEMTSP, and the control algorithm CEHCP. As a preliminary conclusion, it is clear that the CE-based memetic algorithms endowed with HC show the best overall performance and that CE-based algorithms with multiple pmfs represent a valuable alternative to tackle the solution of the ToSP.

Subsequently we have conducted the same analysis but considering the whole set of algorithm families (i.e., all those mentioned in Section 5.2) and Figure 2 shows the rank distribution where $k = 40$ is the number of algorithms involved in the comparison. For each particular algorithm we have considered its best versions (i.e., the three best ones that minimizes the average error considering all the instances according to the results shown in Tables 6-11).

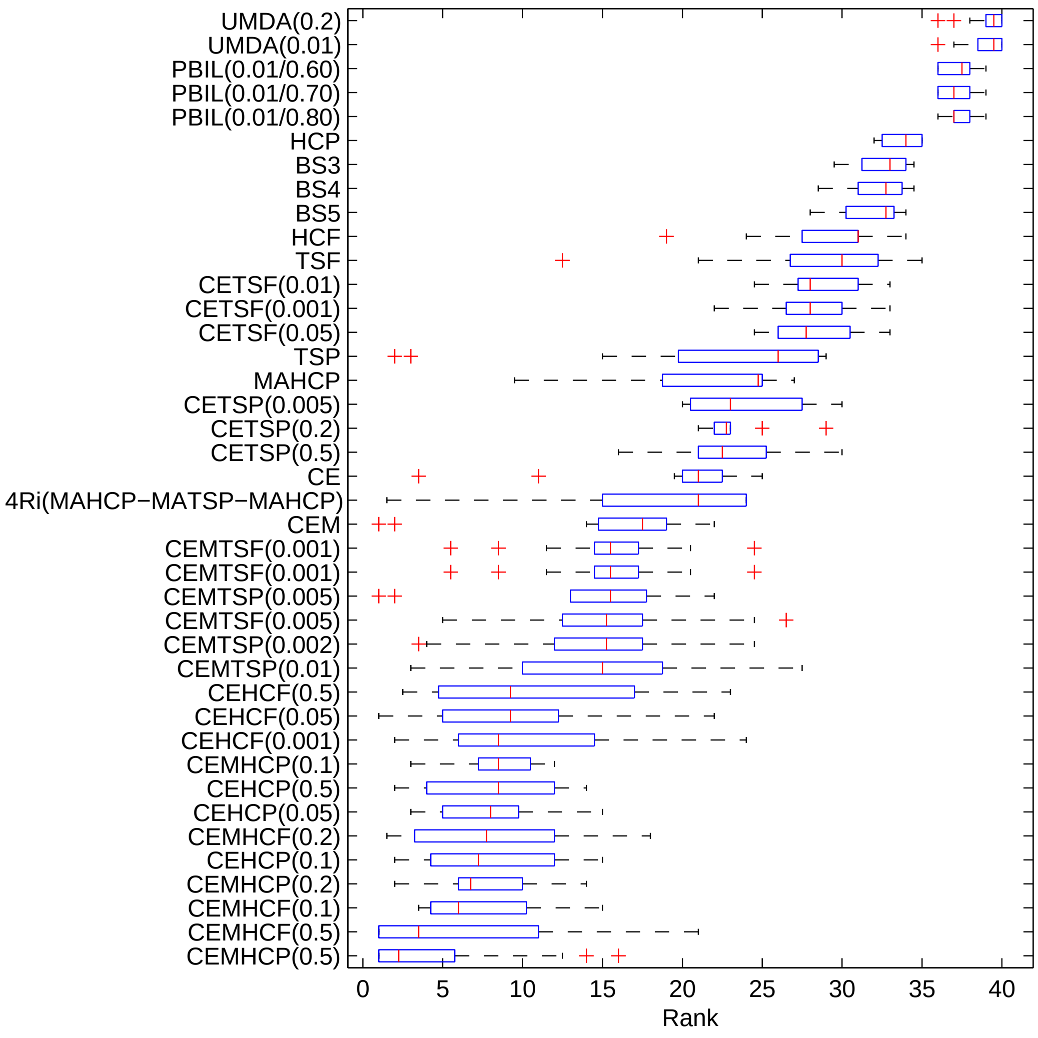

Figure 2: Rank distribution of the best variants of each family of algorithms across all instances.

Table 4: Results of Friedman and Iman-Davenport tests for the techniques in Figure 2.

	Friedman value	X_F^2 value	Iman-Davenport value	F_F value
All 40	550.34	54.57	112.07	1.42
Top 12	28.45	19.68	2.89	1.85

In general, CE and CEM show better performance than BS, HC and TS, although TS performs better than CE in some of the instances with low

Table 5: Results of Holm's test using CEMHCP(0.5) as control algorithm ($\alpha = 0.05$).

i	algorithm	z-statistic	p-value	α/i
1	CEMHCF(0.5)	1.1521	0.1246	0.05
2	CEMHCP(0.2)	2.2798	0.0113	0.025
3	CEMHCF(0.1)	2.3043	0.0106	0.0167
4	CEMHCF(0.2)	2.3288	0.0099	0.0125
5	CEHCP(0.05)	2.3533	0.0093	0.01
6	CEHCP(0.1)	2.5740	0.0050	0.0083
7	CEHCP(0.5)	2.6720	0.0037	0.0071
8	CEHCF(0.05)	3.1378	< 0.001	0.0063
9	CEMHCP(0.1)	3.2604	< 0.001	0.0056
10	CEHCF(0.5)	3.4810	< 0.001	0.0050
11	CEHCF(0.001)	3.5791	< 0.001	0.0045

number of jobs —see Table 6. Memetic versions of CE (i.e., CELS with TS and HC) provide disparate results: while the combination of CE with TS results in worse results than CE alone (yet slightly better than TS alone), the combination with HC is synergetic, providing better results than CE and HC in isolation. This can be interpreted in light of the better tradeoff provided by HC between the computational cost of the local searcher and its capability for intensifying the search. As to multi-pmf versions of CE, they generally perform better than their single-pmf counterparts, providing support to the usefulness of diversifying the search by means of multiple pmfs. The memetic version endowed with HC (in both variants HCP and HCF) provide the overall best results, and can be seen as a way of integrating the best of both worlds (diversification by multiple pmfs, and intensification by HC).

In general, it can be observed that the CE-based memetic algorithms in general perform better than the rest of algorithms. Notice that the best memetic CE-variants incorporating HC tend to use high values of θ, i.e., more frequent application of local search, whereas those incorporating TS typically use lower values. Again, this highlights the different search profile of the local searchers. While TS is good as a stand-alone technique, when combined with CE in a memetic framework the synergy attained is limited. HC on the other hand provides in this case a more cost-effective contribution to the hybrid global/local search.

From a statistical point of view, we can see that there are significant differences among both the whole set of techniques and the top 12 (whose relative performance is again better that the rest of techniques), as shown in Table 4. Furthermore, Holm's test indicates (see Table 5) that taking CEMHCP(0.5) as the control algorithm, differences are statistically significant ($\alpha = 0.05$) with the remaining techniques in the top 12 tier, except for CEMHCF(0.5).

6. Conclusions and future work

We have proposed a generic schema from which a number of Cross-Entropy (CE) based algorithms can be devised, including memetic versions hybridized with local search operators. Among the instantiations of the schema, there are algorithms that —to the best of our knowledge— are novel, in particular the CE-based memetic algorithms endowed with local search and multiple pmfs. In order to show the applicability of the schema, and also with the aim of checking the goodness of the novel hybrid algorithms proposed, we have conducted an exhaustive experimental analysis involving 19 different algorithms, and 103 versions of these, on the tool switching problem (ToSP), a complex combinatorial optimization problem in the field of flexible manufacturing systems. All CE-based algorithms produced as instances of the generic schema and considered in the experimental section have been, as far as we know, applied for the first time to this problem. Moreover, the results effectively demonstrate that the new proposals, namely the CE-based memetic algorithms with multiple pmfs, are not only promising but also show better performance than the rest of the algorithms considered in the comparison, including a (non CE-based) memetic algorithm that was the state-of-the-art algorithm in the optimization of this problem.

There are many avenues for future developments. An immediate line of research will try to provide further results of the techniques devised in this work. To be precise, we are currently working on other problems in the area of machine scheduling with encouraging results. Work is also in progress in the area of cooperative models for optimization^{47}, where we aim to integrate memetic CE-based techniques with other memetic methods.

Acknowledgements

Thanks are due to the anonymous reviewers for useful comments and suggestions to improve the readability of the paper. This work is partially supported by Spanish MICINN under projects NEMESIS (TIN2008-05941) and ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project TIC-6083 (DNEMESIS).

References

1. Rubinstein, R.: The cross-entropy method for combinatorial and continuous optimization. Methodology and Computing in Applied Probability **1** (1999) 127–190
2. Chaslot, G., Winands, M.H.M., Szita, I., van den Herik, H.J.: Cross-entropy for monte-carlo tree search. ICGA Journal **31**(3) (2008) 145–156
3. Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. Evol. Comput. **13**(1) (2005) 1–27
4. Li, F., Mannor, S., Lippman, A.: Random tree optimization for energy-efficient broadcast in all-wireless networks. First IEEE International Conference on Sensor and Ad Hoc Communication and Networks, (SECON '04) (Octuber 2004)
5. Ernst, D., Glavic, M., Stan, G.B., Manor, S., Wehenkel, L.: The cross-entropy method for power system combinatorial optimization problems. In: 2007 IEEE Power Tech Conference, Lausanne, Switzerland, IEEE (July 2007) 1290–1295
6. Perelman, L., Ostfeld, A.: Water distribution systems optimal design using cross entropy. In: Genetic and EvolutionaryComputation Conference 2005, New York, NY, USA, ACM (2005) 647–648
7. de Boer, P., Kroese, D., Mannor, S., Rubinstein, R.: A tutorial on the cross-entropy method. Annals of Operations Research **134**(1) (2005) 19–67
8. Rubinstein, R.Y.: Cross-entropy and rare events for maximal cut and partition problems. ACM Transactions on Modeling and Computer Simulation (TOMACS) **12**(1) (2002) 27–53
9. Pelikan, M., Goldberg, D.E., Lobo, F.G.: A survey of optimization by building and using probabilistic models. Comput. Optim. Appl. **21**(1) (2002) 5–20
10. Larrañaga, P., Lozano, J., Mühlenbein, H.: Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. Inteligencia Artificial **7**(19) (2003) 149–168
11. Hu, J., Fu, M.C., Marcus, S.I.: Stochastic optimization using model reference adaptive search. In Chick, S.E., et al., eds.: WSC'03: Proceedings of the 37th Winter Simulation Conference, Orlando, FL, USA, ACM (2005) 811–818
12. Evans, G.E., Keith, J.M., Kroese, D.P.: Parallel cross-entropy optimization. In Henderson, S.G., et al., eds.: WSC'07: Proceedings of the 39th Winter Simulation Conference, Piscataway, NJ, USA, IEEE Press (2007) 2196–2202
13. Lü, Q., Bai, Z.H., Xia, X.Y.: Leader-based parallel cross entropy algorithm for maximum clique problem. Journal of Software **19**(11) (2008) 2899–2907
14. Laguna, M., Duarte, A., Martí, R.: Hybridizing the cross-entropy method: An application to the max-cut problem. Computers & Operations Research **36**(2) (2009) 487–498
15. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Volume 57 of International Series in Operations Research & Management Science. Kluwer Academic Press, New York, USA (2003) 105–144
16. Hart, W., Krasnogor, N., Smith, J.: Memetic Evolutionary Algorithms. In: Recent Advances in Memetic Algorithms. Volume 166 of Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg (2005) 3–27
17. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation **9**(5) (2005) 474–488
18. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation **2** (2012) 1–14
19. Neri, F., Cotta, C., Moscato, P.: Handbook of Memetic Algorithms. Volume 379 of Studies in Computa-

tional Intelligence. Springer-Verlag, Berlin Heidelberg (2012)

20. Raidl, G.: A Unified View on Hybrid Metaheuristics. In Almeida, F., Blesa Aguilera, M., Blum, C., Moreno Vega, J., Pérez Pérez, M., Roli, A., Sampels, M., eds.: Hybrid Metaheuristics. Volume 4030 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg (2006) 1–12
21. Bard, J.F.: A heuristic for minimizing the number of tool switches on a flexible machine. IIE Transactions **20**(4) (1988) 382–391
22. Tang, C.S., Denardo, E.V.: Models arising from a flexible manufacturing machine, Part I: minimization of the number of tool switches. Operations Research **36**(5) (1988) 767–777
23. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, London, UK, Springer-Verlag (1996) 178–187
24. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Volume 192 of Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg (2006)
25. Brownlee, A.E.I., McCall, J.A.W., Zhang, Q., Brown, D.F.: Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In: 2008 IEEE Congress on Evolutionary Computation, Hong Kong, IEEE (1-6 June 2008) 2621–2628
26. De Bonet, J., Isbell, C., Viola, P.: MIMIC: Finding optima by estimating probability densities. Advances in Neural Information Processing Systems **9** (1997) 424–430
27. Baluja, S., Davies, S.: Using optimal dependency-trees for combinational optimization. In Fisher, D.H., ed.: 14th International Conference on Machine Learning, San Francisco CA, Morgan Kaufmann (1997) 30–38
28. Pelikan, M., Goldberg, D., Cantú-Paz, E.: BOA: The bayesian optimization algorithm. In Banzhaf, W., et al., eds.: Genetic and Evolutionary Computation Conference 1999. Volume 1., San Francisco CA, Morgan Kaufmann (1999) 525–532
29. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, USA (1994)
30. Hohfeld, M., Rudolph, G.: Towards a theory of population-based incremental learning. In: 1997 IEEE Conference on Evolutionary Computation, Indianapolis IN, IEEE (1997)
31. Ortíz-García, E.G., Pérez-Bellido, Á.M.: Hybrid cross-entropy method/Hopfield neural network for combinatorial optimization problems. In Yin, H., et al., eds.: Intelligent Data Engineering and Automated Learning 2007. Volume 4881 of Lecture Notes in Computer Science., Birmingham, UK, Springer-Verlag (2007) 1160–1169
32. Campelo, F., Guimaraes, F.G., Ramirez, J.A., Igarashi, H.: Hybrid estimation of distribution algorithm using local function approximations. IEEE Transactions on Magnetics **45**(3) (March 2009) 1558–1561
33. Santana, R., Larrañaga, P., Lozano, J.A.: Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. Journal of Heuristics **14**(5) (2008) 519–547
34. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem. In Larrañaga, P., et al., eds.: Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithm, Springer-Verlag (2006) 281–292
35. Peña, J.M., Robles, V., Larrañaga, P., Herves, V., Rosales, F., Pérez, M.S.: GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In Orchard, R., Yang, C., Ali, M., eds.: 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Volume 3029 of Lecture Notes in Computer Science., Ottawa, Canada, Springer-Verlag (2004) 361–371
36. Zhou, Y., Wang, J., Yin, J.: A discrete estimation of distribution particle swarm optimization for combinatorial optimization problems. In: ICNC '07: Proceedings of the Third International Conference on Natural Computation, Washington, DC, USA, IEEE Computer Society (2007) 80–84
37. Ahn, C.W., An, J., Yoo, J.C.: Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs. Information Sciences (July 2010) In press.
38. Sudholt, D.: The impact of parametrization in memetic evolutionary algorithms. Theoretical Computer Science **410**(26) (2009) 2511–2528
39. Houck, C., Joines, J., Kay, M., Wilson, J.: Empirical investigation of the benefits of partial lamarckianism. Evolutionary Computation **5**(1) (1997) 31–60
40. Nguyen, Q.H., Ong, Y.S., Krasnogor, N.: A study on the design issues of memetic algorithm. In Srinivasan, D., Wang, L., eds.: 2007 IEEE Congress on Evolutionary Computation, Singapore, IEEE (25-28 September 2007) 2390–2397
41. Tzur, M., Altman, A.: Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. IIE Transactions **36**(2) (2004) 95–110

42. Belady, L.: A study of replacement algorithms for virtual storage computers. IBM Systems Journal **5** (1966) 78–101
43. Błażewicz, J., Finke, G.: Scheduling with resource management in manufacturing systems. European Journal of Operational Research **76** (1994) 1–14
44. Oerlemans, A.: Production planning for flexible manufacturing systems. Ph.D. Dissertation, University of Maastricht, Maastricht, Netherlands (October 1992)
45. 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
46. Amaya, J.E., Cotta, C., Fernández, A.J.: A Memetic Algorithm for the Tool Switching Problem. In Blesa, M.J., et al., eds.: Hybrid Metaheuristics, 5th International Workshop, HM 2008. Volume 5296 of Lecture Notes in Computer Science., Málaga, Spain, Springer (2008) 190–202
47. Amaya, J.E., Cotta, C., Fernández, A.J.: Hybrid cooperation models for the tool switching problem. In Pelta, D., et al., eds.: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Volume 284 of Series on Studies in Computational Intelligence., Granada, Spain, Springer (2010) 39–52
48. Zhou, B.H., Xi1, 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) (Mayo 2005) 876–882
49. Al-Fawzan, M.A., Al-Sultan, K.S.: A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. Computers & Industrial Engineering **44**(1) (2003) 35–47
50. 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
51. Amaya, J.E., Cotta, C., Fernández, A.J.: Memetic cooperative models for the tool switching problem. Memetic Computing **3**(3) (2011) 199–216
52. Lehmann, E., D'Abrera, H.: Nonparametrics: statistical methods based on ranks. Prentice-Hall, Englewood Cliffs, NJ (1998)
53. 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
54. Iman, R., Davenport, J.: Approximations of the critical region of the Friedman statistic. Communications in Statistics **9** (1980) 571–595
55. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6** (1979) 65–70

Table 6: Computational results with HC, TS, Beam Search —as proposed in [48]— for several values of the *width beam* (i.e., β), memetic algortihms with HC, standard CE and CEM with 4 pmfs. $\bar{x}$ = mean number of tool switches.$\overline{\sigma}$ = mean standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		HCF	HCP	TSF	TSP	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	4Ri(MAHCP, MATSP, MAHCP)	MAHCP	CE	CEM
$4\zeta_{10}^{9}$	$\bar{x}$	8.4	8.4	8.12	8.08	8.4	8.4	8.4	8.4	8.4	**7.9**	8.1	8.12	8.06
	$\overline{\sigma}$	0.39	0.35	0.22	0.22	0.0	0.0	0.0	0.0	0.0	0.81	0.75	0.36	0.44
$4\zeta_{10}^{10}$	$\bar{x}$	9.34	9.6	9.06	8.8	10.0	9.8	9.6	9.6	9.6	**8.78**	8.94	9.04	9.04
	$\overline{\sigma}$	0.39	0.43	0.24	0.27	0.0	0.0	0.0	0.0	0.0	1.72	1.62	0.43	0.26
$6\zeta_{10}^{15}$	$\bar{x}$	14.38	14.7	13.82	**13.68**	15.2	14.8	14.8	14.8	14.8	13.82	13.89	14.02	14.0
	$\overline{\sigma}$	0.32	0.57	0.26	0.15	0.0	0.0	0.0	0.0	0.0	2.01	1.99	0.47	0.51
$6\zeta_{15}^{12}$	$\bar{x}$	18.32	20.1	17.08	16.46	18.2	17.6	17.6	17.4	17.4	15.92	16.26	16.04	**15.64**
	$\overline{\sigma}$	0.24	0.83	0.53	0.55	0.0	0.0	0.0	0.0	0.0	1.86	1.79	0.9	0.63
$6\zeta_{15}^{20}$	$\bar{x}$	24.76	26.54	23.4	23.02	26.2	25.8	25.2	25.2	25.2	**22.98**	23.18	23.3	23.26
	$\overline{\sigma}$	0.76	0.89	0.63	0.58	0.0	0.0	0.0	0.0	0.0	1.92	1.96	1.0	0.9
$8\zeta_{20}^{15}$	$\bar{x}$	24.46	28.9	24.3	23.62	27.0	26.0	25.6	25.2	25.2	22.66	22.86	22.28	**21.58**
	$\overline{\sigma}$	0.7	1.36	0.86	0.95	0.0	0.0	0.0	0.0	0.0	3.22	3.41	0.89	0.7
$8\zeta_{20}^{16}$	$\bar{x}$	28.76	33.78	28.76	27.92	29.4	29.4	29.4	29.4	29.4	26.96	27.24	26.18	**25.78**
	$\overline{\sigma}$	0.17	1.35	0.93	0.95	0.0	0.0	0.0	0.0	0.0	2.07	2.22	1.09	0.67
$10\zeta_{20}^{20}$	$\bar{x}$	34.4	37.46	31.78	30.72	34.2	33.6	33.4	33.4	33.4	30.18	30.53	30.28	**29.28**
	$\overline{\sigma}$	0.41	1.5	0.91	0.89	0.0	0.0	0.0	0.0	0.0	2.16	2.49	0.69	0.72
$10\zeta_{30}^{25}$	$\bar{x}$	69.6	85.46	85.34	67.72	73.6	70.8	70.8	70.8	70.6	64.2	64.32	60.24	**59.04**
	$\overline{\sigma}$	0.0	2.79	0.78	1.47	0.0	0.0	0.0	0.0	0.0	2.23	2.4	1.46	0.97
$15\zeta_{30}^{40}$	$\bar{x}$	103.0	121.2	104.28	101.72	111.6	110.0	109.2	107.8	107.8	99.1	99.7	96.72	**94.94**
	$\overline{\sigma}$	0.63	2.39	1.56	1.71	0.0	0.0	0.0	0.0	0.0	12.39	12.82	1.59	1.28
$15\zeta_{40}^{30}$	$\bar{x}$	100.34	127.54	130.5	101.9	105.2	103.2	102.8	102.8	102.4	95.08	95.86	88.62	**86.3**
	$\overline{\sigma}$	0.99	3.5	0.82	1.97	0.0	0.0	0.0	0.0	0.0	8.09	7.52	1.33	1.32
$20\zeta_{40}^{60}$	$\bar{x}$	214.42	248.7	255.8	213.74	221.8	220.0	218.8	218.6	218.6	205.78	206.3	199.1	**195.98**
	$\overline{\sigma}$	2.95	4.31	1.08	2.43	0.0	0.0	0.0	0.0	0.0	7.82	8.81	2.13	2.01
$24\zeta_{20}^{30}$	$\bar{x}$	25.8	30.24	25.72	25.04	33.0	32.6	32.4	32.4	32.4	24.42	24.78	23.98	**23.46**
	$\overline{\sigma}$	0.0	1.23	0.99	0.76	0.0	0.0	0.0	0.0	0.0	3.48	3.29	0.83	0.59
$24\zeta_{20}^{36}$	$\bar{x}$	48.48	53.24	47.04	45.9	54.0	54.0	53.8	53.8	53.4	44.82	44.87	45.12	**43.96**
	$\overline{\sigma}$	0.34	1.3	1.1	0.8	0.0	0.0	0.0	0.0	0.0	8.46	7.55	1.28	0.98
$25\zeta_{50}^{40}$	$\bar{x}$	150.88	191.02	196.28	153.58	167.2	164.0	162.8	162.8	161.8	143.22	144.18	131.88	**130.76**
	$\overline{\sigma}$	6.1	4.52	0.9	2.61	0.0	0.0	0.0	0.0	0.0	11.46	11.94	2.0	1.74
$30\zeta_{20}^{40}$	$\bar{x}$	44.4	49.12	42.82	42.12	52.2	50.2	50.2	50.2	50.2	40.6	41,.3	40.84	**40.08**
	$\overline{\sigma}$	0.0	1.58	1.12	1.01	0.0	0.0	0.0	0.0	0.0	4.34	4.41	1.24	1.1

Table 7: Computational results with memetic versions of CE hybridized with HC – i.e., with full neighborhood exploration CEHCF(θ) and partial neighborhood exploration CEHCP(θ) – and several values of θ, i.e., probability of local search. $\bar{x}$= mean number of tool switches. $\overline{\sigma}$ = mean standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		CEHCF(0.001)	CEHCF(0.002)	CEHCF(0.005)	CEHCF(0.01)	CEHCF(0.02)	CEHCF(0.05)	CEHCF(0.1)	CEHCF(0.2)	CEHCF(0.5)	CEHCP(0.001)	CEHCP(0.002)	CEHCP(0.005)	CEHCP(0.01)	CEHCP(0.02)	CEHCP(0.05)	CEHCP(0.1)	CEHCP(0.2)	CEHCP(0.5)
$4\zeta^{9}_{10}$	$\bar{x}$	8.06	8.0	7.98	8.0	7.98	8.04	8.0	8.06	8.08	7.96	7.96	8.04	**7.92**	7.94	7.98	7.94	8.0	7.94
	$\overline{\sigma}$	0.33	0.2	0.2	0.23	0.2	0.32	0.31	0.26	0.28	0.2	0.2	0.19	0.22	0.19	0.31	0.19	0.2	0.24
$4\zeta^{10}_{10}$	$\bar{x}$	8.92	8.88	8.88	9.0	**8.82**	8.92	9.04	8.96	8.9	8.94	8.94	8.88	8.92	8.96	8.94	9.0	8.94	8.94
	$\overline{\sigma}$	0.22	0.27	0.3	0.2	0.33	0.33	0.38	0.36	0.34	0.38	0.33	0.33	0.31	0.32	0.3	0.38	0.43	0.22
$6\zeta^{15}_{10}$	$\bar{x}$	13.74	13.76	13.72	13.74	13.82	**13.7**	13.86	13.76	13.78	13.74	13.92	13.74	13.78	13.8	13.74	13.76	13.82	13.82
	$\overline{\sigma}$	0.24	0.19	0.23	0.22	0.32	0.21	0.36	0.33	0.31	0.24	0.51	0.19	0.28	0.42	0.34	0.25	0.28	0.32
$6\zeta^{12}_{15}$	$\bar{x}$	15.46	15.48	15.32	15.32	15.4	**15.28**	15.38	15.52	15.5	15.42	15.6	15.6	15.52	15.4	15.32	15.36	15.44	15.3
	$\overline{\sigma}$	0.59	0.62	0.73	0.54	0.58	0.7	0.67	0.66	0.63	0.62	0.73	0.64	0.64	0.74	0.66	0.7	0.68	0.57
$6\zeta^{20}_{15}$	$\bar{x}$	22.54	22.54	22.6	22.42	22.74	**22.36**	22.78	22.5	22.46	22.68	22.58	22.66	22.64	22.74	22.54	22.5	22.64	22.76
	$\overline{\sigma}$	0.78	0.69	0.76	0.7	0.75	0.7	0.82	0.58	0.74	0.76	0.83	0.76	0.94	0.77	0.85	0.78	0.68	0.83
$8\zeta^{15}_{20}$	$\bar{x}$	21.18	21.2	21.26	21.44	**21.04**	21.2	21.28	21.18	21.2	21.26	21.32	21.16	21.18	21.08	21.1	21.24	21.42	21.3
	$\overline{\sigma}$	0.82	0.81	0.52	0.74	0.64	0.82	0.79	0.65	0.72	0.81	0.76	0.54	0.72	0.61	0.57	0.56	0.83	0.63
$8\zeta^{16}_{20}$	$\bar{x}$	25.36	25.36	25.26	25.46	25.32	25.26	25.4	25.54	25.32	25.36	25.26	25.32	25.36	25.5	25.28	25.46	25.34	**25.18**
	$\overline{\sigma}$	0.68	0.95	0.71	0.79	0.82	0.85	0.67	0.85	0.62	0.81	0.78	0.59	0.66	0.79	0.79	0.76	0.75	0.52
$10\zeta^{20}_{20}$	$\bar{x}$	28.56	28.72	28.84	28.54	28.66	28.62	28.66	28.84	28.62	28.72	28.96	28.74	28.74	28.62	28.92	28.7	**28.42**	28.8
	$\overline{\sigma}$	0.65	0.71	0.68	0.87	0.8	0.8	0.72	0.78	0.8	0.81	0.71	0.68	0.88	0.75	0.76	0.72	0.78	0.79
$10\zeta^{25}_{30}$	$\bar{x}$	58.98	59.02	59.06	58.96	58.9	58.66	58.76	58.88	59.06	58.5	58.68	58.58	58.52	**58.32**	58.72	58.62	58.6	58.7
	$\overline{\sigma}$	0.85	0.91	1.09	0.79	1.1	1.04	1.04	1.04	1.22	0.86	1.02	0.83	0.84	0.81	1.12	0.96	0.93	0.88
$15\zeta^{40}_{30}$	$\bar{x}$	93.38	**93.1**	93.28	93.8	93.4	93.96	93.3	93.92	93.56	93.4	93.24	93.4	93.54	93.46	93.32	93.12	93.78	93.24
	$\overline{\sigma}$	1.31	1.52	1.5	1.29	1.36	1.22	1.37	1.38	1.43	1.21	1.63	1.26	1.4	1.57	1.49	1.41	1.18	1.5
$15\zeta^{30}_{40}$	$\bar{x}$	95.96	95.48	96.58	96.42	96.06	96.52	96.1	96.0	95.56	88.84	89.08	89.9	**88.1**	89.16	89.3	88.96	89.52	89.12
	$\overline{\sigma}$	2.38	2.48	2.56	2.64	2.41	2.16	2.25	3.04	2.49	1.9	2.27	2.3	1.88	1.74	2.08	2.05	2.07	2.24
$20\zeta^{60}_{40}$	$\bar{x}$	196.26	196.58	196.68	197.14	196.46	196.04	196.2	196.48	196.8	194.72	194.18	194.5	194.02	**194.0**	194.5	194.68	194.16	194.48
	$\overline{\sigma}$	2.28	2.4	2.4	2.69	2.07	2.47	2.1	2.47	2.3	1.68	2.06	1.49	1.48	1.79	1.9	1.62	2.16	1.44
$24\zeta^{30}_{20}$	$\bar{x}$	23.0	23.18	23.24	**22.94**	23.1	23.06	22.96	23.14	22.96	23.14	23.24	23.16	23.26	23.12	23.06	22.9	22.96	23.08
	$\overline{\sigma}$	0.85	0.75	0.9	0.66	0.64	0.7	0.74	0.74	0.79	0.64	0.67	0.66	0.56	0.79	0.67	0.67	0.64	0.65
$24\zeta^{36}_{20}$	$\bar{x}$	43.04	43.2	42.96	**42.82**	42.98	43.16	42.88	43.0	42.98	43.08	43.0	42.94	43.16	43.14	42.9	43.3	42.98	42.96
	$\overline{\sigma}$	1.01	0.95	1.0	1.04	1.07	1.23	1.16	0.88	1.08	1.06	1.3	1.04	0.97	1.03	0.72	1.09	0.92	1.05
$25\zeta^{40}_{50}$	$\bar{x}$	157.7	156.72	156.88	157.92	157.86	156.12	155.9	156.76	156.32	138.72	139.58	138.58	138.78	**138.2**	138.22	138.44	138.26	138.64
	$\overline{\sigma}$	3.49	3.61	3.66	3.73	4.03	3.65	3.51	3.61	3.72	2.93	3.04	3.17	2.17	2.77	2.38	2.57	2.49	2.9
$30\zeta^{40}_{20}$	$\bar{x}$	39.38	39.36	39.44	39.26	39.48	39.38	39.32	39.4	39.2	39.34	39.4	39.38	39.46	39.16	39.28	39.26	**38.98**	39.06
	$\overline{\sigma}$	1.17	1.01	0.99	0.85	1.04	0.91	0.96	0.87	1.13	0.79	0.99	1.01	0.97	1.0	1.05	1.22	0.93	0.96

Table 8: Computational results with memetic versions of CE hybridized with TS – i.e., with full neighborhood exploration CETSF(θ) and partial neighborhood exploration CETSP(θ) – and several values of θ, i.e., probability of local search. $\bar{x}$= mean number of tool switches. $\overline{\sigma}$ = mean standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		CETSF(0.001)	CETSF(0.002)	CETSF(0.005)	CETSF(0.01)	CETSF(0.02)	CETSF(0.05)	CETSF(0.1)	CETSF(0.2)	CETSF(0.5)	CETSP(0.001)	CETSP(0.002)	CETSP(0.005)	CETSP(0.01)	CETSP(0.02)	CETSP(0.05)	CETSP(0.1)	CETSP(0.2)	CETSP(0.5)
$4\zeta_{10}^{9}$	$\bar{x}$	8.18	8.2	8.26	8.14	8.18	8.34	8.16	8.04	8.16	8.1	8.14	8.22	8.16	8.12	8.16	8.12	8.1	**8.06**
	$\overline{\sigma}$	0.4	0.25	0.35	0.29	0.42	0.51	0.39	0.32	0.41	0.41	0.36	0.38	0.39	0.41	0.32	0.4	0.32	0.19
$4\zeta_{10}^{10}$	$\bar{x}$	9.2	9.14	9.14	9.18	9.2	9.16	9.22	9.26	9.22	9.3	9.18	9.28	**9.06**	9.12	9.14	9.22	9.1	9.18
	$\overline{\sigma}$	0.4	0.41	0.49	0.39	0.49	0.48	0.42	0.39	0.41	0.52	0.37	0.46	0.34	0.47	0.51	0.49	0.41	0.36
$6\zeta_{10}^{15}$	$\bar{x}$	**14.02**	14.18	14.04	14.1	14.2	14.16	14.1	14.26	14.06	14.22	13.98	14.08	14.12	14.04	14.16	14.04	14.12	**14.02**
	$\overline{\sigma}$	0.46	0.57	0.47	0.51	0.64	0.63	0.62	0.63	0.48	0.57	0.46	0.54	0.5	0.6	0.62	0.56	0.6	0.51
$6\zeta_{15}^{12}$	$\bar{x}$	16.18	16.26	16.32	16.14	16.34	16.14	16.34	16.22	16.28	16.22	16.32	16.32	16.28	16.18	16.18	16.2	16.06	**16.04**
	$\overline{\sigma}$	0.88	0.75	0.81	0.64	0.99	0.64	0.82	0.6	0.86	0.97	0.82	0.84	0.83	0.86	0.74	0.65	0.88	0.62
$6\zeta_{15}^{20}$	$\bar{x}$	23.54	23.7	23.82	23.62	23.4	23.62	23.82	23.98	23.34	23.4	23.82	23.62	23.36	23.58	**23.32**	23.56	23.48	23.64
	$\overline{\sigma}$	0.89	0.87	0.81	0.96	0.85	0.89	0.76	0.93	0.98	0.87	0.97	0.92	0.9	1.0	0.77	0.9	0.94	0.77
$8\zeta_{20}^{15}$	$\bar{x}$	23.38	23.66	23.38	23.3	23.38	23.1	23.88	23.44	23.86	22.54	22.6	22.24	22.44	22.62	**22.08**	22.24	22.5	22.52
	$\overline{\sigma}$	1.18	1.06	0.93	1.11	1.25	0.96	1.22	1.27	1.21	0.95	0.85	0.88	0.95	0.97	0.87	0.88	1.02	0.74
$8\zeta_{20}^{16}$	$\bar{x}$	27.3	27.6	27.36	27.56	27.4	27.44	27.36	27.56	27.38	26.64	26.72	26.62	**26.42**	26.64	26.58	26.64	26.58	26.56
	$\overline{\sigma}$	0.91	1.08	0.96	1.2	1.06	0.86	1.04	1.14	1.08	1.14	1.02	0.82	0.97	0.96	0.82	0.77	1.04	0.79
$10\zeta_{20}^{20}$	$\bar{x}$	31.08	30.58	30.78	30.8	30.76	30.68	30.66	30.46	30.76	30.4	30.46	**29.98**	30.04	30.3	30.22	30.44	30.04	30.44
	$\overline{\sigma}$	1.08	1.2	1.1	1.08	0.95	1.0	1.14	0.92	0.99	0.79	1.01	0.93	0.75	0.83	0.88	0.98	1.02	1.06
$10\zeta_{30}^{25}$	$\bar{x}$	70.56	70.02	71.32	71.06	70.72	71.08	70.5	71.06	69.7	61.96	61.52	**61.42**	61.8	61.88	61.56	61.8	61.98	61.78
	$\overline{\sigma}$	2.9	2.81	3.02	2.69	2.87	3.04	2.65	2.44	2.98	1.2	1.27	1.34	1.33	1.25	1.64	1.21	1.65	1.56
$15\zeta_{30}^{40}$	$\bar{x}$	102.94	103.66	103.84	104.84	103.9	104.2	104.52	103.98	104.36	96.84	**96.44**	96.48	96.9	97.04	97.3	96.82	97.2	96.56
	$\overline{\sigma}$	2.57	3.2	3.27	2.75	2.71	3.22	2.85	2.61	3.45	1.77	1.53	1.41	1.84	1.81	1.57	1.33	1.87	1.84
$15\zeta_{40}^{30}$	$\bar{x}$	117.48	118.1	118.18	117.36	117.82	116.96	118.08	117.26	117.36	100.66	101.1	100.46	100.92	100.9	100.38	100.46	**100.28**	100.9
	$\overline{\sigma}$	4.52	4.1	4.03	4.83	4.2	4.11	4.68	3.77	3.6	3.31	2.68	3.29	2.43	2.04	2.86	2.8	2.44	2.78
$20\zeta_{40}^{60}$	$\bar{x}$	224.32	222.32	223.52	223.84	222.3	224.06	222.28	222.58	224.0	200.54	200.78	200.64	200.46	201.32	**200.32**	201.22	200.8	200.42
	$\overline{\sigma}$	4.17	4.47	4.62	4.29	4.62	4.0	4.47	4.24	5.49	2.2	2.14	2.01	2.15	2.73	2.5	1.94	2.57	2.27
$24\zeta_{20}^{30}$	$\bar{x}$	25.82	26.2	25.9	25.6	25.86	25.6	25.62	25.86	26.1	24.66	24.58	24.32	24.5	24.34	24.32	24.66	24.28	**24.22**
	$\overline{\sigma}$	1.16	1.05	1.34	1.34	1.14	1.24	1.13	1.16	1.36	0.91	0.83	0.93	0.81	1.09	0.95	0.83	1.0	0.83
$24\zeta_{20}^{36}$	$\bar{x}$	45.5	45.36	45.6	45.44	45.6	45.26	45.6	45.48	45.66	45.24	45.28	**44.58**	45.02	45.1	45.0	45.1	44.76	44.9
	$\overline{\sigma}$	1.34	1.32	1.27	1.29	1.42	1.33	1.46	1.43	1.19	1.27	1.25	1.05	1.17	1.11	1.3	1.35	0.88	1.2
$25\zeta_{50}^{40}$	$\bar{x}$	186.16	187.36	187.84	187.32	187.1	186.68	186.04	187.36	187.62	160.06	160.76	160.66	159.74	**159.42**	159.82	160.4	159.5	159.84
	$\overline{\sigma}$	5.61	5.87	5.37	6.23	5.65	6.54	7.74	5.52	5.5	3.78	3.88	4.48	3.79	4.37	4.64	3.88	3.94	3.74
$30\zeta_{20}^{40}$	$\bar{x}$	42.0	41.84	41.96	42.16	42.02	41.68	41.78	42.04	42.12	41.26	41.08	41.04	41.06	41.28	41.08	41.36	**41.02**	41.3
	$\overline{\sigma}$	1.48	1.08	1.18	1.37	1.51	1.16	1.28	1.24	1.55	1.32	1.07	1.04	0.93	0.97	1.23	1.28	1.16	1.26

Table 9: Computational results with PBIL(ρ,α) and UMDA(ρ) for $\rho \in \{0.01, 0.2\}$ and several values of α ranging in $[0.60, 0.95]$. $\bar{x}$= mean number of tool switches. $\overline{\sigma}$ = standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		PBIL(0.01/0.60)	PBIL(0.01/0.65)	PBIL(0.01/0.70)	PBIL(0.01/0.75)	PBIL(0.01/0.80)	PBIL(0.01/0.85)	PBIL(0.01/0.90)	PBIL(0.01/0.95)	UMDA(0.01)	PBIL(0.2/0.60)	PBIL(0.2/0.65)	PBIL(0.2/0.70)	PBIL(0.2/0.75)	PBIL(0.2/0.80)	PBIL(0.2/0.85)	PBIL(0.2/0.90)	PBIL(0.2/0.95)	UMDA(0.2)
$4\zeta_{10}^{9}$	$\bar{x}$	9.8	**9.7**	10.08	10.18	9.92	10.1	10.26	10.5	10.72	12.7	12.38	12.46	12.36	12.5	12.42	12.44	12.78	12.6
	$\overline{\sigma}$	0.92	0.75	0.76	0.95	0.68	0.95	1.05	1.1	0.74	1.41	1.32	1.64	1.46	1.36	1.46	1.21	1.44	1.71
$4\zeta_{10}^{10}$	$\bar{x}$	**10.74**	11.02	11.16	11.22	11.38	11.16	11.78	11.62	12.72	13.5	13.56	13.36	13.84	13.28	13.12	13.14	13.38	13.5
	$\overline{\sigma}$	0.75	1.04	0.81	0.82	0.88	1.0	1.07	1.04	1.2	1.52	1.41	1.33	1.73	1.7	1.38	1.47	1.43	1.39
$6\zeta_{10}^{15}$	$\bar{x}$	**16.44**	17.12	16.58	17.14	17.24	17.16	17.76	17.96	18.56	19.58	20.18	19.36	19.86	19.8	19.84	19.82	19.8	19.72
	$\overline{\sigma}$	1.1	1.04	0.98	1.14	0.92	1.26	0.99	1.35	1.35	1.88	1.92	1.71	1.73	1.71	1.63	1.86	1.56	1.49
$6\zeta_{15}^{12}$	$\bar{x}$	25.34	24.94	**24.86**	25.16	25.2	25.36	25.72	**24.86**	25.94	26.92	27.3	27.78	27.48	27.28	26.68	27.1	27.0	27.24
	$\overline{\sigma}$	2.37	2.57	2.35	2.2	2.32	2.67	2.44	2.42	3.39	2.61	2.34	1.71	2.39	2.34	2.2	2.67	2.31	2.52
$6\zeta_{15}^{20}$	$\bar{x}$	31.84	31.68	32.3	31.66	31.76	31.72	32.12	**31.6**	31.82	33.56	33.66	33.86	34.22	34.12	33.42	33.48	33.8	33.9
	$\overline{\sigma}$	2.19	2.49	2.14	2.21	2.34	2.28	2.67	2.21	2.33	2.25	2.06	2.31	1.89	2.16	2.42	2.22	2.13	2.21
$8\zeta_{20}^{15}$	$\bar{x}$	37.44	37.24	37.3	37.4	38.22	37.46	**36.94**	37.82	37.12	39.08	38.44	39.36	39.26	38.16	39.36	38.66	39.68	38.28
	$\overline{\sigma}$	2.63	3.06	2.56	2.65	2.8	2.62	2.37	2.9	3.79	2.65	2.25	2.65	3.06	2.49	2.81	3.08	3.0	2.65
$8\zeta_{20}^{16}$	$\bar{x}$	41.68	42.4	43.16	42.42	**42.0**	42.06	42.72	43.18	42.46	44.44	43.32	43.5	43.58	43.48	43.72	44.3	43.3	43.22
	$\overline{\sigma}$	3.21	2.92	2.81	3.19	3.71	3.19	3.44	3.12	4.56	2.83	3.0	3.63	2.69	3.15	3.06	2.65	2.88	2.91
$10\zeta_{20}^{20}$	$\bar{x}$	47.12	47.04	47.8	47.2	46.68	47.32	47.16	**46.54**	48.9	48.3	47.74	47.74	47.62	47.92	48.64	47.88	49.06	47.7
	$\overline{\sigma}$	3.31	2.9	2.58	3.15	3.12	3.4	3.39	3.41	6.2	3.82	2.74	3.33	3.05	3.46	3.17	3.77	2.72	3.74
$10\zeta_{30}^{25}$	$\bar{x}$	102.16	101.98	101.7	**99.66**	101.36	100.22	101.36	100.26	101.76	101.78	102.08	103.32	102.92	102.98	102.32	102.56	102.46	102.92
	$\overline{\sigma}$	4.33	5.12	4.41	3.94	4.82	5.17	4.12	4.49	5.06	4.92	3.4	3.8	3.83	4.18	4.59	5.32	4.53	5.14
$15\zeta_{30}^{40}$	$\bar{x}$	138.48	139.16	139.48	140.86	139.14	139.42	139.52	**137.8**	143.28	140.78	139.98	141.82	139.66	142.06	141.22	140.72	141.56	140.3
	$\overline{\sigma}$	4.19	4.56	6.7	4.6	5.31	5.24	5.09	5.56	7.75	5.34	4.78	5.68	4.93	4.67	4.12	5.74	5.48	5.5
$15\zeta_{40}^{30}$	$\bar{x}$	**146.16**	148.04	147.92	148.28	147.68	147.76	149.4	148.1	148.76	150.3	149.4	147.94	150.16	148.8	148.5	146.94	149.48	147.96
	$\overline{\sigma}$	6.5	5.08	4.44	5.41	5.24	5.01	6.35	6.47	6.24	6.3	6.8	6.17	4.99	5.21	4.8	5.29	4.69	5.09
$20\zeta_{40}^{60}$	$\bar{x}$	276.94	276.7	274.7	**274.08**	275.1	277.14	274.8	277.02	280.2	277.86	278.02	277.02	278.7	276.5	275.9	275.3	278.76	275.0
	$\overline{\sigma}$	8.23	7.83	6.12	8.5	6.03	8.6	6.77	6.77	10.27	6.28	8.31	7.93	8.42	8.26	6.39	6.11	8.19	7.59
$24\zeta_{20}^{30}$	$\bar{x}$	37.96	37.6	**37.2**	37.34	37.58	37.98	37.88	37.3	39.5	37.54	38.84	38.3	38.86	38.16	39.02	38.82	39.0	39.28
	$\overline{\sigma}$	2.75	3.03	3.0	2.7	2.8	2.44	2.67	2.58	4.44	2.67	2.76	2.99	2.87	2.71	2.65	2.26	2.57	2.51
$24\zeta_{20}^{36}$	$\bar{x}$	63.92	**63.16**	63.3	63.7	63.64	63.76	63.94	64.26	66.88	65.12	64.78	64.88	65.24	65.94	64.96	63.62	64.44	65.78
	$\overline{\sigma}$	3.71	3.66	2.71	4.38	4.04	3.96	3.45	4.39	6.93	3.27	3.11	3.29	3.68	3.55	3.72	4.01	3.85	4.06
$25\zeta_{50}^{40}$	$\bar{x}$	215.88	216.54	217.76	216.66	217.88	217.2	217.6	217.96	222.9	216.6	218.58	217.4	218.04	219.36	216.46	219.54	218.52	**215.42**
	$\overline{\sigma}$	7.31	6.42	7.01	5.84	6.84	7.9	7.08	7.25	7.81	6.46	6.71	7.1	5.99	7.11	6.37	6.29	7.02	6.59
$30\zeta_{20}^{40}$	$\bar{x}$	59.52	60.68	**57.88**	59.62	59.44	59.88	58.96	60.08	62.94	61.06	60.94	60.74	61.4	61.52	62.36	60.84	61.1	60.62
	$\overline{\sigma}$	4.34	3.34	3.98	3.65	3.38	3.55	3.57	3.71	6.34	3.71	3.6	3.76	4.28	3.24	3.07	3.38	3.19	3.54

Table 10: Computational results with memetic versions of CE with 4 pmfs hybridized with HC – i.e., with full neighborhood exploration CEMHCF(θ) and partial neighborhood exploration CEMHCP(θ) – and several values of θ, i.e., probability of local search. $\overline{x}$= mean number of tool switches. $\overline{\sigma}$ = mean standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		CEMHCF(0.001)	CEMHCF(0.002)	CEMHCF(0.005)	CEMHCF(0.01)	CEMHCF(0.02)	CEMHCF(0.05)	CEMHCF(0.1)	CEMHCF(0.2)	CEMHCF(0.5)	CEMHCP(0.001)	CEMHCP(0.002)	CEMHCP(0.005)	CEMHCP(0.01)	CEMHCP(0.02)	CEMHCP(0.05)	CEMHCP(0.1)	CEMHCP(0.2)	CEMHCP(0.5)
$4\zeta_{10}^{9}$	$\overline{x}$	8.1	8.02	8.06	8.14	8.08	8.0	7.92	8.04	7.92	7.79	7.8	7.73	7.7	7.73	7.79	7.71	7.69	**7.67**
	$\overline{\sigma}$	0.21	0.26	0.41	0.35	0.41	0.33	0.22	0.26	0.23	1.04	1.06	1.04	1.04	1.06	1.12	1.01	1.01	0.95
$4\zeta_{10}^{10}$	$\overline{x}$	9.16	9.04	9.12	9.06	8.98	8.92	8.96	**8.78**	8.9	9.24	9.02	9.2	9.02	9.02	9.02	8.94	8.98	8.96
	$\overline{\sigma}$	0.34	0.31	0.33	0.39	0.33	0.34	0.32	0.27	0.35	0.47	0.29	0.4	0.37	0.32	0.27	0.25	0.44	0.34
$6\zeta_{10}^{15}$	$\overline{x}$	14.04	14.0	14.16	13.92	13.88	13.8	13.7	13.86	13.8	14.04	14.02	13.76	13.96	13.88	13.84	13.76	13.82	**13.66**
	$\overline{\sigma}$	0.49	0.49	0.51	0.45	0.45	0.39	0.22	0.38	0.27	0.56	0.58	0.26	0.54	0.41	0.42	0.39	0.23	0.14
$6\zeta_{15}^{12}$	$\overline{x}$	15.48	15.58	15.6	15.62	15.5	15.4	15.42	15.52	**15.26**	15.54	15.72	15.72	15.54	15.64	15.42	15.58	15.36	15.28
	$\overline{\sigma}$	0.65	0.68	0.66	0.61	0.66	0.51	0.64	0.71	0.56	0.53	0.76	0.72	0.6	0.54	0.62	0.67	0.58	0.63
$6\zeta_{15}^{20}$	$\overline{x}$	23.08	23.08	22.9	22.82	22.86	22.7	22.7	22.6	**22.46**	22.98	23.16	22.7	22.74	22.68	22.66	22.66	22.6	22.54
	$\overline{\sigma}$	0.85	0.95	0.77	0.79	0.77	0.81	0.8	0.86	0.68	0.84	0.81	0.9	0.71	0.76	0.73	0.65	0.78	0.63
$8\zeta_{20}^{15}$	$\overline{x}$	21.44	21.38	21.32	21.44	21.22	21.26	21.08	20.94	20.94	21.48	21.46	21.76	21.44	21.5	21.6	21.16	21.16	**20.84**
	$\overline{\sigma}$	0.69	0.66	0.61	0.61	0.62	0.61	0.63	0.62	0.65	0.73	0.68	0.76	0.7	0.82	0.71	0.66	0.75	0.67
$8\zeta_{20}^{16}$	$\overline{x}$	25.46	25.42	25.66	25.32	25.32	25.36	25.24	25.18	**24.96**	25.74	25.42	25.46	25.7	25.52	25.28	25.26	24.98	25.18
	$\overline{\sigma}$	0.66	0.73	0.66	0.66	0.77	0.64	0.79	0.6	0.56	0.75	0.82	0.64	0.77	0.81	0.71	0.73	0.55	0.66
$10\zeta_{20}^{20}$	$\overline{x}$	29.12	28.9	28.84	29.02	28.9	28.68	28.8	28.62	28.74	29.3	29.32	29.06	29.1	29.12	28.72	28.84	28.64	**28.36**
	$\overline{\sigma}$	0.69	0.75	0.82	0.83	0.83	0.67	0.63	0.71	0.59	0.7	0.78	0.84	0.7	0.82	0.82	0.73	0.69	0.66
$10\zeta_{30}^{25}$	$\overline{x}$	58.93	59.01	59.11	58.94	58.82	58.82	58.62	58.48	58.96	59.08	58.78	59.06	58.84	58.86	58.78	58.8	58.48	**58.24**
	$\overline{\sigma}$	5.07	4.51	5.06	5.0	4.64	1.04	0.77	0.78	1.02	0.81	0.93	0.83	0.92	0.86	0.89	0.98	0.87	0.76
$15\zeta_{30}^{40}$	$\overline{x}$	94.5	94.14	94.44	94.32	94.16	93.98	93.46	93.4	93.02	94.36	94.2	93.96	94.12	93.78	93.98	93.92	93.64	**92.3**
	$\overline{\sigma}$	1.63	1.35	1.22	1.38	1.28	1.52	1.13	1.34	1.32	1.16	1.29	1.26	1.35	1.2	1.39	1.07	1.25	1.42
$15\zeta_{40}^{30}$	$\overline{x}$	85.07	**84.73**	85.12	84.93	84.98	85.2	86.52	88.83	96.02	86.36	86.32	86.2	86.92	86.26	86.26	86.6	87.18	90.86
	$\overline{\sigma}$	6.85	5.94	7.94	5.85	6.19	7.32	8.1	10.1	10.08	1.22	0.97	1.15	6.51	1.13	1.24	0.97	1.15	1.9
$20\zeta_{40}^{60}$	$\overline{x}$	195.28	195.56	195.3	195.6	194.96	194.76	194.88	194.76	198.16	195.76	195.88	195.4	195.26	195.12	195.56	195.1	195.0	**193.09**
	$\overline{\sigma}$	1.8	1.61	1.77	1.65	1.77	1.62	1.8	1.6	2.66	1.3	1.86	1.83	1.94	2.08	1.92	2.16	1.91	10.13
$24\zeta_{20}^{30}$	$\overline{x}$	23.38	23.38	23.24	23.24	23.4	23.16	23.12	23.06	**22.78**	23.28	23.26	23.34	23.32	23.18	23.14	23.04	23.04	22.98
	$\overline{\sigma}$	0.6	0.82	0.68	0.65	0.66	0.74	0.65	0.65	0.7	0.54	0.65	0.59	0.6	0.66	0.61	0.76	0.62	0.65
$24\zeta_{20}^{36}$	$\overline{x}$	43.8	43.66	45.42	43.47	43.34	43.26	42.94	42.67	**42.63**	43.72	44.08	43.48	43.58	43.64	43.32	43.06	43.18	42.96
	$\overline{\sigma}$	1.07	1.0	5.42	5.23	5.23	4.98	5.22	5.1	4.73	1.28	1.14	0.95	1.11	1.27	1.2	1.08	0.93	1.0
$25\zeta_{50}^{40}$	$\overline{x}$	131.36	131.6	132.5	133.12	134.54	137.2	141.74	146.08	155.74	**131.22**	132.0	131.7	131.52	131.58	132.18	133.3	135.6	139.42
	$\overline{\sigma}$	1.61	2.04	1.79	1.86	2.63	2.21	2.32	2.78	2.93	1.79	2.08	2.12	1.93	1.96	2.35	1.9	2.55	2.1
$30\zeta_{20}^{40}$	$\overline{x}$	40.1	39.8	39.76	40.0	39.4	39.82	39.08	39.44	**38.78**	39.82	40.02	39.92	40.0	40.22	39.43	39.4	39.24	38.87
	$\overline{\sigma}$	1.0	0.86	0.83	0.94	0.78	0.95	0.98	0.89	0.82	0.78	1.0	0.93	4.04	4.8	5.17	5.06	4.54	4.32

Table 11: Computational results with memetic versions of CE with 4 pmfs hybridized with TS – i.e., with full neighborhood exploration CEMTSF(θ) and partial neighborhood exploration CEMTSP(θ) – and several values of θ, i.e., probability of local search. $\bar{x}$= mean number of tool switches. $\overline{\sigma}$ = mean standard deviation. Recall we are using the notation $C\zeta_n^m$, where C is the magazine capacity, m is the total number of tools and n is the number of jobs.

		CEMTSF(0.001)	CEMTSF(0.002)	CEMTSF(0.005)	CEMTSF(0.01)	CEMTSF(0.02)	CEMTSF(0.05)	CEMTSF(0.1)	CEMTSF(0.2)	CEMTSF(0.5)	CEMTSP(0.001)	CEMTSP(0.002)	CEMTSP(0.005)	CEMTSP(0.01)	CEMTSP(0.02)	CEMTSP(0.05)	CEMTSP(0.1)	CEMTSP(0.2)	CEMTSP(0.5)
$4\zeta_{10}^{9}$	$\bar{x}$	8.06	8.22	8.14	8.18	8.06	8.08	8.1	8.08	8.22	8.06	8.0	8.1	**7.96**	8.16	8.14	8.08	8.18	8.22
	$\overline{\sigma}$	0.34	0.38	0.35	0.42	0.22	0.29	0.28	0.35	0.43	0.43	0.2	0.37	0.33	0.35	0.38	0.37	0.41	0.4
$4\zeta_{10}^{10}$	$\bar{x}$	9.02	9.12	9.12	**8.92**	9.1	9.0	9.32	9.16	9.24	9.12	9.12	9.02	9.1	9.16	9.22	9.02	9.16	9.12
	$\overline{\sigma}$	0.36	0.35	0.28	0.33	0.43	0.32	0.46	0.31	0.48	0.38	0.41	0.29	0.38	0.43	0.4	0.41	0.46	0.4
$6\zeta_{10}^{15}$	$\bar{x}$	14.06	13.96	13.98	14.04	14.12	14.02	14.04	14.02	14.16	13.94	**13.92**	13.96	14.1	13.98	14.08	14.0	14.04	14.1
	$\overline{\sigma}$	0.45	0.47	0.52	0.55	0.53	0.56	0.65	0.64	0.66	0.37	0.49	0.48	0.47	0.43	0.58	0.53	0.57	0.53
$6\zeta_{15}^{12}$	$\bar{x}$	15.64	**15.5**	15.64	15.62	15.72	15.6	15.6	15.76	15.94	15.8	15.74	15.7	15.7	15.7	15.72	15.66	15.6	15.74
	$\overline{\sigma}$	0.63	0.71	0.61	0.6	0.73	0.69	0.94	0.7	0.84	0.78	0.63	0.71	0.63	0.53	0.74	0.67	0.57	0.61
$6\zeta_{15}^{20}$	$\bar{x}$	23.2	23.3	23.16	23.02	23.08	23.0	23.28	23.22	23.06	23.32	23.14	**22.9**	23.18	22.98	23.24	23.34	23.02	23.26
	$\overline{\sigma}$	0.88	0.67	0.81	0.84	0.81	0.72	0.94	0.85	0.84	0.77	0.86	0.86	0.77	0.86	0.77	0.81	0.77	0.82
$8\zeta_{20}^{15}$	$\bar{x}$	21.58	21.52	21.4	21.48	21.56	21.66	21.46	21.78	22.86	21.54	21.48	**21.38**	21.6	21.58	21.62	21.62	21.52	21.8
	$\overline{\sigma}$	0.78	0.69	0.65	0.71	0.58	0.66	0.59	0.79	1.07	0.76	0.72	0.71	0.72	0.73	0.57	0.61	0.75	0.56
$8\zeta_{20}^{16}$	$\bar{x}$	25.48	25.54	**25.4**	25.7	25.56	25.54	25.58	25.78	26.7	25.6	25.66	25.62	25.64	25.44	25.56	25.64	25.74	25.7
	$\overline{\sigma}$	0.56	0.63	0.65	0.82	0.66	0.62	0.7	0.74	0.82	0.71	0.63	0.65	0.74	0.77	0.83	0.69	0.74	0.81
$10\zeta_{20}^{20}$	$\bar{x}$	29.14	**29.1**	29.16	29.36	29.36	29.18	29.2	29.3	29.98	29.24	29.14	29.24	**29.1**	29.2	29.32	29.14	29.28	29.36
	$\overline{\sigma}$	0.62	0.73	0.8	0.7	0.91	0.69	0.79	0.68	1.0	0.83	0.62	0.87	0.78	0.8	0.75	0.87	0.7	0.77
$10\zeta_{30}^{25}$	$\bar{x}$	59.1	59.1	59.0	59.08	59.16	59.32	60.18	62.74	69.82	59.1	58.78	59.38	**58.88**	58.98	59.36	59.15	59.22	60.8
	$\overline{\sigma}$	1.0	0.99	4.97	0.87	0.94	0.88	1.09	1.55	2.45	0.84	1.09	1.19	0.94	0.98	1.12	4.46	0.76	1.24
$15\zeta_{30}^{40}$	$\bar{x}$	94.5	94.72	94.56	94.4	96.34	94.94	94.7	96.78	103.12	94.88	94.2	**94.14**	94.3	94.26	94.18	94.4	94.42	95.36
	$\overline{\sigma}$	1.44	1.19	1.43	1.66	5.81	0.88	1.39	1.83	2.07	1.34	1.4	1.36	1.43	1.5	1.44	1.59	1.48	1.3
$15\zeta_{40}^{30}$	$\bar{x}$	86.62	86.42	86.5	87.28	88.6	93.72	99.64	104.96	114.86	86.62	86.4	**86.0**	86.38	86.62	87.3	88.7	117.12	99.34
	$\overline{\sigma}$	1.05	1.41	1.13	1.33	1.97	2.92	3.36	3.54	3.4	6.3	6.55	1.26	1.14	1.43	1.38	1.9	3.9	2.16
$20\zeta_{40}^{60}$	$\bar{x}$	195.68	195.6	195.66	195.76	196.52	196.08	199.92	208.52	220.84	196.56	196.04	195.82	195.52	195.86	196.22	**195.36**	196.3	200.38
	$\overline{\sigma}$	1.89	1.84	2.07	1.76	2.64	1.88	2.5	3.55	4.4	9.97	2.04	1.87	1.94	1.9	1.58	1.85	1.65	2.7
$24\zeta_{20}^{30}$	$\bar{x}$	23.16	23.36	23.24	23.5	**23.1**	23.28	23.38	23.56	25.7	**23.1**	23.14	23.16	23.56	23.3	23.36	23.34	23.62	23.7
	$\overline{\sigma}$	0.66	0.85	0.7	0.7	0.56	0.74	0.69	0.85	1.14	0.6	0.63	0.66	0.85	0.72	0.76	0.79	0.67	0.76
$24\zeta_{20}^{36}$	$\bar{x}$	44.14	43.64	43.74	43.68	**43.5**	43.72	44.16	43.64	44.56	43.78	43.92	43.77	43.89	43.88	43.91	43.73	43.89	43.9
	$\overline{\sigma}$	1.05	0.89	0.91	1.14	0.89	1.0	1.15	1.02	1.04	0.93	1.31	5.39	5.48	5.45	5.28	5.57	5.35	4.98
$25\zeta_{50}^{40}$	$\bar{x}$	132.36	133.02	134.64	138.64	143.82	151.91	160.76	170.1	182.5	131.62	131.88	**131.48**	132.44	134.0	136.66	142.0	146.78	157.46
	$\overline{\sigma}$	2.41	2.35	3.22	3.18	3.25	28.52	23.65	5.07	6.12	1.95	1.64	1.89	2.02	2.21	2.27	2.28	3.0	3.21
$30\zeta_{20}^{40}$	$\bar{x}$	39.7	39.96	39.72	39.84	40.02	40.14	39.98	39.9	40.94	39.88	40.22	39.96	**39.5**	40.12	39.84	40.1	39.96	40.37
	$\overline{\sigma}$	0.88	1.0	0.68	0.95	0.84	0.93	0.91	1.08	1.18	0.76	0.77	1.07	0.86	0.93	1.07	0.83	0.87	4.3