2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Cooperation Mechanism for Distributed Resource Scheduling Through Artificial Bee Colony Based Self-Organized Scheduling System

Ana Madureira
GECAD Research Group-School of Engineering
Polytechnic Institute of Porto
Porto, Portugal
amd@isep.ipp.pt

Bruno Cunha
GECAD Research Group-School of Engineering
Polytechnic Institute of Porto
Porto, Portugal
bmaca@isep.ipp..pt

Ivo Pereira
GECAD Research Group-School of Engineering
Polytechnic Institute of Porto
Porto, Portugal
iaspe@isep.ipp.pt

***Abstract*— In this paper a Cooperation Mechanism for Distributed Scheduling based on Bees based Computing is proposed. Where multiple self-interested agents can reach agreement over the exchange of operations on cooperative resources. Agents must collaborate to improve their local solutions and the global schedule. The proposed cooperation mechanism is able to analyze the scheduling plan generated by the Resource Agents and refine it by idle times reducing taking advantage from cooperative and the self-organized behavior of Artificial Bee Colony technique. The computational study allows concluding about statistical evidence that the cooperation mechanism influences significantly the overall system performance.**

Keywords— Self-Organization; Cooperation; Distributed Resource Scheduling; Distributed Resource Scheduling; Artificial Bee Colony.

I. INTRODUCTION

Cooperation in an economic and social context can be stated as the process of two or more entities or organizations working or acting throughout common objectives [1]. The current financial/economic scenario and the increasing competitiveness of markets could be identified as the main factors responsible for the promotion of research and development of new approaches for solving scheduling problems in manufacturing systems in the perspective of supporting human decisions. Making the right decision at the right instant can effectively strengthen the performance, agility and position of organizations over competitors.

In this paper, we start with the belief that by understanding the solutions that nature employs in its quotidian, we can use this acquired knowledge to solve real world problems in direct analogy with Biomimetic Computing. Using insights from biological systems to specify intelligent computational systems has focused on optimization techniques inspired by biological observations, including Neural Networks, Genetic Algorithms, and more recently Swarm based Algorithms [2]. Artificial Bee Colony (ABC) is one of the most recently introduced swarm-based algorithms. ABC simulates the intelligent foraging behavior of a honeybee colony. In this paper, the modified version of ABC is used. Biologically inspired systems have been gaining importance and it is clear that many other ideas can be developed by taking advantage of the examples that nature offers. Some social systems encountered in nature can present an intelligent collective behavior although they are composed by simple individuals with limited capabilities. The intelligent solutions to problems can naturally emerge from the self-organization and indirect communication of the individuals. These systems provide important techniques that can be used in the development of distributed artificial intelligent systems [2][3].

Social issues in the organization of Multi-Agent Systems (MAS) have been increasingly relevant over the last decades as a platform to define the basis of agent interactions in open and dynamic environments [3]. A Cooperation Mechanism for Distributed Resource Scheduling (CMDS) for Artificial Bee Colony based Self-Organized Scheduling System is proposed. This CMDS is embedded in the AutoDynAgents scheduling system architecture [5], which consists on a system with a community of agents that model a real manufacturing system subject to perturbations. Agents must cooperate to improve their local solutions and the global schedule. The proposed CMDS is able to analyze the scheduling plan generated by the Resource Agents and refine it by idle times reducing taking advantages from cooperative and the self-organized behavior of Artificial Bee Colony technique.

The remaining sections are organized as follows. Section II compiles issues and contributions for cooperation in Multi-Agent Systems. Theoretical foundations, the biological motivation and fundamental aspects of Bee based Computing with focalization on the design and implementation of an ABC algorithm and some recent applications for scheduling resolution are summarized in section III. In Section IV the proposed Cooperation Mechanism is described. Section V presents a computational study and discusses the obtained results. Finally, the paper presents some conclusions and puts forward some ideas for future work.

II. COOPERATION IN MULTI-AGENT SYSTEMS

Multi-Agent Systems highlight the joint behaviors of agents with some degree of autonomy and the complexities arising from their interactions [6]. Due to the interactions among the agents, MAS complexity can increase rapidly with the number of agents or their behavioral intricacy.

In a multi-agent environment there is more than one agent, interacting with each other and constraints such that agents may not, at any given time, know everything about the environment that other agents know, including the internal state of the other agents themselves [6]. Due to these interactions and constraints, it is necessary to incorporate coordination and cooperative behaviors between agents, leading to cooperative MAS, which are the ones that cooperatively solve tasks or maximize utility [6].

Natural evolution is based on the law of survival, in which only the fittest survive. As a way to counteract that natural rule, groups of individuals of the same species (or even different species) tend to cooperatively join their behaviors, in order to fight for survival [7]. Cooperation exists at multiple levels of activity in a wide range of populations. People pursue their own goals through communication and cooperation with other people or machines. Animals interact (with limited language), cooperate with each other, and form communities. Particles interact with each other and compose different types of material and phases of matter [8]. These are examples of how cooperation is ubiquitous in the real world and can be observed at different organizations, ranging from microorganisms and animal groups to human societies [9].

Solving the problem of how cooperation emerges among self-interested entities is a challenging issue that has motivated scientists and practitioners from different disciplines including psychology, sociology, economics, and computer science, amongst others. The emergence of cooperation is often studied in the context of social dilemmas, in which selfish individuals must decide between a socially reciprocal cooperative behavior and a self-interested behavior of defection to pursue their own short-term benefits. Social dilemmas often arise in many situations in MAS, e.g., file sharing in peer-to-peer systems, load balancing/packet routing in wireless sensor networks and bandwidth allocation/frequency detection in telecommunication systems [10]. As Kraus [8] mentioned, "cooperation (not merely coordination) may improve the performance of the individual agents or the overall behavior of the system they form". This paper proposed a cooperation mechanism embedded in a Multi-Agent Scheduling System.

In the literature, it is possible to find several approaches of cooperation in MAS. Adler et al. [11] explored the use of cooperative MAS to improve dynamic routing and traffic management. On the supply-side, real-time control over the transportation network is accomplished through an agent-based distributed hierarchy of system operators. Allocation of network capacity and distribution of traffic advisories are performed by agents that act on behalf of information service providers. The cooperation between agents seeks a more efficient route allocation across time and space. Results from simulation experiments suggest that this cooperation can achieve increased network performance and driver satisfaction.

De Gennaro and Jadbabaie [12] proposed a decentralized cooperative controller for a group of mobile agents. The control design was based on the navigation function formalism. The aim of the group control law was to generate a pattern in a given workspace while avoiding obstacles and collisions. The desired goal was specified in terms of distances among the agents. Authors showed that it is always possible to design a control law as the gradient of a suitably-defined navigation function whose minimum corresponds to the desired configuration. Srinivasan and Choy [13] described a cooperative multi-agent approach employing multiple hybrid agents to provide effective signal control for real-time traffic management. The detailed MAS use cooperative behaviors to improve individual agents' learning process and adaptability. Each agent cooperates with other agents within its cooperative zone, the size of which changes dynamically according to the changing needs of the agent. The performance of the proposed cooperative MAS was tested on a complex traffic network and compared against two other approaches. The results demonstrate the efficacy of the cooperative multi-agent approach in dealing with the approximated version of an infinite horizon dynamic problem. Choi et al. [14] presented an algorithm and analysis of distributed learning and cooperative control for MAS so that a global goal of the overall system can be achieved by locally acting agents. The proposed algorithm is executed by each agent independently, in order to estimate an unknown field of interest from noisy measurements and to coordinate multiple agents in a distributed manner to discover peaks of the unknown field. The proposed algorithm is based on a recursive spatial estimation of an unknown field. Authors showed that the closed-loop dynamics of the proposed MAS can be transformed into a form of a stochastic approximation algorithm and proved its convergence.

III. BEES BASED COMPUTING

Optimization algorithms inspired from the collective behavior of biological populations that can be observed in nature such as ants, bees, fish, and birds are associated with the paradigm of Swarm Intelligence and have been referred as a creative approach to optimization problem solving. ACS, PSO and ABC algorithms are among the most promising Swarm Intelligence (SI) optimization techniques for scheduling resolution [2]. The Bees Algorithm is a relatively new population-based search algorithm, initially proposed in 2005 by Pham et al. [15] and Karaboga et al. [16] independently. The algorithm mimics the food foraging behavior of swarms of honey bees. In its basic version, the algorithm performs a kind of neighborhood search combined with random search and can be used for optimization problems. In 2005, Pham proposed a Bees Algorithm in a technical report [15] inspired in the foraging behavior of honey bees to find food sources. At the same time Karaboga [16] proposed an Artificial Bee Colony (ABC) algorithm that drew a similar inspiration in the foraging behavior of the bees.

Real bees are social insects living in an organized group called hive. In a beehive, the group has some specific tasks performed by specialized individuals. The goal of this organization is to maximize the amount of nectar in the colony by getting the utmost from food sources. Natural bees are known for the production of honey and for the adaptation to changes in the environment in a collective intelligent way. Bees have photographic memory, proper navigation systems, use the nectar for energy and use the pollen as protein for the creation of their offspring [4][17].

Generally, a bee colony contains a reproductive female (queen), a few thousand males (drones), and many thousands of sterile females (workers). After mating with multiple drones,

the queen produces many young bees ushering in a new generation [4][17]. The queen has a life expectancy of 3 to 5 years, is the mother of all members of the colony, and its main task is mating with the drones, a reproductive operation that is known as mating flight. The fertilized eggs become females (workers and future queen) and unfertilized eggs become males (drones). Drones are the males in the hive, have a life expectancy of around 90 days and do not live longer than six months. Its main task is to mate with the queen and die after having done so successfully. Worker bees are females without reproductive capacity, representing most of the bees in the hive, and usually live between 4-9 months. In the first half of their life, they are responsible for many tasks in the hive, from maintenance to defense. In the second half of their life, the main task of these bees is looking for food, and initially they make short flights in order to learn the location of the hive and the topology of the environment. Searching for food is the most important task of a bee colony [4][17]. This process begins with the search for food sources by exploiting a group of flowers, with the aim of extracting nectar. The value of a food source depends on several factors such as proximity to the hive, the concentration of energy, and ease of food extraction. For the sake of simplicity, the profitability of a food source can be represented by a single numerical value which represents the quality [4]. After unloading the nectar, the explorer bee who found the food source becomes "used" and performs a special move, a sort of dance, in order to share information about the source of food with spectator bees, causing them to explore this food source as well. Besides the value of the food source, this shared information also involves the direction and distance from the hive [17][19]. The sharing of information between the bees is the most important part in the formation of collective knowledge. So there are three different types of dances that bees can do: round dance, waggle dance, and tremble dance [17][18]. If the distance of the food source in relation to the hive is less than 100 meters, the bees perform a round dance; otherwise they perform a waggle dance. While the round dance gives no information about the direction of the food source, the waggle dance gives information about the direction to the sun, the distance to the hive and the quality of the food source. Finally, the tremble dance is used when the bee detects delays when discharging its nectar.

Spectator bees watch various dances from different bees and decide which food source they want to use, and there is a higher probability of choosing more profitable sources. Once they choose and locate the food source, spectator bees become worker bees by exploiting this source to extract its nectar. When the food source is exhausted, employed bees become explorers [4][17].

A. Artificial Bee Colony algorithm

In this work the Modified ABC (MABC) algorithm proposed by Karaboga and Akay [4] will be used. The Modified ABC algorithm (Table I) has three main phases, corresponding to three types of specialized bees, Employed, Onlooker and Scout, that represent a minimal model of the real swarm intelligent forage selection [4][17]. Employed bees are in the same number as food sources (solutions) and are responsible for exploring one and only one food source at the time and give information to other bees. When an employed bee leaves its food source it becomes a scout bee. Onlooker bees wait in the hive until employed bees inform them of the whereabouts of a good food source. Scouts bees search the environment trying to find a new food source depending on an internal motivation or external clues or randomly. Half of the hive is composed by employed bees and the other half by onlooker bees. A solution represents a food source that is measured by the nectar amount corresponding to the quality of the solution.

TABLE I. MODIFIED ABC ALGORITHM

```
Input: ABC Parameters and scheduling data problem
Output: Best solution

1:  Begin
2:     Initialization of Bee Population
3:     Cycle = 1
4:     While cycle != Maximum Cycle Number
5:        Employed Bees Phase
6:        Calculate Probabilities for Onlookers
7:        Onlooker Bees Phase
8:        Scout Bees Phase
9:      Keep the best solution achieved so far
10:     Increment Cycle
11:   EndWhile
12: End
```

In the initialization phase, the algorithm randomly generates *sn/2* initial solutions, were *sn* is the size of the population, which will be the food field for the employed bees. Each x_i ($i=1, 2, sn/2$) is a dimensional vector D. Values between the limits of the parameterization are assigned to the solution and a $failure_i$ value is also added to analyze when this solution i must be abandoned. After validating the population, the algorithm repeats a specified number of cycles of employed, onlooker and scout bees' phases.

1) *Employed bees phase*

An employed bee performs a change in their position of food source based on equation (1) and evaluates the nectar amount in the new position/solution v_{ij}[4]:

$$v_{ij} = \begin{cases} x_{ij} + \emptyset(x_{ij} - x_{kj}), \ if \ R_j < MR \\ x_{ij}, \ otherwise \end{cases} \quad (1)$$

where $k \in \{1,2,\ldots,sn\}$ is a randomly chosen index that has to be different from i, and $\emptyset$ is an uniformly distributed random real number in the range of $[-1,1]$. R_j is an uniformly distributed random real number in the range $[0,1]$ and MR is a control parameter of the ABC algorithm in the range of $[0,1]$ which controls the number of parameters to be modified. Then the algorithm selects the solution by the following rules:

- Two realizable solutions – selects the one with the best amount of nectar (fitness) value;
- One solution realizable and one unrealizable – select the realizable;
- Two unrealizable solutions - select the one with the smaller degradation factor.

Finished the search, the employed bees share the information with the onlooker bees and the solutions are selected based on a probability of the value of fitness or

violation of the solutions depending if they are realizable or not.

2) *Onlooker bees phase*

Onlooker bees select their own food source based on a probabilistic rate, according to the amount of nectar on the solution. The algorithm uses the equation (1) to create a new food source, validating and adjusting the new solution according to the parameterization.

3) *Scout bees phase*

After the above steps, all food sources that are not being explored anymore are abandoned. The employed bees that left the food source get a new position resulting from the scout bees search.

For additional information and full explanation of MABC, see [4].

B. Parameters Configuration

The parameters for an SI based algorithm can have a major influence on the efficiency and effectiveness of the search. The setting of parameters to use is not obvious at first. The parameters' values depend on the problem, instances' structure and the available time to solve the problem. In general, there are no universal values for the parameters considered for SI based algorithms. Being widely accepted that its definition must result from a careful experimental effort, towards their tuning.

The MABC algorithm has a certain number of parameters that need to be set appropriately [4][16]. 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. After some preliminary parameter tuning the parameters of MABC algorithm were defined, considering identical computational effort, to allow a better comparison in efficiency (computational time) and effectiveness (quality of solution). The selected parameterization is as follows: A population size of 50 (100), a maximum failure of 1000(2000) and the cycle number being 3000 (4500) for small/biggest instances. The solutions are encoded by a 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). The initial bee colony generation process consists in applying some mechanism generator to a starting individual. The initial solution is defined by the priority rule Earliest Due Date (EDD), in which an initial solution (bee) is defined by the due dates increasing ordering, thus giving priority to tasks with small due dates.

In order to evaluate the performance of the proposed cooperation mechanism using the MABC for local optimization, it has a certain number of specific parameters that need to be set appropriately. An extensive computational effort has been made for parameter tuning of the SI techniques in order to ensure identical computational effort. In Table II the different parameterization values for MABC are summarized. As performance evaluations' criteria were considered the *makespan* [20], also referred in the literature as C_{max} (as the total execution time, that is, when all the jobs have finished processing), and system utilization (as the proportion of the available time that the resources are operating).

IV. COOPERATION MECHANISM FOR DISTRIBUTED SCHEDULING

The proposed Cooperation Mechanism for Distributed Scheduling (CMDS) aims to provide the system with cooperative intelligence in order to analyze the scheduling plan generated by the Resource Agents and improve it by reducing idle times. This procedure has the main objective of minimizing completion times (*makespan* or C_{max}) and maximizing of system utilization [20].

The proposed Cooperation Mechanism is incorporated in a Collaborative Dynamic Scheduling architecture-AutoDynAgents scheduling system- that consists in MAS in which a community of agents models a real manufacturing system subject to perturbations and imponderables [5][21-22]. Agents must be able to learn and manage their internal behavior and their relationships with other autonomic agents, by cooperating in accordance with business policies defined by managers and operational managers.

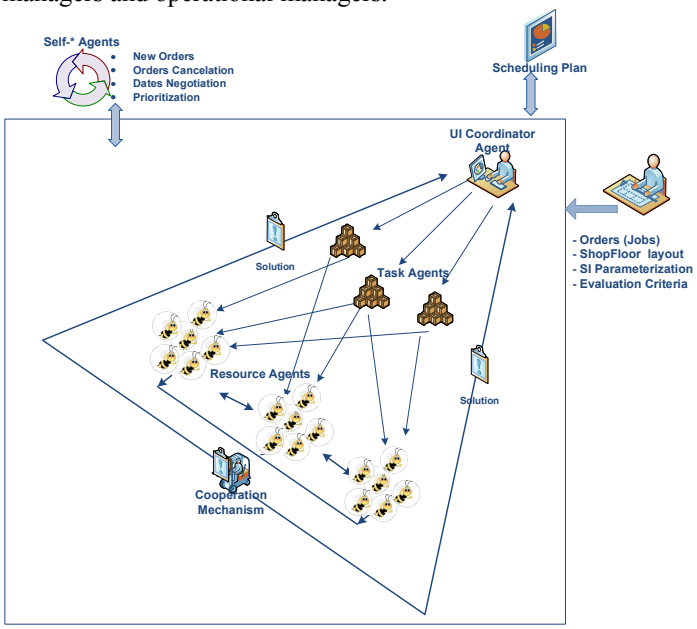

Fig. 1. AutoDynAgents scheduling system model with MABC

The Cooperation Mechanism is used when a global solution has been attained by the scheduling module. The cooperation mechanism pretends to improve the final solution. As a principal concept for this module we pretend to minimize the machines' (resources) idle times, by swapping operations, keeping all restrictions imposed by the problem. A backup copy of the best scheduling solution is maintained, which is updated every time the cooperation module reaches a new best solution. This behavior guarantees that even when the final solution is not improved by the cooperation mechanism, the system has always a possible solution that will never be degraded. When the cooperation module reaches a better solution, it updates the backup copy, when a worse solution is obtained the previous one is restored.

The scheduling module is based on a Resource Oriented decomposition approach where the scheduling problem is decomposed into a series of Single Machine Scheduling Problems (SMSP)[21-22]. The Resource Agents (which have a MABC method associated) obtain local solutions and later cooperate in order to overcome inter-agent constraints and achieve a better global schedule. This considers a specific kind of social interaction, known as competitive problem solving, where the group of agents work together to achieve a good solution for the problem.

The AutoDynAgents architecture model (Fig. 1) is based on four different types of agents: User Interface Coordinator Agent, Task Agents, Resource Agents (RA) and self-* Agents [5]. The model envisages representing the main components of a dynamic scheduling in a manufacturing environment and it is designed to simulate resources and tasks in a scheduling decision making process involving cooperation. Each RA is responsible for scheduling the operations that require processing in the machine supervised by the agent, and must be able to find an optimal or near optimal local solution through MABC for SMSP and cooperate with other agents (Fig. 1). Additionally, RA must also deal with dynamism (new jobs arriving, cancelled jobs, changing jobs attributes, etc), change/adapt the parameters of the basic algorithm according to the current situation and cooperate with other agents. SMSP aims at sequencing a set of jobs on a single machine [20].

The User Interface Coordinator agent (UI Coordinator agent) is responsible for coordinating and integrating the single solutions obtained by each Resource Agent solution in order to obtain a global schedule for the original scheduling problem, and by coordinating the process of Cooperation Mechanism. Self-* agents are responsible for guaranteeing agility and adaptation.

A. Algorithm

Table II illustrates the notation used for the Cooperation Mechanism algorithm.

TABLE II. NOTATION

Symbol	Description
ρ	Current plan
ρ_b	Backup plan
c	Cost of ρ
c_b	Cost of ρ_b
v	Set of idle times
f_{ini}	Flag to know if it is the first time the algorithm is being executed
f_{end}	Flag to know if the algorithm is finished
f_{coop}	Flag to know when Cooperation should be started from scratch

The system is designed to loop until the cooperation cannot create improvements to the plan. The whole cycle is detailed in Fig. 2. The system takes the scheduling plan generated by the resource agents and integrates it into a global schedule by UI Coordinator. The global solution is then saved (ρ_b) in order to be compared later with the best solution (so that the improvement can be measured). The Cooperation Mechanism cycle begins with the original plan.

Cooperation mechanism algorithm (Table III) consists, at the lowest level, of operations swapping. It starts by calculating the biggest idle time in the plan and finding which operations are related to it (`getIdleTimes`). Next, it calculates which of the two possible actions can be applied to the plan in order to obtain a better solution. The first one will investigate if it is possible to swap the current operation with its following, in the same machine (`testOperationsSwapSucceeding`). The second one tries to switch the operation that precedes the one that is being analyzed in the task with its predecessor (`testOperationsSwapPreceding`). The second hypothesis is only applied if the first one is not possible. If neither of them are feasible, the mechanism starts over, but using instead the larger idle time after the one that was used (therefore the need to use the clause "`If fcoop`" to ensure that the idle times set is not repopulated). The cooperation is finished when there are no idle times left to test.

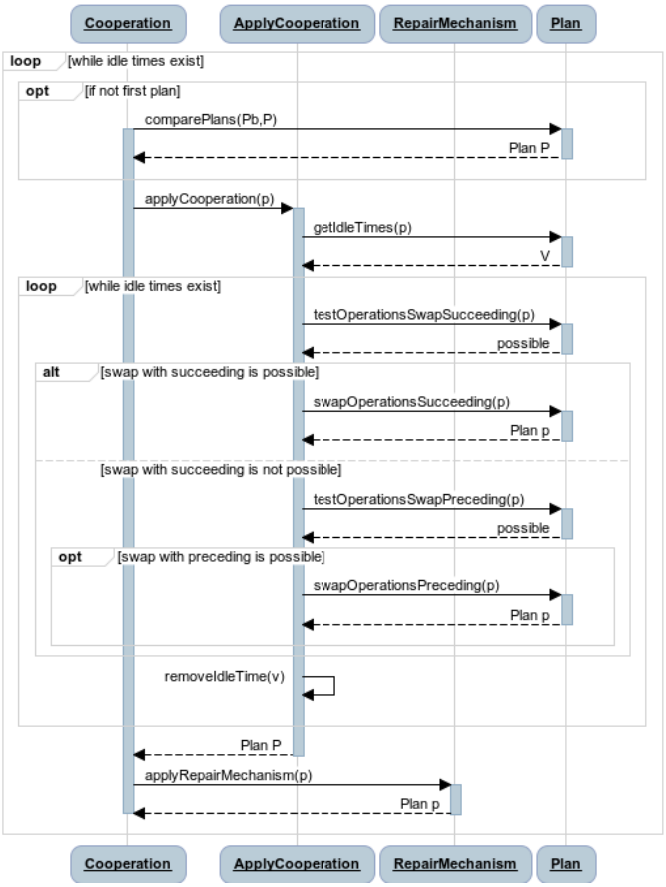

Fig. 2. Sequence Diagram

After the swapping operations (if possible), the system sends back the current plan, which needs to go through the UI Coordinator in order to ensure all restrictions imposed by the problem. The full loop is then restarted with the new plan, but now its cost is compared (`comparePlans`) against the backup that has been stored (ρ_b) previously, and the best overall solution is the one that is kept. With this behavior it is guaranteed that not only is the solution never downgraded, but also that there exists a plan in case that improvements are not generated. If the backup plan (ρ_b) is better than the new plan, the cooperation starts over, but now it skips the first idle time (otherwise it would generate the same plan).

If the generated plan has improved the backup, it is kept and the cooperation will start from scratch (no restriction on which idle time to use). The system only stops when all of the idle times from the best plan are analyzed and the related operations are used as a "starting point to swap" – until there is no more room for improvement by switching any operations. The synchronization between the modules is maintained using flags (f_{ini}, f_{end}, f_{coop}). The overall cycle - backup the current best plan, apply cooperation, repair and compare it with the backup - keeps is repeated until an assurance that the best possible solution (using the original plan) was obtained. With this mechanism, the system is able to gradually improve and refine the overall solution by cooperation, which will permit getting an improved scheduling plan. At the end the new local schedules will be sending to Resource Agents in order to restore current behavior.

TABLE III. COOPERATION MECHANISM ALGORITHM

Input: ρ // Repaired plan
Output: ρ // Plan after cooperation
Begin
  If f_{coop} **then** //if first time running the cooperation for ρ
    $v \leftarrow getIdleTimes(\rho)$ // sorted in descending order
  EndIf
  If *sizeOf*(v) > 0 ***Then*** // If idle times (still) exist
    If *testOperationsSwapSucceeding*(ρ) ***Then***
      $\rho \leftarrow$ *swapOperationsSucceeding* (ρ)
    Else If *testOperationsSwapPreceding* (ρ) ***Then***
      $\rho \leftarrow$ *swapOperationsPreceding*(ρ)
    EndIf
    $v \leftarrow removeIdleTime(v)$
  Else
    $f_{end} \leftarrow true$
  EndIf
return ρ
End

B. *Illustrative Example*

Using the following scheduling problem, with 3 tasks processed on 3 machines (Table IV):

TABLE IV. SCHEDULING PROBLEM EXAMPLE

Task	Operation	Machine	t_{proc}	t_{start}
	T1.1	M1	5	0
T1	T1.2	M2	10	5
	T1.3	M3	4	15
	T2.1	M1	4	0
T2	T2.2	M2	5	4
	T2.3	M3	8	9
	T3.1	M1	5	0
T3	T3.2	M2	3	5
	T3.3	M3	7	8

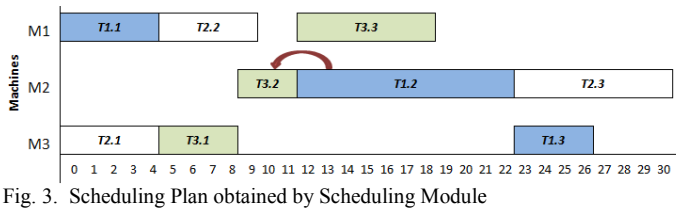

Fig. 3. Scheduling Plan obtained by Scheduling Module

The minimization of the *makespan* will be used as optimization criteria. All Resource agents will use that behavior to generate the initial plan. For this example, we take the plan described in Fig. 3 as the result from the system's scheduling module. This solution is stored as the best solution encountered up to this point. The cooperation mechanism complements the schedule mechanism, being used as a method to improve the final solution. By the end of the cooperation, the plan will (if possible) be more continuous, reducing machine idle times and, consequently, reducing completion times of all tasks.

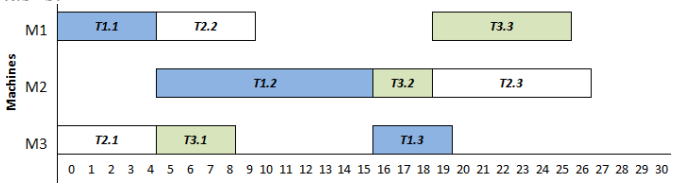

Fig. 4. Plan after the first cooperation iteration

TABLE V. IDLE TIMES WITHOUT THE COOPERATION MECHANISM

Machine	Number of stops	Idle Times
M1	1	2
M2	1	9
M3	1	13
Total	**3**	**24**

TABLE VI. IDLE TIMES WITH THE COOPERATION MECHANISM

Machine	Number of stops	Idle Times
M1	1	8
M2	1	5
M3	1	6
Total	**3**	**19**

On the first iteration, the idle time on M3 is analyzed, and it corresponds to operation ***t1.3*** – which is the last operation on the machine, so the first cooperation test fails. Nevertheless, the second test is successful, as it is possible to swap the preceding operation in the task (***t1.2***) with its preceding in the machine (***t3.2***). This allows ***t1.2*** to start earlier, affecting the succeeding operations and improving the overall solution by 4 time units (Fig. 4). The cooperation mechanism will stop because no additional improvement is possible.

When analyzing the initial solution (Fig. 3), it had the idle times presented in Table V. And after going through the cooperation mechanism, the idle times presented in the plan are shown in Table VI. At the end, the overall solution was improved. There is the same number of stops, the task completion times and the machine idle times are reduced permitting an increase in the system's efficiency through the maximization of system usage. Even with a small example like this it is possible to realize the contribution of this mechanism. In problems that have an extensive number of machines and each one has a diverse set of operations, the cooperation is almost mandatory to increase system effectiveness.

V. COMPUTATIONAL STUDY

A software tool was developed to support the computational study aiming to analyze and evaluate the scheduling system's performance and the influence of the proposed cooperation mechanism, on the minimization of the *makespan* (C_{max}) and the maximization of system usage or

utilization. The computational tests were performed on an Intel® Core™ 2 Quad Q6600 @ 2.40 GHz processor, 4 GB of RAM memory, a 250 GB 7200 rpm disc, and Windows 7 64-bit as operative system. Considering that academic benchmark problems are an effective evaluation framework, since multiple authors and diverse application areas have used them over the years, the performance was tested on 36 benchmark instances of Job-Shop Scheduling Problem (JSSP) of different sizes [23]. The instances analyzed were selected based on their dimension (number of jobs). Therefore, for this study we used different problem instances from Fisher and Thompson [24], Lawrence [25], Adams, Balas and Zawack [26], Storer, Wu and Vaccari [27] and Yamada and Nakano [28].

In this paper, the performance of the proposed Cooperation Mechanism CMDS was tested based on benchmark instances of JSSP of different sizes, considering that they give an insight of global behavior and performance on a class of scheduling problems, which is our main objective.

A. Minimization of makespan

In order to evaluate the performance of the proposed Cooperation Mechanism, the system was executed with a MABC. For each instance under analysis, n=5 simulations were computed. The Cooperation Mechanism for Distributed Scheduling (CMDS) obtained improvement on 89% of the instances.

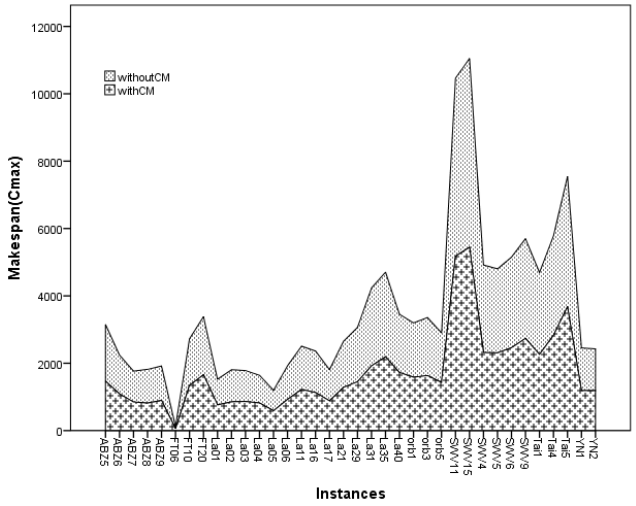

Fig. 5. Cooperation mechanism influence for C_{max}

The graph from Fig. 5 permits to conclude about the advantage and influence of CMDS when analyzing *makespan* minimization criteria.

The boxplot from Fig. 6 allows the analysis of location, dispersion and asymmetry of data, with and without the application of the cooperation mechanism. From its analysis it is possible to conclude that there are outliers or extreme values. It is also possible to observe the influence of the cooperation mechanism in the system's performance, in terms of minimization of the *makespan* (C_{max}), when comparing median of differences between the values obtained with and without the application of the cooperation mechanism. However, from the graph analysis (Fig. 6), the influence of the cooperation mechanisms in the overall system performance is not clear.

To evaluate the significance of the Cooperation Mechanism's influence on the performance of the scheduling system, the Related Samples Wilcoxon Signed Ranks Test has been used. From inferential statistical analysis it is possible to conclude with statistical evidence that the CMDS influence the performance of the system with α=5% of significance level. The null hypothesis H0, that considers CMDS does not influence the performance of system, was rejected with 95% of confidence level ($p=0<\alpha$).

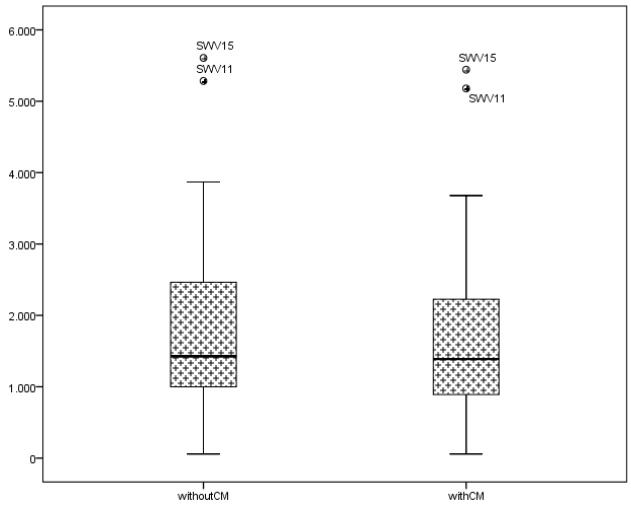

Fig. 6. Cooperation mechanism performance significance on minimization of *makespan*

B. Maximization of system utilization

The graph, from Fig.7 permits to conclude about the advantage and influence of the CMDS on the system performance, when analyzing the maximization of system utilization criteria.

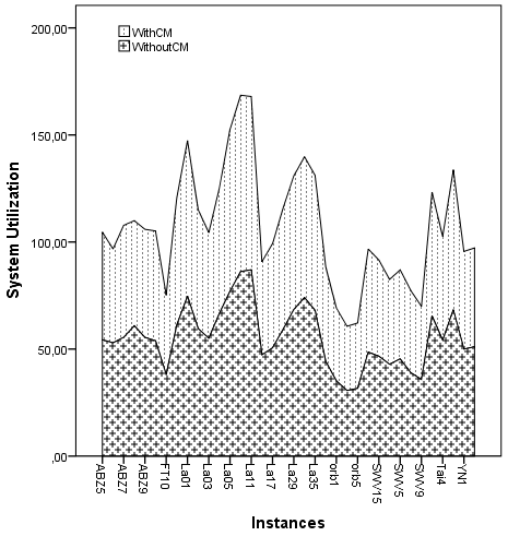

Fig. 7. Cooperation mechanism Influence for system utilization

The boxplot from Fig. 8 permits to conclude about the influence of the Cooperation Mechanism, in terms of maximization of system utilization in the resolution of the analyzed instances.

The Related Samples Wilcoxon Signed Ranks Test was used for inferential statistical analysis. It is possible to

conclude with statistical evidence that the Cooperation Mechanism influence the performance of the system with α=5% of significance level when considering system utilization. The null hypothesis H0, that consider the Cooperation Mechanism does not influence significantly the performance of the system on maximization of system utilization, was rejected with 95% of confidence level ($p=0<\alpha$).

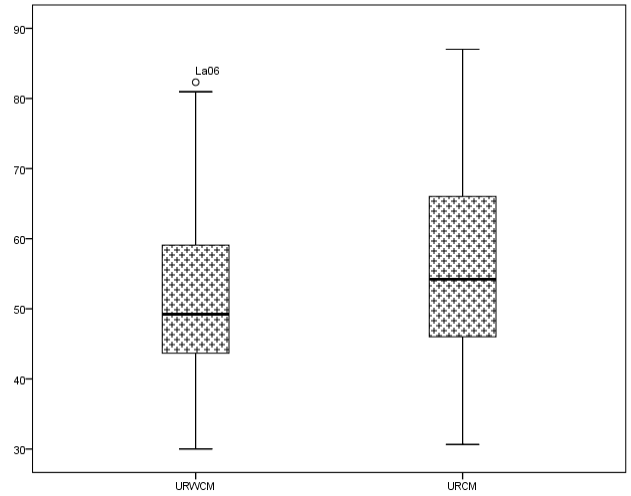

Fig. 8. Cooperation mechanism performance significance on maximization of system utilization

VI. CONCLUSION

A Cooperation Mechanism for Distributed Scheduling based on Bees based Computing is described, where multiple self-interested agents can reach agreement over the exchange of operations on competitive resources. Resource Agents collaborate to improve their local solution and the global schedule. The proposed cooperation mechanism is able to analyze the scheduling plan generated by the Resource Agents and integrated by the Coordinator Agent, and refine it by idle times reduction.

Experimental analysis was performed in order to validate the influence of the proposed cooperation mechanism in the scheduling system's performance. From the obtained results it was possible to conclude with statistical evidence that the cooperation mechanism influences significantly the overall system's performance, even when analyzing *makespan* minimization and system utilization maximization.

Future work includes the refinement of the Cooperation Mechanism, and its validation under dynamic environments subject to several random perturbations.

ACKNOWLEDGMENT

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.

REFERENCES

[1] A.Diekmann, S.Lindenberg,: Sociological aspects of cooperation, Intern. Encyclop.of the Social Sciences and behavioral sciences, Elsevier, 2001.

[2] M. Dorigo, Swarm Intelligence, New York, Springer, 2007.

[3] M. Luck, P. McBurney, O. Shehory, S. Willmott, Agent Technology: Computing as Interaction, A Roadmap for Agent-Based Computing, AgentLink III, 2005.

[4] D. Karaboga, B. Akay, A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, Applied Soft Computing, vol. 11, pp. 3021-3031, 2011.

[5] A. Madureira, N. Sousa, I. Pereira, Self-organization for Scheduling in Agile Manufacturin,10th IEEE International Conference on Cybernetic Intelligent Systems (IEEE CIS 2011), London, UK, 2011.

[6] L. Panait and S. Luke, Cooperative Multi-Agent Learning: The State of the Art, Autonomous Agents and Multi-Agent Systems 11(3), 2005.

[7] R. Axelrod, The Evolution of Cooperation, Basic Books, 2006.

[8] S. Kraus, Negotiation and cooperation in multi-agent environments, Artificial Intelligence 94(1-2), pp. 79-97, 1997.

[9] L. Hofmann, N. Chakraborty, and K. Sycara, The evolution of cooperation in self-interested agent societies: a critical study, in Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, pp. 685–692, 2011.

[10] N. Salazar, J. Rodriguez-Aguilar, J. Arcos, A. Peleteiro, and J. Burguillo-Rial, Emerging cooperation on complex networks, in The 10th International Conference on Autonomous Agents and Multiagent Systems, pp. 669–676, 2011.

[11] J.L. Adler, G. Satapathy, V. Manikonda, B. Bowles, and V.J. Blue, A multi-agent approach to cooperative traffic management and route guidance, Transportation Research Part B: Methodological 39(4), pp. 297-318, 2005.

[12] M.C. De Gennaro and A. Jadbabaie, Formation control for a cooperative multi-agent system using decentralized navigation functions, American Control Conference, 2006.

[13] D. Srinivasan and M.C. Choy, Cooperative multi-agent system for coordinated traffic signal control, in Proceedings of Intelligent Transport Systems 153(1), pp. 41-50, 2006.

[14] J. Choi J., S. Oh, and R. Horowitz,Distributed learning and cooperative control for multi-agent systems,Automatica45(12), pp. 2802-2814, 2009.

[15] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S.Rahim, M. Zaidi, The Bees Algorithm, Manufacturing Engineering Centre, Cardiff University, United Kingdom, 2005.

[16] D. Karaboga, An Idea based on Honey Bee Swarm for Numerical Optimization, Technical report -TR06,Erciyes University, Engineering Faculty, Computer Engineering Department 2005.

[17] D. Karaboga and B. Akay., A survey: algorithms simulating bee swarm intelligence. Artif. Intell. Rev. 31, 1-4, 61-85,2009.

[18] T.D. Seeley, S. Camazine, J. Sneyd, Collective decision-making in honey bees: how colonies choose among nectar sources, Behavioral Ecology and Sociobiology, Springer-Verlag, 28:277-290, 1991.

[19] T.D. Seeley, W.F. Towne, Tactics of dance choice in honey bees: Do foragers compare dances? Behav. Ecol. Sociobiol. 30, 59-69, 1992.

[20] K. Baker and D.Trietsch, Principles of Sequencing and Scheduling, Wiley, 2007.

[21] A. Madureira, I. Pereira, A. Abraham. Towards Scheduling Optimization through Artificial Bee Colony Approach, IEEE 5th World Congress on Nature and Biologically Inspired Computing, USA, 2013.

[22] A. Madureira, C. Ramos, S.C. Silva, A Coordination Mechanism for Real World Scheduling Problems Using Genetic Algorithms, IEEE World Congress on Computational Intelligence - (CEC'2002), Honolulu - Hawai (EUA), 2002.

[23] OR-Library–http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

[24] H. Fisher and G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, pp.225-251, 1963.

[25] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques, Graduate School of Industrial Adminis., Carnegie-Mellon University, Pennsylvania, 1984.

[26] J. Adams, E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34, pp. 391-401, 1988.

[27] R.H. Storer, S.D. Wu, R. Vaccari, New search spaces for sequencing instances with application to job shop,Management, Science38,pp. 1495-1509, 1992.

[28] T. Yamada, R. Nakano, A genetic algorithm applicable to large-scale job-shop instances, R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, pp. 281-290, 1992.