

# Introducing Bar Systems: A Class of Swarm Intelligence Optimization Algorithms

Esteve del Acebo and Josep Lluís de la Rosa<sup>1</sup>

**Abstract.** We present Bar Systems: a family of very simple algorithms for different classes of complex optimization problems in static and dynamic environments by means of reactive multi agent systems. Bar Systems are in the same line as other Swarm Intelligence algorithms; they are loosely inspired in the behavior a staff of bartenders can show while serving drinks to a crowd of customers in a bar or pub. We will see how Bar Systems can be applied to CONTS, a NP-hard scheduling problem, and how they achieve much better results than other greedy algorithms in the "nearest neighbor" style. We will also prove this framework to be general enough to be applied to other interesting optimization problems like generalized versions of flexible Open-shop, Job-shop and Flow-shop problems.

## 1 INTRODUCTION

The origin of the term Swarm Intelligence, which so vast amount of attention has drawn in the last years amongst the Artificial Intelligence, Artificial Life and Distributed Problem Solving communities is to be found in the observation of social insect colonies. A commonly accepted and used definition of it is: "the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge". Doubtless, the paradigm of a Swarm Intelligence system is an ant colony. In it, individual ants' behavior is controlled by a small set of very simple rules, but their interactions (also very simple) with the environment allow them to solve complex problems (such as finding the shortest path from one point to another one). Ant colonies (and we could say the same about human beings) are intelligent systems with great problem solving capabilities, formed by a quantity of relatively independent and very simple subsystems which do not show individual intelligence. It is the "many dummies make a smart" phenomenon of emergent intelligence.

Swarm Intelligence problem solving techniques present several advantages over more traditional ones. On one hand, they are cheap, simple and robust; on the other hand, they provide a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model. Over the last years they have been used in the resolution of a very heterogeneous class of problems: Two of the most successful Swarm Intelligence techniques currently in use are Ant Colony Optimization [5] and Particle Swarm Optimization [8]. Ant Colony Optimization techniques, also known as Ant Systems, are based in ants' foraging behavior, and have been applied to problems ranging from determination of minimal paths in TSP-like problems to network traffic

rerouting in busy telecommunications systems. Particle Swarm Optimization techniques, inspired in the way a flock of birds or a school of fish moves, are general global minimization techniques which deal with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Other Swarm Intelligence applications include collective robotics, vehicle navigation, planetary mapping, streamlining of assembly lines in factories, coordinated robotic transport, banking data analysis and much more. The interested reader can find a lot of useful references about self-organization and Swarm Intelligence theory and applications in [1], [7], [9], [2], [6], and [3].

The class of systems we present in this paper, Bar Systems, are reactive multi agent systems whose behavior is loosely inspired in that of a staff of bartenders, and can be enclosed in the broader class of Swarm Intelligence systems.

The paper is organized as follows: in the next section we will present and formalize the concept of Bar System, in section 3 we present the CONTS problem, a NP-hard scheduling problem for multi agent systems which will serve us to test the performance of Bar Systems. In sections 4 and 5 we will see how to solve the CONTS using a Bar System and we will comment the results. Finally, in section 6, we will draw some conclusions and we will discuss some directions toward which future work can be directed.

## 2 BAR SYSTEMS

Anybody who has tried to get served a pint in a bar crowded with customers will have had more than enough time to wonder with boredom about the method used by waiters, if there is any, to decide which customer to pay attention to at each time. Sometimes there is not much point, to be served before, in having been waiting for long or in yelling at the waiter. Details like the bar area where the customer is, his/her sex, whether the waiter knows him/her or whether the waiter likes the customer's face determine to a high extent the way in which orders are served.

Let us examine the situation from the bartenders' point of view: a heap of customers are ordering drinks at once, new ones arrive all the time, and the bartenders have to do all they can to serve them. Of course, they cannot do it in a random way; they have to try to maximize some kind of utility function which will typically take into account aspects such as average serving time, average serving cost or average customer/boss satisfaction. They will have to pay attention, then, to facts such as that some of them can prepare certain drinks more quickly or better than others, that the order in which the drinks are served influences the time or the total cost of serving them, and that also moving from one place in the bar to another costs time. All of this without forgetting, on one hand, that the order in which orders

<sup>1</sup> Agents Research Lab. Institut d'Informàtica i Aplicacions Universitat de Girona, Spain, email: acebo@ima.udg.es peplluis@eia.udg.es

take place has to be respected as much as possible and, on the other hand, that they have to try to favor the best customers by giving them special preferential attention and keeping them waiting for a shorter time.

The problem is not at all trivial, (actually we will see that it can be proved to be NP-hard), bartenders have to act in a highly dynamic, asynchronous and time-critical environment, and no obvious greedy strategy (such as serving first the best customer, serving first the nearest customer or serving first the customer who has arrived first) gives good results. Nevertheless, a staff of good bartenders usually can manage to serve a lot of customers in such a way that the vast majority of them were, more or less, satisfied. The way they accomplish the task seems to have little to do with any global planning or explicit coordination mechanisms but, arguably, with trying to maximize, every time they choose a customer to serve, some local utility function which takes into account aspects like the importance of the customer, the cost for the waiter of serving her/him and the time that he/she has been waiting for service.

In the next section, we will try to give a general formalization of this type of problem solving behaviors, which we call Bar Systems.

## 2.1 Definition

We will define a Bar System as a quadruple  $(E, T, A, F)$  where:

1.  $E$  is a (physical or virtual) environment. The state of the environment at each moment is determined by a set of state variables  $V_E$ . One of those variables is usually the time. We define  $S$  as the set of all possible states of the environment  $E$ , that is, the set of all the possible simultaneous instantiations of the set of state variables  $V_E$ .
2.  $T = \{t_1, t_2, \dots, t_M\}$  is a set of tasks to be accomplished by the agents within the environment  $E$ . Each task  $t_i$  has associated:
  - $pre(t_i)$ . A set of preconditions over  $V_E$  which determine whether the task  $t_i$  can be done.
  - $imp(t_i)$ . A nonnegative real value which reflects the importance of the task  $t_i$ .
  - $urg(t_i)$ . A function of  $V_E$  which represents the urgency of task  $t_i$  in the current state of the environment  $E$ . It will be usually a nondecreasing function of time.
3.  $A = \{a_1, a_2, \dots, a_N\}$  is a set of agents situated into the environment  $E$ . Each agent  $a_i$  can have different problem-dependent properties (i.e. weight, speed, location, response time, maximum load...). For each agent  $a_i$  and each task  $t_j$ ,  $cost(a_i, t_j)$  reflects the cost for agent  $a_i$  to execute the task  $t_j$  in the current state of the environment. This cost can be divided in two parts: on one hand, the cost for  $a_i$  to make the environment fulfill the preconditions of task  $t_i$  (this can include the cost of stop doing his current task) and, on the other hand, the cost for  $a_i$  to actually execute  $t_j$ . If an agent  $a_i$  is unable to adapt the environment to the preconditions of the task  $t_j$  or if it is unable to carry the task out by itself then we define  $cost(a_i, t_j)$  as infinite.
4.  $F : S \times A \times T \rightarrow \mathcal{R}$  is the function which reflects the degree to which agents are "attracted" by tasks. Given a state  $s$  of the environment, an agent  $a_i$  and a task  $t_j$   $F(s, a_i, t_j)$  must be defined in a way such that it increases with  $imp(t_j)$  and  $urg(t_j)$  and it decreases with  $cost(a_i, t_j)$ .

In Bar Systems, agents operate concurrently into the environment in an asynchronous manner, eliminating, thus, the typical operation

cycles of other SI systems (Ant Systems, Particle Swarm Optimization Systems, Cellular Automata...). The general individual behavior of agents is given by the following algorithm:

```

REPEAT
  Find the most attractive free task MAFT;
  IF the agent is currently doing MAFT THEN
    keep doing it;
  ELSE
    Stop doing the current task, if any;
    IF pre(MAFT) hold THEN start doing MAFT
    ELSE do some action to fulfill pre(MAFT);
  ENDIF
ENDIF
UNTIL no tasks left;

```

The crucial step in the algorithm above is the choice of the task which the agent has to execute for the next time step. In its simplest form, it can consist in choosing the one which maximizes the attraction function  $F$ . We will see in the next sections that it can also involve some kind of negotiation between agents and even some kind of local planning.

It is worth to stress the fact that the algorithm allows the agents to respond in real time to changes in the environment like the appearance of new urgent tasks or the temporal impossibility of fulfilling the set of preconditions of a given task.

### 2.1.1 Inter-agent communication

Even if Bar Systems don't require from the agents any communicative skills, they are indispensable in order for the system to attain the coordinated and self organized behavior typical of Swarm Intelligence Systems. We can identify three main purposes to which communication can serve in order to increase Bar Systems problem solving capabilities:

- *Conflict resolution and negotiation.* The way we defined Bar Systems makes unavoidable the occurrence of conflicting situations in which two or more agents choose the same task to carry out. Lack of communication will lead to a waste of resources because of several agents trying to fulfill the preconditions of the same task, even if only one of them will finally carry it out. In such situations it would be convenient to have some kind of negotiation method which can be as simple as "the first one which saw it goes for it". In the case study, in section 3, we will discuss a couple of more elaborated negotiation strategies.
- *Perception augmentation.* In the case that agents have limited perception capabilities (we refer to capability to perceive the tasks), communication can allow an agent to transmit to the others information about pending tasks they are not aware of. Let's suppose we want to do some kind of exploratory task in a vast terrain where points of interest must be identified and explored by means of a Bar System. It would be useful that agents had the ability to share information about the points of interest which they have located during their exploratory activity, this way agents would have access to information about the location of points of interest which lie beyond their perceptual capabilities.
- *Learning.* The attraction function  $f$  defined in section 2.1 does not need to be fixed in advance. Agents can learn it through their own activity and their communicative interactions with other agents. For example, an agent can find out that a certain kind of task has a high cost and communicate this fact to the other agents. Not only

that, agents can even learn from other agents the way of carrying out new tasks.

On the other side, It is worth to differentiate two main classes of inter-agent communicative processes:

- *Direct.* Agents establish direct communication with each other via some channel and following some kind of consensuated protocol.
- *Indirect.* Agents communicate with each other through their actions, which cause changes in the environment. In the Bar Systems framework, it can be seen as agents generating “communicative tasks” which, when carried out by other agents, increase the information they possess (about the environment, the task set . . . ). This is the case of Ant Systems, which, from this point of view, can be seen as a particular case of Bar Systems.

### 2.1.2 Local planning

Although there is nothing like global planning in the way a set of bartenders work, they have tricks that allow them to spare time and effort. For example if two customers are asking for a pint and they are close enough to each other in the bar, the bartender will usually serve them at once. In a similar way, a taxi driver who is going to pick up a passenger will surely take advantage of the opportunity if he finds in his way a new passenger and he can transport him without deviating too much from his original route. The inclusion of this sort of very simple, problem-dependent, local planning techniques in the choice of the tasks is not difficult and can be done through different methods ranging from local search to the use of expert rules.

## 3 THE CONTS PROBLEM

A class of problems frequently found in “real life” involves some kind of scheduling in the transport of goods or people from one place to another. The problem which we present as a framework for the study of Bar Systems applicability and efficiency is inspired in the problem which has to be solved by a group of loading robots in a commercial harbor. The task of these robots is to transport the containers from their storage place to the docks where the corresponding ships have to be loaded. Of course, this transport has to be done in such a way that the containers arrive in time to be loaded and with the lowest possible cost. Next we state a formalization (and simplification) of the problem, which we will call CONTS. Afterward we are going to study its complexity and we will see how we can use a Bar System to solve it efficiently.

### 3.1 Definition of the problem

Let  $C = \{c_1, c_2, \dots, c_n\}$  be a set of containers, let  $L = \{l_1, l_2, \dots, l_m\}$  be a set of loading robots and let  $P = \{(x, y) \in \{0..MaxX\} \times \{0..MaxY\}\}$  be a set of positions. Each container  $c_i$  has the following associated properties:

- $p(c_i) \in P$ . The position where the container lies.
- $dest(c_i) \in P$ . The position to which the container has to be carried to.
- $weight(c_i) \in \mathbb{R}^+$ . The weight of the container.
- $dline(c_i) \in \mathbb{R}^+$ . The latest instant of time in which the container can arrive to the dock in order to be loaded in time into the ship.

In order not to complicate the problem too much, we will assume that all the containers have the same importance. There are also several properties associated to each loading robot  $l_i$ :

- $p(l_i) \in P$ . The place where the robot is at each instant.
- $maxload(l_i) \in \mathbb{R}^+$ . The maximum weight the robot is able to carry.
- $maxdist(l_i) \in \mathbb{R}^+$ . The distance beyond which the robot can’t “hear”. It allows us to model the perceptual limitations of the robot.
- $speed(l_i) \in \mathbb{R}^+$ . The speed at which the agent can move.

Robots can perform different actions, they can move toward any position, load (if container and robot are in the same position) containers which weigh less or the same as its *maxload* value and download containers.

The problem consists in finding, if it exists, a sequence of actions that allows the robots, departing from their initial positions, to transport every container to its destination point, in such a way that no container arrives after its deadline. In order to simplify the problem, we will assume that the robots always move at the same speed, that uploading and downloading operations are instantaneous and that robots can only carry one container at a time.

### 3.2 Complexity of the CONTS problem

Of course, before trying to solve the problem we have to get an idea of its complexity. Using an heuristic method might not make much sense if there was some exact method of polynomial complexity. On the contrary, if the problem was very complex, using heuristic methods which gave approximate solutions, like Bar Systems, would be justified. The fact is that the problem is not at all trivial. The associated state space is enormous (it is not only necessary to take into account which containers each robot will move and in which order; the solution of some instances of the problem implies moving some containers to a different position from the one of delivery and leave them there to return to take them later) and it is also extremely sensitive to initial conditions, as most of NP-hard problems usually are. In [4] an in-depth study of the problem can be found with a proof of it to be at least as complex as a NP-hard problem. In general terms, the proof reduces the Traveling Salesman Problem (TSP) to CONTS by showing that every instance of the TSP problem is equivalent to an instance of CONTS where there is a single robot and all the containers have the same deadline and have to be delivered in the same position where they lie. We have also programmed an exhaustive search method that finds optimal solutions, but, as expected, it can only deal with extremely simple instances of the problem.

## 4 A BAR SYSTEM FOR SOLVING THE CONTS PROBLEM

Once the option of solving the problem in an exact way in the general case has been discarded, we now look at the possibility of using an heuristic method like a Bar System. The idea on which we are going to base it is very simple: to simulate an environment where the containers “shout” to the agents asking for somebody to take them to their destination. The intensity of the shout of each container depends on the remaining time before its deadline and the distance between its position and the delivery position (it could also depend on the importance of each container, but we must remember that the way we defined the problem, they are all equally important). The robots hear the calls of the containers diminished by the distance, so they go and take the ones they hear better. In order to achieve this behavior in the robots we will use a linear attraction function. Following the notation introduced in section 2, we define, for all container  $c$  and for all robot  $l$ , the attraction function  $F$  in the following way:

$$F(c, l) = \begin{cases} -\infty, & \text{if } c \text{ has been delivered.} \\ -\infty, & \text{if } c \text{ is being delivered for a} \\ & \text{robot other than } l. \\ K_1 \cdot urg(c) - K_2 \cdot cost(c, l), & \text{ow.} \end{cases} \quad (1)$$

Where  $K_1$  and  $K_2$  are adjustable parameters. The urgency function  $urg(c)$  is defined as inversely proportional to the time which remains to  $c$ 's deadline and takes into account the time required for transporting the container to its destination point:

$$urg(c) = curtime + \frac{d(p(c), dest(c))}{meanspeed} - dline(c) \quad (2)$$

Where  $d$  is the Euclidean distance,  $curtime$  is the current time and  $meanspeed$  is an environmental constant which averages agents' speeds. The  $cost$  function is defined as follows:

$$cost(c, l) = \begin{cases} \infty, & \text{if } weight(c) \geq maxload(l). \\ \infty, & \text{if } d(p(l), p(c)) \geq maxdist(l). \\ \frac{d(p(l), p(c)) + d(p(c), dest(c))}{speed(l)}, & \text{ow.} \end{cases} \quad (3)$$

The election of this attraction function  $F$  is quite arbitrary. A non-linear function would probably better reflect the "hearing" metaphor we talked about before. In the same way, we could also have defined a more sophisticated urgency function, non-linearly increasing depending on the time to the containers' deadline, for example. Bar Systems are general enough to use any attraction, cost or urgency functions. The question is finding, for each problem, the function which will give the best results. Our choice of the attraction function  $F$  is based in its simplicity, in spite of which, it has allowed us to obtain very good results.

The behavior of the robots will be very simple and it will obey the algorithm described in section 2.1. Each robot will choose a container to go for and will go toward its position, will load it (if not any other robot has arrived first) and will take it to the delivery point. After that, it will repeat the cycle until no containers left to transport.

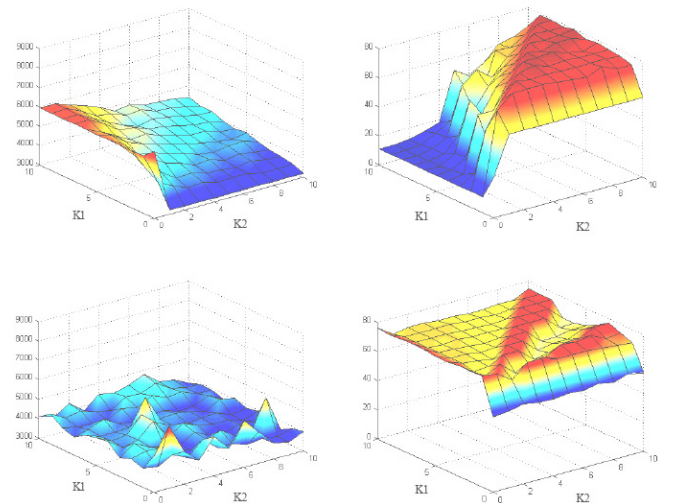
#### 4.1 Inter-agent communication and local planning for the CONTS problem

Aiming to the study of the utility of interagent communication, we will investigate two different methods for the choice of the next container to go for. If no communication between agents is allowed, each agent will simply choose the one which maximizes the attraction function. On the other hand, if the possibility of communication between agents is activated, each robot will ask to the others (perhaps not all of them but only those which communication is feasible) which containers they prefer and, in case of conflict (that is, another robot preferring the same container), a small negotiation process will start, the goal of which is to give preference to the agent who will be able to carry faster the container to its delivery position. The agent which finds itself in the situation where other agents have priority over it to transport its favorite container will try with the next best container, in order of preference according to its point of view, until it finds one for which it will have more priority than any other agent. It would be easy to devise more sophisticated negotiation processes taking into account the second-best options of the agents in conflict in such a way that one agent could resign carrying its preferred container, even if it has the higher preference over it, whenever the preference difference between the best and the second-best containers was small enough.

We have also implemented a very straightforward planning-like strategy in our Bar System. Whenever a robot has a container to go for, it looks if there exists another one such that it is possible to transport it without deviating too much from its original way to the first container position. If so, the agent transports it before resuming its original way to the first container position.

## 5 RESULTS

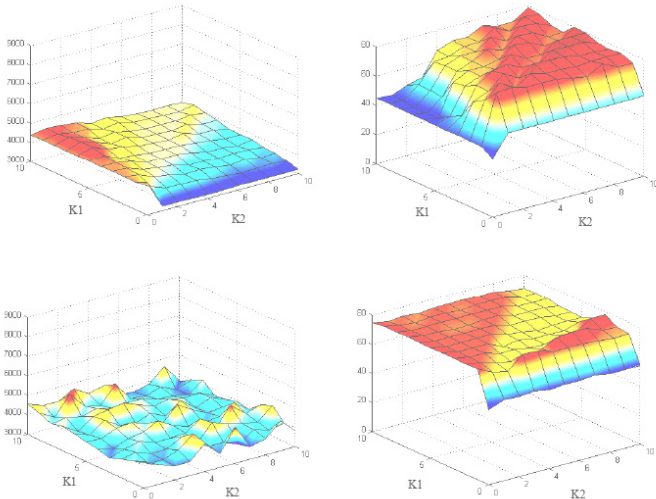
In order to analyze the efficiency of our method and experiment with different settings and parameter values, we have programmed a graphical simulator for the problem. We have chosen an instance of the problem with eighty containers randomly scattered on a  $300 \times 300$  rectangular area with random delivery points and deadlines and four carrier robots, all of them with the same value for the parameter  $maxdist$  and different speeds. We have done two main sets of simulations experimenting with different values of the parameters  $K1$  and  $K2$ . In the first set (figure 1) we don't allow agents to communicate or perform any local planning, whereas in the second set (figure 2) communication and local planning are permitted.



**Figure 1.** Left: Total time needed by the system to deliver all the containers for different values of the parameters  $K1$  and  $K2$  and for different values of the parameter  $maxdist$  (top row  $maxdist = 300$ , bottom row  $maxdist = 100$ ). Right: Number of containers delivered before their deadlines. Communication and local planning are deactivated.

We can see in figures 1 and 2 the results of the two sets of simulations. Each row represents a series of 121 simulations (for values of the  $K1$  and  $K2$  parameters ranging from 0 to 10 in increases of 1). The charts in the left columns show the time used to deliver all the containers and the charts in the right columns show the number of containers delivered before their deadlines. The two rows correspond to different values (300 and 100) of the  $maxdist$  parameter.

We can draw several conclusions. On one hand, it is clear that, for some values of the parameters  $K1$  and  $K2$ , the system finds much better solutions than those which can be obtained by using nearest neighbor-like methods. We can observe the performance of those methods in the top row of figure 1, when  $K1 = 0$  the preference function  $F$  depends only on the  $cost$  function and the systems



**Figure 2.** Left: Total time needed by the system to deliver all the containers for different values of the parameters  $K1$  and  $K2$  and for different values of the parameter  $maxdist$  (top row  $maxdist = 300$ , bottom row  $maxdist = 100$ ). Right: Number of containers delivered before their deadlines. Communication and local planning are permitted.

behaves in the “nearest container” way. The results are a low total delivery time and a considerable number of containers delivered after its deadline. The case  $K2 = 0$  is even worse. The system follows the “most urgent container” behavior, resulting in very long displacements which cause a big total delivery time and, consequently, a big number of containers delivered with retard. It is worth to remark that the improvement over those greedy methods achieved by our Bar System for some values of the parameters  $K1$  and  $K2$  is not attained in exchange of a greater complexity; in fact, the complexity of the system, understood as the amount of work which each agent has to do in order to decide the next container to go for, increases lineally with the number of containers.

We can also observe how the quality of the solutions found depends on the perceptual capabilities of the agents. When this capability is very limited (not shown in the figures), robots’ behavior is too local, resembling somewhat like a mixture of “nearest container” and random walk. On the other side, very good solutions are found for certain values of the parameters  $K1$  and  $K2$  when the agents are able to perceive the environment almost entirely ( $maxdist = 300$ ). This augmented perceptual ability implies, nevertheless, the possibility of appearance of several phenomena which can affect system’s efficiency, like, for instance, that it will be necessary to evaluate more alternatives, that the probability of conflicts will increase and that, depending on the values of the parameters, the system can arrive to very bad solutions if the agents must perform long displacements. Thus, a bit paradoxically, more perception power can yield poorer results. The most interesting case, from our point of view, is when the agents have a perceptual capability between the two extreme points. We have tested the case  $maxdist = 100$  and we can see in the bottom row of figure 1 how the system finds good solutions for most values of the parameters  $K1$  and  $K2$ . There are particularly, two big zones in the parameters space where the solutions found are as good as the ones obtained by the agents of the first row of the figure, which have a perceptual power nine times greater.

In figure 2, we can see how the inter agent communication or negotiation and local planning can improve greatly, depending on the

values of the parameters, the quality of the solutions found. Clearly, the importance of communication between agents increases with the possibility of conflict, which is proportional to the agents’ perception power and decreases with the relative magnitude of the parameter  $K2$  regarding  $K1$ . The more  $K2$  grows regarding  $K1$ , the more importance is given to the distance to the container in the calculation of the preference function, the robots tend to prefer the nearest containers and the number of conflicts decrease, as well as the utility of communication.

It is interesting to note that for some values of the parameters  $K1$  and  $K2$ , communication and local planning capabilities does not improve system’s results. This is probably due to the fact that the results for those parameter values in the Bar System without communicative or planning capabilities are near-optimal (all the containers delivered in time). Nevertheless, it is clear that more work in this direction is needed in order to clarify communication and local planning effects.

## 6 CONCLUSIONS AND FUTURE WORK

We have presented Bar Systems, a new class of reactive multi-agent systems for distributed problem solving. They are loosely inspired in the way a group of waiters work behind a bar. We have formalized the definition and we have given the general algorithm which rules the behavior of the individual agents. Several of the characteristics of Bar Systems are:

- **Simplicity.** Agents in Bar Systems are simple. They share a similar structure and operate in a decentralized manner in a similar way optimizing a local function. Interactions between agents are simple, too.
- **Efficiency.** Bar Systems have lineal complexity with respect to the number of tasks.
- **Robustness.** Faults in individual agents do not decrease dramatically the efficiency of the system. Moreover, Bar Systems’ problem solving capabilities increase steadily with the addition of new agents.
- **Responsiveness.** Bar Systems respond easily to the occurrence of unforeseen events as the appearance of new high priority tasks.

All those characteristics, jointly with the capability to seamless integrate different more or less sophisticated negotiation and local planning techniques, make Bar Systems very suitable to solve problems in asynchronous, dynamical, partial information and time-critical environments.

To check the efficiency and applicability of Bar Systems, we have defined a NP-Hard problem called CONTS, based on the work which a set of robots has to perform to transport a set of containers in time to their destination. The Bar System used to solve it has proved to give much better results than other greedy algorithms of the nearest neighbor type and has established, in our opinion, the usefulness of Bar Systems as a general framework for solving this type of real time problems. We have also seen that communication amongst agents and local planning allows improving the results greatly without increasing the complexity of the system significantly.

Our work in Bar Systems is just starting and we are aware that there are many aspects that require more study and testing. Some of the directions in which it would doubtless be worth working and which essentially refer to the nature of the attraction function  $F$  are:

- Study of more sophisticated negotiation strategies in case of conflict (two agents preferring the same task) and new local planning operators and its impact in system’s performance.

- Study of Bar Systems' performance in highly dynamical, time-critical environments. We are currently considering its use in the ROBOCUP and RESCUE environments.
- Study of the applicability of Bar Systems to other kinds of problems. At a first glance it could seem Bar Systems to be a bit too restrictive with respect to the kind of problems which they can tackle, It must be remarked, nevertheless, that its application is not limited to problems involving the transportation of goods or people (and we don't mean it to be a narrow application field, it is wide enough to contain problems ranging from service assignation in cab companies to multi-agent autonomous terrain exploration), they can also be useful in other problems which do not necessarily involve physical movement of the agents or goods transportation, such as resource allocation problems in the style of flexible Open-Shop problems where the order in which a set of machines, in a factory, for instance, has to perform a set of operations has to be decided. In this type of problems, machines would correspond to the robots in the CONTS problem, tasks would correspond to containers transportations and the preconditions and postconditions for the tasks would correspond to the initial and destination positions of the containers in the yard. Moreover, with an appropriate definition of the attraction function  $F$ , a Bar System can be used for solving flexible Job-Shop problems, where there is a set of independent jobs, each formed by a series of tasks which have to be done in a specific order. In this kind of problems, for each job there is at all times, at the most, just one feasible task, and it would be sufficient to define the attraction functions in such a way that all job's not done tasks "transmit" their urgency to the feasible one. The same idea could be used in a more general setting, where there would simply be any type of non-cyclical precedence relations over the set of tasks. It can also be worth to study the applicability of Bar Systems in competitive environments.

## REFERENCES

- [1] Holland O.E. Beekers, R. and Deneubourg J.L., *From Local Actions to Global Tasks: Stigmergy in Collective Robotics*, 181–189, Artificial Life IV, MIT Press, Cambridge, MA, 1994.
- [2] E. Bonabeau, M. Dorigo, and G. Thraulaz, *Swarm Intelligence. From Nature to Artificial Systems*, Oxford University Press, 1st edn., 1999.
- [3] E. Bonabeau and G Thraulaz, 'Swarm smarts', *Scientific American*, **March 2000**, 72–79, (2000).
- [4] E. del Acebo, 'Agents, sistemas multiagent i soluci distribuïda de problemes', *Research Report. Institut d'Informàtica i Aplicacions. Universitat de Girona*, (2005).
- [5] M. Dorigo and T. Sttzele, *Ant Colony Optimization*, MIT Press, 2004.
- [6] P. Engelbrecht A., *Fundamentals in Computational Swarm Intelligence*, John Wiley and Sons, 2006.
- [7] M. A. Lewis and G. A. Bekey, *The Behavioral Self-Organization of Nanorobots Using Local Rules*, Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992.
- [8] K.E. Parsopoulos and M.N. Vrahatis, 'Recent approaches to global optimization problems through particle swarm optimization', *Neural Computing*, **1 (2-3)**, 235–306, (2002).
- [9] M. Resnick, *Turtles, Termites and Traffic Jams, Explorations in Massively Parallel Microworlds*, MIT Press, 1997.