Developing Issues for Ant Colony System based Approach for Scheduling Problems

Ana Madureira[1], Ivo Pereira[1] and Ajith Abraham[2,3]

[1]GECAD - Knowledge Engineering and Decision Support Group, Institute of Engineering – Polytechnic of Porto, Porto, Portugal, {amd,iaspe}@isep.ipp.pt
[2]Machine Intelligence Research Labs (MIR Labs). Scientific Network for Innovation and Research Excellence, Auburn, USA
[3]IT4Innovations - Center of Excellence, VSB-Technical University of Ostrava, Czech Republic
ajith.abraham@ieee.org

Abstract. This paper describes some developing issues for ACS based software tools to support decision making process and solve the problem of generating a sequence of jobs that minimizes the total weighted tardiness for a set of jobs to be processed in a single machine. An Ant Colony System (ACS) based algorithm performance is validated with benchmark problems available in the OR library. The obtained results were compared with the optimal (best available results in some cases) and permit to conclude about ACS efficiency and effectiveness. The ACS performance and respective statistical significance was evaluated.

Keywords: Scheduling, Optimization, Weighted Tardiness, Swarm Intelligence, Ant Colony System

1 Introduction

An important aspect of manufacturing organizations is the improvement of resource utilization. A classical approach of resource utilization optimization is through Scheduling Theory developments. As defined in Baker [1], scheduling is concerned with the problem of allocating scarce resources to activities over time. Scheduling problems are in general nontrivial and exhaustive enumeration of the scheduling solutions set is not usually efficient.

Scheduling problems are generally complex, large scale, constrained, and multi-objective in nature, and classical operational research techniques are often inadequate to effectively solving them [2]. With the advent of computation intelligence, there is a renewed interest in solving scheduling problems through Swarm Intelligence (SI) based techniques.

SI is an innovative computational and behavioral paradigm for solving distributed problems based on self-organization. SI main principles are similar to those underlying the behavior of natural systems consisting of many individuals, such as ant colonies and flocks of birds [3][4]. SI is continuously incorporating new ideas, algorithms, and principles from the engineering and basic science communities [3][4].

adfa, p. 1, 2011.

SI represents a family of approximate optimization techniques that gained a lot of popularity in the past two decades in Metaheuristics research area which is identified as a field of optimization in Computer Science and Operations Research that are related to algorithms and Theory of Computational Theory. They are among the most promising and successful optimization techniques.

In this paper a set of general guidelines for developing a software tool for solving an optimization problem and support computational study and decision making is described. An Ant Colony System (ACS) based algorithm is proposed to solve the Single Machine Weighted Tardiness Scheduling Problem (WT) and its efficiency and effectiveness is analyzed.

The remaining sections are organized as follows. In Section 2 the Weighted Tardiness problem and some approaches presented in the literature for its resolution are presented. Theoretical foundations, the biological motivation and fundamental aspects of SI paradigm with focalization on the design and implementation of an ACS algorithm and some recent applications of ACS to WT resolution are summarized in section 3. Section 4 a set of general guidelines for developing a software tool for solving an optimization problem is systematized. In Section 5 the ACS proposed approach developing for WT is described. Section 6 presents computational study and discusses obtained results. Finally, the paper presents some conclusions and puts forward some ideas for future work.

2 Problem Definition

One important scheduling problem consists in sequencing a set of jobs for processing on a single processor or machine. The study of Single Machine Scheduling Problem (SMSP) is identified to be very important for several technological and economic reasons, probably the most relevant of which is that good solutions to this problem provide a relevant support to manage and model the behavior of complex systems. In these systems it is important to understand the working of their components, and quite often the SMSP appears as an elementary component in a larger scheduling problem [1][2]. Sometimes the basic SMSP is solved independently, and then results are incorporated into the larger and more complex problem. For example, in a multistage multiple machine problems there are often a critical machine, the bottleneck, whose processing capacity is lower than the necessary. The analysis and treatment of the bottleneck as a SMSP may determine the properties of the entire schedule.

Let us consider the problem of scheduling n jobs for processing without interruption, on a single machine that can handle only one job at a time. For each job j (j=1,..,n), let pj be its processing time, dj its due date and wj the penalty incurred for each unit of time late. The processing of the first job begins at time t =1. The tardiness of a job is given by Tj=Max {tj + pj – dj, 0} where tj is the start time of job j. The objective is to find a sequence that minimizes the sum of Weighted Tardiness (WT) defined on equation 1:

$$\text{Min} \sum W_j T_j, \text{ with } T_j = Max\{t_j + p_j - d_j, 0\} \quad (1)$$

The single machine problem, here considered is characterized by the following main conditions:

- a set of n independent jobs (j=1,..,n) is available for processing at time zero and the job descriptors are known in advance;
- a machine is continuously available and is never kept idle while working is waiting;
- the set-up times for the jobs are independent of job sequence and can be included in processing times;
- jobs are processed to completion without preemption.

Under these conditions there is a one-to-one correspondence between a sequence of these n jobs and a permutation of the job indices. In this work we consider a solution such a sequence of jobs where i is the ith job position in the sequence. The total number of different solutions to the SMSP under these conditions is n!.

The SMSP for minimizing total weighted tardiness 1||WT is NP-complete [2][5]. Optimal algorithms for this problem would therefore require a computation time that grows exponentially with the problem size, presenting exponential complexity. Hence, only small sized instances of this problem can be solved in an efficient way. Several branch-and-bound procedures and dynamic programming techniques have been proposed in literature [5][6]. As indicated in [6] the simple dispatching heuristics (EDD, SWPT, COVERT or AU) do not consistently produce good quality solutions. In recent years, much attention has been dedicated to Metaheuristic that are considered to be efficient tools for solving hard combinatorial optimization problems.

Ant Colony Optimization (ACO) is probably the most successful example of artificial/engineering Metaheuristic based optimization techniques with numerous applications to real-world problems[4], and for minimization of total weighted tardiness. Merkle and Middendorf [7] describe a contribution for solving permutation problems to SMSP for Total Weighted minimization. Liao and Juan [8] propose an ACO to minimize the tardiness in a SMSP with utilization of setup times. In Yagmahan and Yenisey [9] a multi-objective scheduling problem approach based on ACO for scheduling to reduce the total scheduling cost is proposed. Anghinolfi and Paolucci [10] describe a new ACO approach to face the single machine total weighted tardiness scheduling with sequence dependent setup times problem. In Srinivasa Raghavan and Venkataramana [11] a parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization is proposed. Additionally, some relevant works could be identified, see for example [1][12][13].

3 Ant Colony Optimization

Ant Colony Optimization (ACO) has been formalized as a Metaheuristic by Dorigo and collaborators [3][14] that are inspired and mimic natural metaphors to solve

complex optimization problems . A Metaheuristic can be defined as a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide range of different optimization problems [4].

A subset family of Metaheuristics, SI is considered an innovative and creative approach to problem solving that takes inspiration from the collective intelligence of swarm of biological populations, and was discovered through simplified social behaviors model simulation of insects and other animals [4][15]. ACO algorithm is among the most promising SI inspired optimization class of optimization techniques.

Table 1. Analogy between Natural and Artificial Ants

Natural Ant Colony	***Artificial Ant Colony***
Ant	Agent
Ant Colony	Set of Ants/Iterations
Pheromone	Diversity Mechanism
Path	Solution
Evaporation	Pheromone update

The ACO algorithm takes inspiration from the foraging behavior of some ant species (Table I). These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving optimization problems. The ACO algorithm is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. For this reason description of ACO algorithms are normally accompanied through Traveling Salesman Problem (TSP) notation and illustrative examples [16-18]. This algorithm, initially proposed by Marco Dorigo in his PhD thesis [14], is a member of ACO family and it constitutes some Meta-Heuristic optimizations.

The first proposed ACO algorithm is known as Ant System [16] that was aiming to search for an optimal path in a graph. It was based on the foraging behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several algorithms have emerged, drawing on several aspects of the behavior of ants [4].

Table 2. Non-exhaustive ACO algorithms list [4]

Algorithm	***Authors***	***Year***
Ant System(AS)	Dorigo et al.	1991
Elitist AS	Dorigo et al.	1992
Ant-Q	Gambardella & Dorigo	1995
Ant Colony System	Dorigo & Gambardella	1996
MAX-MIN AS	Stutzle&Hoos	1996

Hyper-Cube AS	Blum et al.	2001

The designation ACO is a generic term that includes algorithms based on the behavior of ants. Since the early 90s, when the first ACO algorithm - Ant System - was proposed in [16], different algorithms (Table II) and successful applications have been described and a substantial theoretical results have becoming available that provides useful guidelines to researchers and practitioners in further applications of ACO based algorithms [17-19].

The general ACO algorithm is described in Table III. After initialization, the ACO iterates over three main steps: at each iteration, a number of solutions are constructed by the ants; these solutions could be then improved, optionally, through a local search, and finally the pheromone is updated through two possible events: evaporation and by increasing the pheromone levels associated with a chosen set of good solutions.

Table 3. Ant Colony Optimization Algorithm

```
Set ACO parameters.
Initialize pheromone trails
While termination criteria not met do
  Construct AntSolutions
  Apply Localsearch (optional)
  Update Pheromones
EndWhile
```

A more detailed description of the three phases can be stated as follows[4]:

- ConstructAntSolutions: A set of m artificial ants constructs solutions from elements of a finite set of available solution components $C = \{c_{ij}\}$, $i = 1, \ldots, n$, $j = 1, \ldots, |D_i|$. A solution construction starts from an empty partial solution $sp = \emptyset$. At each construction step, the partial solution sp is extended by adding a feasible solution component from the set $N(sp) \subseteq C$, which is defined as the set of components that can be added to the current partial solution sp without violating any of the constraints in Ω. The process of constructing solutions can be regarded as a walk on the construction graph $GC = (V, E)$ as stated in [4]. The selection of a solution component from N(sp) is guided by a stochastic mechanism, which is biased by the pheromone associated with each of the elements of N(sp). The rule for the stochastic choice of solution components vary across the different proposed ACO algorithms but, in all of them, it is inspired by the Goss model (experimental setup for the double bridge experiment) of the behavior of real ants assuming that at a given

moment in time m1 ants have used the first bridge and m2 the second one, the probability p1 for an ant to choose the first bridge is given by [4] :

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \tag{2}$$

where parameters k and h are to be fitted to the experimental data. Monte Carlo simulations showed a very good fit for $k \approx 20$ and $h \approx 2$.

- ApplyLocalSearch: Once solutions have been constructed, and before updating the pheromone, it is common to improve the solutions obtained by the ants through a local search. This phase, which is highly problem-specific, is optional although it is usually included in state-of-the-art ACO algorithms.
- UpdatePheromones: The aim of the global pheromone update is to increase the pheromone values associated with good or promising solutions, and to decrease those that are associated with bad ones. Usually, this is achieved by decreasing all the pheromone values through pheromone evaporation, and by increasing the pheromone levels associated with a chosen set of good solutions.

Several ACO algorithms have been proposed in the literature, which differ in some decisions characterizing the construction of solutions and update pheromone procedures [4]. Among the most successful variants we have chosen the ACS algorithm to apply to the SMSP to WT resolution.

The most interesting contribution of ACS [4][13] is the introduction of a local pheromone update and the pheromone update performed at the end of the construction process (named offline pheromone update).

ACS algorithm can be stated as follows [13]: m ants are initially positioned on n cities chosen according to some initialization rule (randomly, for example). Each ant builds a tour (feasible solution) by repeatedly applying a stochastic greedy rule (the state transition rule). While constructing its tour/path, an ant also modifies the amount of pheromone on the visited edges (cities) by applying the local updating rule. Once all ants have terminated their tour/path, the amount of pheromone on edges/cities is modified again, by global updating rule applying. Ants are guided, in building their solutions, by both heuristic information (they prefer to choose short edges), and by pheromone information (an edge with a high amount of pheromone is a very desirable choice). The pheromone updating rules are designed to give more pheromone to edges/cities which should be visited by ants.

The local pheromone update is performed by all ants after each construction step. Each ant applies it only to the last edge traversed:

$$\tau_{ij} \quad (1 \quad \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0 \tag{3}$$

where $\varphi \in [0,1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

The main goal of the local pheromone update is to introduce diversity in the search process performed by subsequent ants during an iteration by decreasing the phero-

mone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and, hence, probably to produce different solutions. This mechanism makes it less likely that several ants produce identical solutions during one iteration.

The offline pheromone update, is applied at the end of each iteration by only one ant, which can be either the iteration-best(Lib) or the best-so-far(Lbs). However, the update formula is slightly different:

$$\tau_{ij} = \begin{cases} (1-\rho)*\tau_{ij} + \rho*\Delta\tau_{ij} & \text{if } (i,j) \text{belongs to the best tour,} \\ \tau_{ij} & \text{otherwise} \end{cases} \quad (4)$$

where τij = 1/Lbest , where Lbest can be either Lib or Lbs.

Another important difference between ACS and AS is in the decision rule used by the ants during the construction process.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(S^p)} \tau_{ij}^{\alpha} * \eta_{ij}^{\beta}} & \text{if } c_{ij} \in N(S^p) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

In ACS, the so-called pseudorandom proportional rule is used by the ants during the construction process: the probability for an ant to move from city *i* to city *j* depends on a random variable *q* uniformly distributed over [0,1], and a parameter q_0; if $q \leq q_0$, $j = \text{argmax}_{c_{ij} \in N(Sp)} \{ \tau_{il} \eta_{il}^{\beta} \}$ otherwise Equation 5 is used.

Additional information about ACO based algorithms details of implementation could be found in [4][13].

4 Developing Issues for Optimization Approaches

The first aspect to be considered when identifying a decision making problem - optimization problem - and the need for specifying a tool for its resolution refers to problem modeling.

The mathematical model is built from the formulation of the problem and can be inspired by theoretical models related in the literature. This will reduce the problem to well-studied optimization models that are in general simplifications of real world problems.

Once the problem is modeled, the following stages are considered relevant for optimization approaches development (Figure 1), following a set of general guidelines systematized by Talbi[20], for solving a given optimization problem.

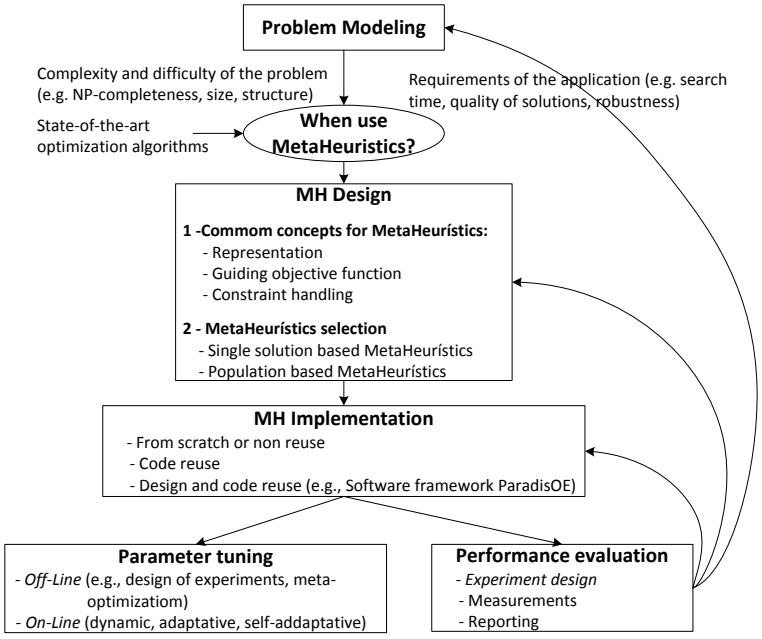

Fig. 1. Developing issues for optimization problem solving [20]

Initially should be addressed, based on the state of the art of optimization methods (exact or approximation), the question of which optimization technique is best suited to solve the problem considering the complexity and difficulty of the optimization problem (NP class, size and structure of the instances) and the requirements of the optimization problem (search time, quality of solutions and robustness).

The use of exact methods is suitable when the identified instances of the problem are solved in the time required. Meta-heuristics are a feasible alternative to obtain satisfactory solutions in circumstances where the complexity of the problem or available search time did not allow the use of exact methods.

From the moment the necessity of specifying a meta-heuristic is identified, some questions, common to all Meta-Heuristics, related to the encoding/representation of the solutions, the definition of objective function and constraints handling must be stated.

The development of software tools for the MH is a relevant task considering the variety of optimization problems identified and continuous evolution of models associated with optimization problems. The problem may be modified or require further refinements: some objectives and constraints can be inserted, deleted or changed.

It is clear the interest and the need in developing systems or automatic tools for decision support based MH. For managers it is important to select, implement and apply optimization algorithms without requiring deep knowledge on programming and optimization. For experts in optimization and software development is useful to evaluate and compare different algorithms, transform/adapt algorithms, develop new algorithms, combine and parallelize algorithms. Generally, the literature identifies three

main approaches used for the development and implementation of Metaheuristics [20]: From scratch or no reuse (considering the simplicity of MH implementation, but requires time and effort and it is error prone); Code reuse (consists on reusing free programs codes and libraries Open Source, adapting to the treated problem is often time consuming, error prone, and the coding effort using libraries remains important); and Design and code reuse(Software frameworks, its main objective is to overcome above related problems).

In general, the effective resolution of a problem requires the application of different methods of tuning parameters among others. The tuning of parameters can allow greater flexibility and robustness but requires a careful initialization. The parameters can have a major influence on the efficiency and effectiveness of the search. Becomes not obvious, a priori, the setting of parameters to use. The values for the parameters depend on the problem, instances structure and the time available to solve the problem. There are no universal values for the parameters considered for Metaheuristic based algorithms. Being widespread view that its definition must result from a careful experimental effort, towards their tuning.

Performance analysis corresponds to the last stage of development of MH. The theoretical analysis based on the worst case and average provides some insights in solving some optimization models. However, it is considered that the performance evaluation of the MH must be supported by a comprehensive set of computational tests, following these aspects/phases: definition of the test plan (test objectives, selection of input variables and instances); definition of measurement criteria (quality of solutions, robustness and computational effort) and the reporting and analysis of results (graphic display of results, interpretation of results, statistical analysis).

5 ACS proposed approach Developing for Weighted Tardiness

The scheduling problem to deal with is included into the class of combinatorial optimization problems common in industrial practice. Due to its complex nature and the resolution of such problems to optimality, in an acceptable time for the process of decision making, is virtually impossible. Thus, there is an important issue that refers to the resolution of this class of problems obtaining satisfactory quality solutions on reasonable computing time, for which there is no knowledge of the existence of efficient methods.

A software tool was developed and implemented in Java, to perform the computational study aiming to analyse and evaluate the performance of ACS, on resolution of SMSP for minimization of total weighted tardiness.

The solutions are encoded by the natural representation (string), each position corresponds to a job index and the position of the job index is the correspondent processing order. The number of positions on the string corresponds to the number of jobs (problem size).

Table 4. Ant Colony System for WT

```
Begin
  Set ACS parameters.
  Initialize pheromone trails
  While termination criteria not met do
    Construct Ant Solutions
      Each ant build a solution
      Apply LocalSearch
      Return Constructed Solution
    Apply Localsearch
    Global Update Pheromone
  EndWhile
  Return Best solution
End
```

The initial colony generation process consists in applying some mechanism generator to a starting ant solution. The initial solution is defined by the priority rule EDD rule, in which an initial solution (ant) is defined by the due dates increasing ordering, thus giving priority to tasks with small due dates.

As mentioned above, we implemented an ACS algorithm (Table IV) for WT resolution [21]. The ACS differs from the previous proposed Ant System due to three main aspects [11]: the state transition rule, the global updating rule, and the local updating rule. When applied to the SMSP for WT minimization, each ant constructs a feasible sequence by selecting an unscheduled job j to be on the ith position of the partial sequence constructed so far. This process is influenced by specific heuristic information ηij, as well as the pheromone trails τij.

5.1 Solution Construction

During the construction process the decision of adding job j to the partial sequence is made through ACS state transition rule, which is given by equation 6:

$$j \quad \begin{cases} \arg\max_{l \in \omega} \left[\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta} \right] & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{6}$$

where ω is the set of unscheduled jobs, α controls the relative importance of the pheromone trails, β determines the influence of the heuristic information, q is a random number uniformly distributed over [0,1], q_0 is a parameter ($q_0 \in [0,1]$) and S is a job selected according to the probability defined by equation 7:

$$P_{ij} \quad \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{j \in \omega} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}} & \text{if } j \subset \omega \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The state transition rule favors job selection in terms of pheromone amount versus heuristics information. The parameter q_0 provides a way to balance between exploration of new jobs and exploitation of accumulated knowledge. In other words, when ant m has to choose a job to append to the partial sequence, a random number q is generated, with $q \in [0,1]$, and if $q \leq q_0$ the best job is chosen (exploitation), otherwise a job is chosen (exploration).

The heuristic used by ants to compute the heuristic information is given by equation 8:

$$\eta_{ij} \quad \frac{1}{d_{ij}} \tag{8}$$

where d_{ij} is the total weighted tardiness for the partial sequence of jobs generated so far. The proposed solution construction procedure could be summarized on Table V.

Table 5. Solution Construction Algorithm

```
Begin
  While number of jobs not met do
    Select a new job
    Add new job to the sequence
    Perform Local Pheromone Update
  EndWhile
  Apply LocalSearch
  Return Constructed Solution
End
```

The ants construct the solution as follows: each ant starts from a randomly selected job. Then, at each construction step the ant selects a new job through a transition rule (equation 6). Each ant keeps a "memory" of its path, and the subsequent job is chosen

among the unscheduled jobs. At each construction step, an ant probabilistically chooses the next job to add to the sequence. The probabilistic rule is biased by pheromone values and heuristic information. Once a job is added to the sequence, the local pheromone is updated. This process is repeatedly applied until all jobs are scheduled. Before returning the generated solution, it still undergoes a local search algorithm. The goal is to improve the current solution by iteratively moving to a neighbor solution selected from a neighborhood of solutions according to a defined rule, which in this case, is the minimization of the total weighted tardiness. This process is quite costly in terms of computational time, but improves significantly the generated solution. However, we will address this process in more detail in the subsequent sections.

5.2 Pheromone Update Rule

The ACS pheromone update rule consists of both local and global update rule. The local pheromone update is performed by each ant after adding a new job to the partial sequence, and is given by equation 9:

$$\tau_{ij} \quad (1 \quad \rho) \cdot \tau_{ij} \mid \rho \cdot \tau_0 \tag{9}$$

where $\rho \in [0,1]$ is the pheromone decay coefficient, and τ_0 is the initial trail intensity (equation 10):

$$\tau_0 \quad \frac{1}{n \cdot WT_{EDD}} \tag{10}$$

where WT is the total weighted tardiness for a sequence generated by the EDD rule and n is the number of jobs. The main goal of the local update rule is to make the decision of appending job j on position i less desirable for the others ants so that the exploration of different sequences is favored [12].

The global pheromone update is applied at the end of each iteration by only one ant, which can be either the *iteration-best* or *best-so-far* [4]. We followed the *best-so-far* strategy, where the ant with the best solution so far contributes to the pheromone trail update, according to the equation 11:

$$\tau_{ij} \quad (1 \quad \rho) \cdot \tau_{ij} \mid \rho \cdot \Delta\tau_{ij} \tag{11}$$

where $\Delta\tau_{ij}=1/WT^*$ for all edges (i, j) belonging to the best solution found so far (WT^* is the total weighted tardiness of the best solution).

5.3 Local Search

Local Search performs a blind search, since they only accept sequential solutions which improve the value of the objective function. Essentially, consists of moving from one solution to another, in the neighborhood, according to some defined rules or local changes. The sequence of solutions trajectory depends heavily on the initial

solutions and on the neighborhood generation mechanisms adopted which defines the neighboring structure [22]. In this work, we considered adjacent pairwise interchange as a neighborhood structure to solve the problem at hand [21]. The main weakness of basic Local Search algorithms is their inability to escape from local optima.

The local search strategy described above is applied to all sequences constructed by the artificial ants. This strategy produces better results but is more costly in terms of computational time.

5.4 Illustrative example

A chemical industry produces different types of products, but can only make one at a time. The production manager has to decide on the issue of sequencing of 5 tasks on a single machine (Table VI). For each task j (j = 1, ..., 5), pj is the processing time, dj the delivery date and wj the penalty associated with task j per unit of time delay. The number of admissible solutions for this instance of problem is 5! = 120. The size of the problem is reduced to facilitate their understanding. It becomes clear that in this situation would be feasible the enumeration of all admissible solutions and select the best. However, this is no longer feasible for instances with larger dimension.

Table 6. Illustrative example

Job j	*pj*	*dj*	*wj*
1	2	5	1
2	4	7	6
3	1	11	2
4	3	9	3
5	3	8	2

Table 7. ACS Step by step

Ant i	*Solution$_i$*	*f*
1	[2 4 5 3 1]	12.0
2	[1 3 2 4 5]	13.0
3	[4 2 1 3 5]	14.0
4	[1 2 4 3 5]	10.0
5	[1 2 3 4 5]	13.0
6	[4 3 2 5 1]	20.0
7	[1 2 4 3 5]	10.0
8	[3 1 2 4 5]	13.0
9	[2 5 4 3 1]	11.0
10	[1 3 2 4 5]	13.0
11	[3 1 4 2 5]	28.0
12	[3 1 2 4 5]	13.0
13	[2 5 4 3 1]	11.0
14	[3 4 2 5 1]	20.0
15	[2 3 4 5 1]	14.0
16	[4 5 2 3 1]	26.0
17	[2 5 4 3 1]	11.0
18	[2 5 3 4 1]	14.0

19	[1 2 4 3 5]	10.0
20	[3 1 2 4 5]	13.0
21	[3 2 5 4 1]	14.0
22	[1 2 4 3 5]	10.0

23	[1 2 4 3 5]	10.0
24	[3 1 2 4 5]	13.0
25	[1 2 4 3 5]	10.0

Consider the ACS with 25 ants, a pheromone evaporation rate of 80%, α values (scale the heuristic value) and β (importance of pheromone) equal to 1. Initially the pheromone is zero.

From proposed ACS execution, we obtained the solutions presented in Table VII, where it is possible to verify the quality evolution of the paths chosen by the ants. The paths are constructed incrementally and randomly according to the values of pheromone and heuristic information, with an increased probability of choosing a solution with better heuristic value and greater pheromone. It is possible to confirm that, in the end of the process, ants choose better paths, with a propensity to choose the best so far.

Table 8. Final Pheromone Matix

Job	**1**	**2**	**3**	**4**	**5**
1	0,089	0,083	0,016	0,013	0,013
2	0,011	0,011	0,011	0,1	0,013
3	0,013	0,013	0,011	0,013	0,073
4	0,013	0,013	0,083	0,011	0,013
5	0,013	0,011	0,013	0,013	0,011

Fig. 2. ACS scheduling plan

At the end of the twenty five iterations, the path with more pheromone is [1 2 4 3 5], with a heuristic $f = 10$, which represents the best solution found by ACS (Table VIII). This path was chosen by 6 ants. The scheduling plan is illustrated in Figure 2.

6 Computational Study

A software tool was developed to perform the computational study aiming to analyse and evaluate the performance of ACS, on resolution of SMSP for minimization of total weighted tardiness. The computational tests were carried out on a PC with Intel Xeon W3565 at 3.20 GHz, with the ACS coded in Java. The ACS performance was tested on 75 benchmark instances of WT problem for different sizes n=40, n=50,

n=100, available at OR-Library [23]. We select for testing the first 25 instances for each size and not their instances where better results were obtained.

In this section a computational study is carried out in order to analyse SI based algorithms – ACS - on the resolution of benchmark problems considering quality of solutions and computational times.

Performance analysis of EC is a necessary task to perform and must be done on a fair basis. A theoretical approach is generally not sufficient to evaluate an MH based algorithm. To evaluate the performance experimentally and/or comparing in a systematized way, the following three steps must, generally, be considered [20]:

- **Experimental design**: the goals of the experiments, the selected instances, and optimization criteria have to be defined.
- **Measurement**: the measures to compute are selected. After executing the different experiments, statistical analysis is applied to the obtained results. The analysis of performance must be done based on state-of-the-art optimization algorithms dedicated to the problem.
- **Reporting**: the results must be systematized in a comprehensive way, and an analysis is carried out following the defined goals. Another important issue is related with insurance of the reproducibility of the computational experiments.

ACS based algorithm is evaluated on the resolution of WT instances and its efficiency and effectiveness will be analysed. Statistical analysis is performed in order to estimate the significance and confidence of the obtained results.

We pretend to evaluate the adequacy of Swarm Intelligence based algorithms to the WT resolution. ACS effectiveness and efficiency is evaluated on 75 benchmark instances of WT problem for different sizes (25 instances with 40, 50 and 100 jobs, respectively).

We consider that academic benchmark problems are an effective evaluation framework since they have been used by multiple authors and diverse application areas over the years, allowing an efficient comparing framework with previous work related on literature. Additionally, they permit an insight of global behavior and performance on a class of scheduling problems which are our main objective.

6.1 Parameter Tuning

The ACS algorithm has a certain number of parameters that need to be set appropriately [18]. As such, we performed a preliminary study to identify which set of values would yield better results for minimizing total weighted tardiness, for each size in consideration. The study focused on testing different values for α and β, which are used to regulate the relative influence of the pheromone and heuristic information; m, the number of ants; ρ, the pheromone evaporation rate; and q_0, the probability of choosing the next job as defined on equation 6. In Table IX we present the different tested values, as well as the standard deviation σ for obtained results.

The tests were performed by using four different sets of values, following some conclusions referred in [10]. For each set we computed n=1 simulations for each instance under analysis, as shown in Table IX. The conclusions from the obtained results could be summarized:

- α: This parameter is usually set to 1, and most of the times is not even considered;

- β: The value of β equal to 5 produced better results during most of the runtimes;
- m: The value of ants defined for each instance resulted in a good anytime performance;
- ρ: Higher values of ρ produced better results;
- q_0: Good values of q_0 tend to be close to 1. As such, we used 0.98, which proved to be a good choice.

Table 9. Parameter tuning

No. jobs	α	β	m	ρ	q_0	Σ
40	1	1	30	0.80	0.90	0.1546 ± 0.1616
	1	2	30	0.10	0.98	0.1746 ± 0.1588
	1	5	30	0.30	0.98	0.1273 ± 0.0932
	1	**5**	**30**	**0.60**	**0.98**	**0.0909 ± 0.0839**
50	1	1	50	0.80	0.90	0.1939 ± 0.2582
	1	2	50	0.10	0.98	0.1942 ± 0.1948
	1	5	50	0.30	0.98	0.1688 ± 0.1523
	1	**5**	**50**	**0.60**	**0.98**	**0.1226 ± 0.1073**
100	1	1	80	0.80	0.90	0.5253 ± 0.5659
	1	2	80	0.10	0.98	0.4176 ± 0.3776
	1	5	80	0.30	0.98	0.4285 ± 0.3868
	1	**5**	**80**	**0.60**	**0.98**	**0.3969 ± 0.3753**

The standard deviation presented in Table IX shows the variation in the results relatively to the average deviation from the optimum value. In fact, a low deviation indicates that the results tend to be very close to the average value.

After analyzing the results we concluded that the last set of values (highlighted in bold), could yield better results in the computational tests, as such the implemented ACS algorithm was parameterized with this values. The obtained results are presented and discussed in the next section.

6.2 Discussion of Results

Initially, we developed n=5 simulations for each instance under analysis. In order to analyze the obtained results, performance measures were computed: the best, the average, and worst value and the deviation error from the best obtained value to the optimal (best known available in OR-Library [23]). The relative percentage of deviation error is determined by *%error= (Best-Optimal)/Best* formula.

Table 10. Results for 5 runs

No. jobs	*Average time (s)*	σ	*No. of optimal*

40	15	0.0366 ± 0.0391	8
50	50	0.0552 ± 0.0489	4
100	1086	0.1872 ± 0.1714	0

The obtained solutions values by the proposed ACS algorithm are presented in Figure 3 for n=40, Figure 4 for n=50 and Figure 5 for n=100, also in Table X is presented the average computational time, the standard deviation, and the number of optimum values.

The following figures presents the results obtained from the performed tests. Each figure shows the average deviation, as well as the deviation of each intances from the optimum value. Through the figures is also possible to identify the instances were the optimum value was reached, i.e., the ones were the deviation value is zero. Some of this instances appear to have reached the optimum value, when in fact there is a slight deviation that is not noticeable in the figures. For that reason we present in Table X the total number of optima for n=40, n=50, and n=100.

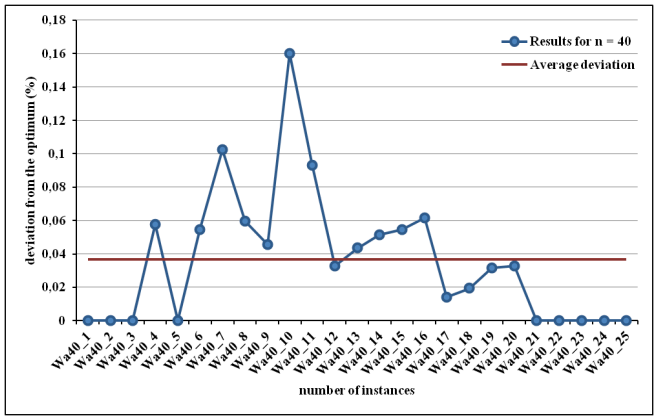

Fig. 3. Results obtained with 5 runs for n=40

In general most of the instances were solved in relatively short computational time. For n=40 the average time was 15 seconds, as for n=50 the average time was 50 seconds. Only the instances of n=100 took more time to be solved (~18 minutes).

As an extension to our study, we tested ACS again, and see if the solution already obtain could improve. Therefore, the number of simulations was increased to n=20 and the exactly same set of parameters were used. The results obtained are presented in Table XI supported by Figure 6 for n=40, Figure 7 for n=50 and Figure 8 for n=100.

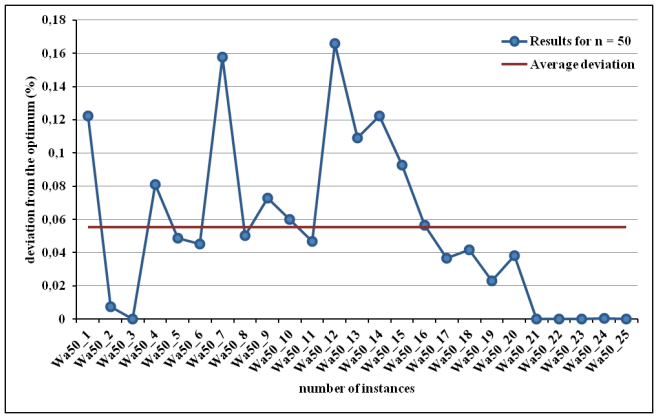

Fig. 4. Results obtained with 5 runs for n=50

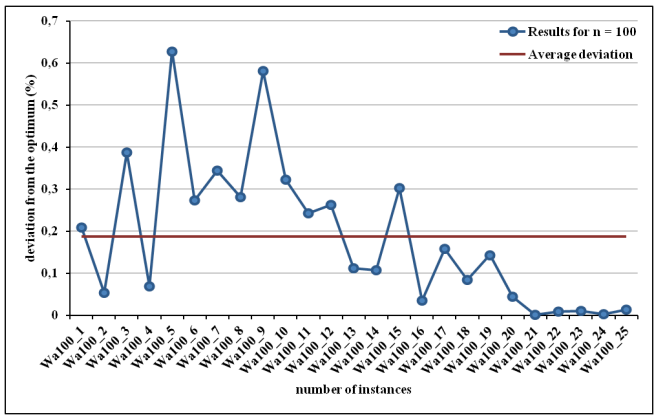

Fig. 5. Results obtained with 5 runs for n=100

Table 11. Results for 20 runs

No. jobs	*Average time (s)*	*σ*	*No. of optimal*
40	14	0.0172±0.0209	11
50	49	0.0374±0.0328	5
100	1072	0.1524±0.1523	0

The average computational time remained the same, and the results improved significantly. As shown in Table XI, and presented in Figure 6, Figure 7, and Figure 8 the number of optima increased. Moreover, in most of the instances the deviation from the optimum value decreased. These results support the assumption that the outcome of running the ACS more times would be to find the best known solutions on all instances.

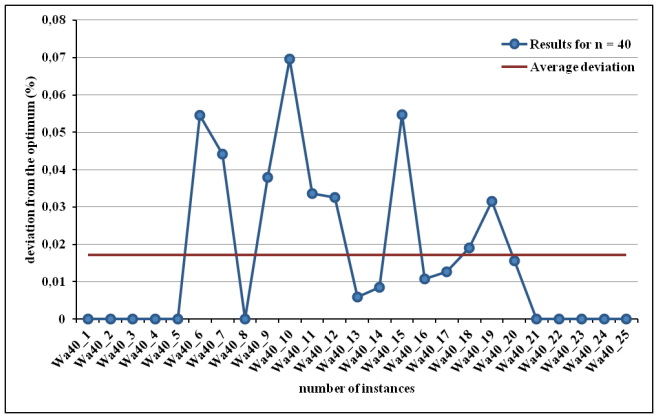

Fig. 6. Results obtained with 20 runs for n=40

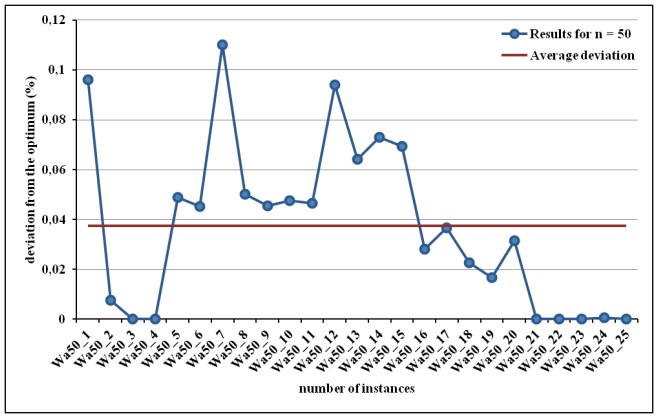

Fig. 7. Results obtained with 20 runs for n=50

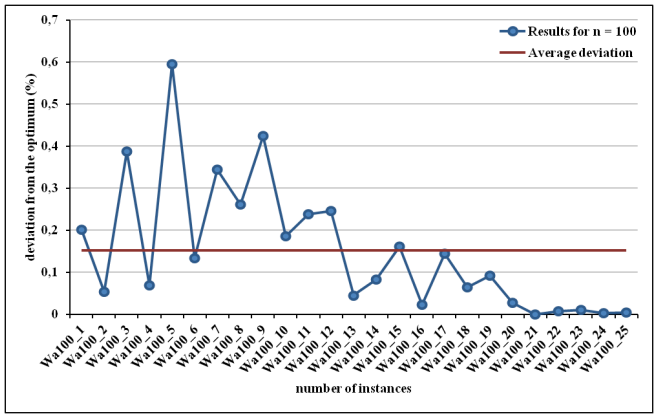

Fig. 8. Results obtained with 20 runs for n=100

6.3 Significance analysis

The boxplot from Figure 9 allows the analysis of confidence interval, making its synthesis and compared mean values by ACS and optimal solutions in terms of minimization for *Weighted Tardiness* (WT) with 40 jobs. From the boxplot analysis we can conclude that ACS has been effective on the resolution of WT considering that its obtained mean values were similar to the optimal solutions (in some instances the best known). It is not clear, from the graph analysis, if exist significant difference of the performance of ACS related with optimal solutions.

Additionally, some statistical sampling were retrieved to summarize important features from the observed instances From the analysis of statistical sampling summary based on WT minimization (Table XII), it is possible to conclude that does not exist statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 40 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents similar variability.

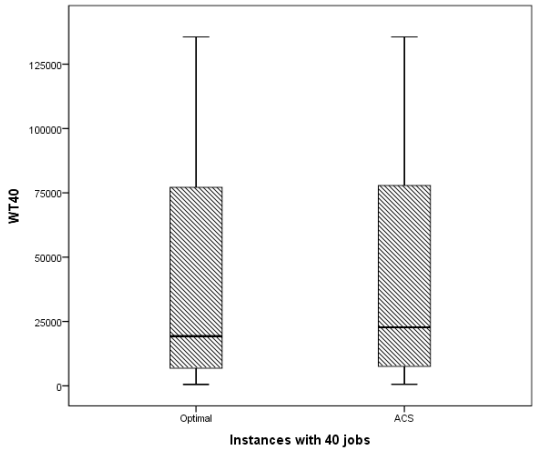

Fig. 9. Boxplot for WT with 40 jobs

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is not statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is no statistically significant difference between performance of ACS (M = 48607.76, SD=46928.024), and optimal solutions related on literature (M=43494.40, SD= 44745.846) $t(24) = -1$, $p=.327$. These results could suggest that ACS have been effective on WT resolution for 40 jobs, considering that does not exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 40 jobs.

Table 12. Statistical Sampling Summary based on WT40

	Optimal	ACS
Mean	43494,40	48607,76
Median	19312,00	26914,00
Variance	2002190770,083	2202239475,023
Std. Deviation	44745,846	46928,024
Interquartile Range	70854	71289
Skewness	,791	,634
Kurtosis	-,737	-1,073

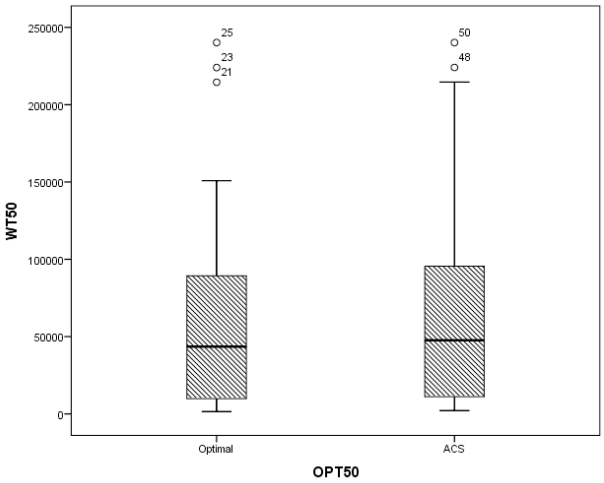

Fig. 10. Boxplot for WT with 50 jobs

Table 13. Statistical Sampling Summary based on WIT50

	Optimal	ACS
Mean	67110,40	70139,44
Median	43504,00	47627,00
Variance	5296763112,167	5234325587,590
Std. Deviation	72778,864	72348,639
Interquartile Range	87856	91550
Skewness	1,266	1,179
Kurtosis	,724	,564

From boxplot, Figure 10, analysis we can conclude that there are outliers or extreme values and the analysis of location, dispersion and asymmetry of data, making its synthesis by ACS, and optimal solutions, that ACS has been effective on the resolution of WT considering that its obtained mean values were similar to the optimal solutions (in some instances the best known). It is not clear, from the graph analysis, the existence of significant difference of the performance of ACS related with optimal solutions.

From the analysis of statistical sampling summary based on WT minimization (Table XIII), it is possible to conclude that exist some statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 50 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents different variability.

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is significant.

A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M=70139.44, SD=72348.64), and optimal solutions related on literature (M= 67110.40, SD= 72778,86) $t(24) = -4{,}652$, $p<0.001$. These results suggest that ACS performance has decreased on WT resolution for 50 jobs when compared with 40 jobs considering that exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 50 jobs.

The boxplot from Figure 11 depicts the values of obtained solutions by ACS on the WT resolution of 25 instances in analysis with 100 jobs. From the boxplot analysis it is possible to conclude about difference of the performance of ACS related with optimal solutions.

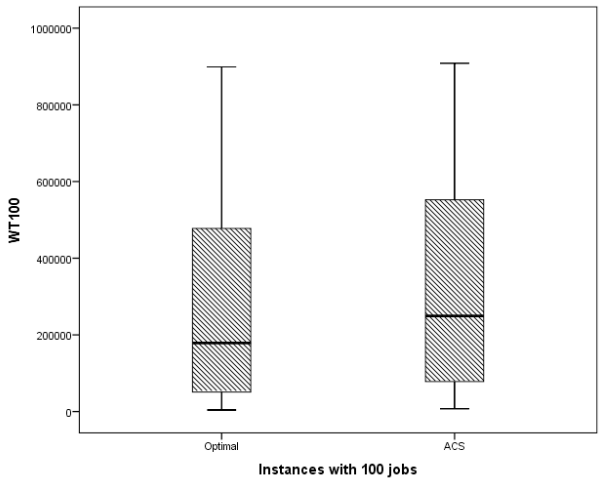

Fig. 11. Boxplot for WT with 100 jobs

Table 14. Statistical Sampling Summary based on WT100

	Optimal	ACS
Mean	268357,52	302118,04
Median	178840	249518
Variance	69219509079,2	70906890332,5
Std. Deviation	263096	266283,5
Interquartile Range	462725	480355
Skewness	,809	,594
Kurtosis	-,337	-,711

From the analysis of statistical sampling summary based on WT minimization (Table XIV), it is not possible to conclude about statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 100 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude about differences on performance by ACS when comparing with optimal solutions.

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M=302118,04, SD=266283,5), and optimal solutions related on literature (M= 268357,5, SD= 263096,01) $t(24) = -5{,}959$, $p<0.001$. These results suggest that exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 100 jobs.

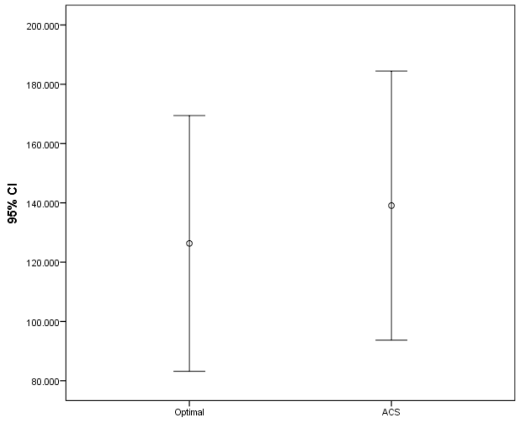

Fig. 12. Error Bar of the WT for all instances

Table 15. Statostical Sampling Summary based on WT

	Optimal	ACS
Mean	126320,77	139080,95
Median	53208,00	75267,00
Variance	35134539958,2	38921816643,2
Std. Deviation	187442,1	197286,1
Interquartile Range	142301	139647
Skewness	2,241	2,045
Kurtosis	4,912	3,764

Previous computational and statistical tests indicate that ACS presents effectiveness on the resolution of WT for 40 jobs, but its performance decreases for 50 and 100 jobs, which can indicate that parameter tuning, must be improved in order to increase ACS performance for those class of instances. Following, in order to evaluate the overall performance of ACS on the resolution of 75 instances of WT scheduling problem with 40, 50 and 100 jobs, additional statistical analysis will be conducted to compare the overall performance of ACS when compared with respective optimal solutions.

From the analysis of statistical sampling summary based on WT minimization (Table XV), it is possible to conclude that exist statistic evidence of the difference on the ACS performance of WT resolution. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents similar variability.

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M= 139080.95, SD=197286.13), and optimal solutions related on literature (M=126320.77, SD= 187442.1) $t(74) = -5.000$, $p<0.001$. These results suggest that exist statistical evidence that its performance is significantly different than optimal solutions for the same WT instances with 100 jobs.

From the obtained results we can conclude that proposed ACS and its parameterization is adequate for 40 jobs instances considering that obtained results indicate that the difference in performance of ACS and optimal solution is not statistically significant. For instances with greater dimensions parameter tuning must be increased, considering a degradation of computational time.

7 Conclusions and Future Work

In this paper, we described some guidelines for ACS based software developing tools to study the effectiveness and efficiency of ACS in the optimization of total

weighted tardiness for SMSP. More than developing algorithms with unquestionable practice utility, the main purpose of this paper was to illustrate, through more simple scheduling problems, the potential effectiveness and efficiency of using MH approaches, with special emphasis on SI based techniques for scheduling problem solving. The obtained results show that proposed ACS algorithm was effective for the instances studied, being possible to find good solutions in short time, i.e., a few CPU seconds. As future work, we will extend our ACS algorithm to the job-shop scheduling problem, where the jobs are distributed across different machines.

Acknowledgments
This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e a Tecnologia" under the project: FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011 and PTDC/EME-GIN/109956/2009. . This work is partially supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

References

1. K. R. Baker, "Introduction to Scheduling," vol. 32, Brussels, 1992.
2. K. R. Baker, D. Trietsch,"Optimization methods for the single machine probelm," in Principles of Sequencing and Scheduling, 1st ed. New York: Wiley, 2009, pp. 34-56.
3. M. Dorigo, "Swarm Intelligence," New York: Springer, 2007 (4)
4. M. Dorigo, M. Birattari, and T. Stützle, "Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique," IEEE Computational Intelligence Magazine, 2006.
5. E. L. Lawer, "A pseudopolinomial algorithm for sequencing Jobs to Minimize Total Tardiness," Annals of Discrete Mathematics, 1997, pp. 331-342.
6. R. G. Reynolds, "An Introduction to Cultural Algorithms," in Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scienfific Publishing, 1994, pp. 131–139.
7. D. Merkle, and M. Middendorf, "On solving permutation scheduling problems with ant colony optimization," International Journal of Systems Science, vol. 36, no. 5, 2005, pp. 255-266.
8. C. Liao, and H. Juan, "An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups," Computers & Operations Research , vol. 34, 2007, pp. 1899-1909.
9. B. Yagmahan, and M. M. Yenisey, "Ant colony optimization for multi-objective flow shop scheduling problem," Computers & Industrial Engineering , vol. 54, 2008, pp. 411-420.
10. D. Anghinolfi, M. Paolucci, "A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem," International Journal of Operations Research, vol. 5, no. 1, 2008, pp. 1-17.
11. N. R. Srinivasa Raghavan, M. Venkataramana, "Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization," The International Journal of Advanced Manufacturing Technology, vol. 41, no. 9-10, 2009, pp. 986-996.

12. M. den Besten, T. Stützle, and M. Dorigo, “Ant colony optimization for the total weighted tardiness problem,” in Proc. PPSN-VI, ser. LNCS, M. Schoenauer et al., Eds., vol. 1917, Springer Verlag, 2000, pp. 611-620.
13. M. Dorigo, and L. M. Gambardella, “Ant Colony System: A cooperative learning approach to the traveling salesman problem,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, 1997.
14. M. Dorigo, Optimization, Learning and Natural Algorithms, PhDThesis, Politecnico di Milano, Italy, (in Italian), 1992.
15. Marco Dorigo and Thomas Stützle, Ant Colony Optimization, MIT Press, 2004.
16. M. Dorigo, V. Maniezzo, and A. Colorni, “Positive feedback as a search strategy,” Dipartimentodi Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.
17. M. Dorigo and L.M. Gambardella, “Ant colonies for the traveling salesman problem,” BioSystems, vol. 43, no. 2, pp. 73–81, 1997.
18. L.M. Gambardella and M. Dorigo, “Solving symmetric and asymmetric TSPs by ant colonies,” in Proc. 1996 IEEE International Conference on Evolutionary Computation (ICEC’96), T. Baeck et al., Eds. IEEE Press, Piscataway, NJ, pp. 622–627, 1996.
19. T. Stützle *et al.*, Parameter Adaptation in Ant Colony Optimization, IRIDIA, Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2010-002, Jan. 2010.
20. El-Ghazali Talbi, Metaheuristics – From Design to Implementation, Wiley, 2009.
21. A. Madureira, I. Pereira and Diamantino Falcão, Ant Colony System Based Approach to Single Machine Scheduling Problems — Weighted Tardiness Scheduling Problem, International Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC'12), 5-9 de November, México, 2012.
22. M. Pirlot, “General Local Search Method,” European Journal of Operational Research, vol. 92, 1996, pp. 493-522.
23. OR-Library - http://people.brunel.ac.uk/~mastjjb/jeb/info.html.