13

Artificial Societies and Social Simulation using Ant Colony, Particle Swarm Optimization and Cultural Algorithms

Alberto Ochoa[1,ض], Arturo Hernández[2], Laura Cruz[3], Julio Ponce[4], Fernando Montes[5,ض], Liang Li[6] and Lenka Janacek[7]
[1]*University of Juarez City,*
[ض] *ISTC-CNR,*
[2]*CIMAT,*
[3]*ITCM,*
[4]*Aguascalientes University,*
[5,ض] *Veracruzana University,*
[6]*University of Singapore,*
[7]*Zilina University*
[1,2,3,4,5]*México,*
[ض]*Italy,*
[6]*Singapore*
[7]*Slovak Republic*

1. Introduction

The proposal of this chapter is to explain the implementation of collective intelligent techniques to improve results in artificial societies and social simulation using diverse concepts such as argumentation, negotiation and reputation models to improve social simulation of artificial societies implementing dioramas, and multivariable analysis in different application domains for example Logistics. These techniques will be useful for answering diverse queries after gathering general information about a given topic. This kind of collective intelligence will be characterized by: ant colony, particle swarm optimization, and cultural algorithms, each one of these implementing diverse models or agents for simulate a social behaviour. Intelligent agents are used to obtain information to take decisions that try to improve the heuristic optimization needed in different application and fields of knowledge.

First, in section 1 of this paper, we approach different concepts related with Artificial Societies and Social Simulation using different strategies to analyze and model the necessary information to support the correct decisions of the evolving models. In other sections we explain the way to generate a specific behaviour with collective-intelligence techniques –ant colony (section 2), particle swarm optimization (section 3), and cultural algorithms (section 4). In section 5 we apply this knowledge in diverse fields and application domains that needs a heuristic optimization and the more innovative perspectives of each technique. In

Source: New Achievements in Evolutionary Computation, Book edited by: Peter Korosec,
ISBN 978-953-307-053-7, pp. 318, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

section 6 we analyse three cases of study: Logistics using a point of view of each one of these techniques (Ant Colony, Particle Swarm Optimization and Cultural Algorithms); Stratagems in an intelligent game board (Cultural Algorithms), and a Friendship Algorithm to demostrate the concept of Six Degrees of Separation in Societies of Memory-Alpha (Cultural Algorithms). Finally, in section 7 we provide our conclusions and our main future research in each technique.

2. Social simulation: basic concepts.

Social simulation is the modeling or simulation, usually with a computer, of social phenomena (e.g., cooperation, competition, markets, social networks dynamics, among others). A subset within social simulations are Agent Based Social Simulations (ABSS) which are an amalgam of computer simulations, agent-based modeling, and the social sciences.

History and Development

The birth of agent based model as a model for social systems was primarily brought by computer scientist (Reynolds, 1994). (Epstein and Axtell, 1996) developed the first large-scale agent model, the Sugarscape, to simulate and explore the role of social phenomena such as seasonal migrations, pollution, sexual reproduction, combat, transmission of diseases, and even culture, more recently.

Types of Simulation and Modeling

Social simulation can refer to a general class of strategies for understanding social dynamics using computers to simulate social systems. Social simulation allows for a more systematic way of viewing the possibilities of outcomes. There are four major types of social simulation:

1. system level simulation.
2. agent based simulation.
3. system level modeling.
4. agent based modeling.

A social simulation may fall within the rubric of computational sociology which is a recently developed branch of sociology that uses computation to analyze social phenomena. The basic premise of computational sociology is to take advantage of computer simulations in the construction of social theories. It involves the understanding of social agents, the interaction among these agents, and the effect of these interactions on the social aggregate. Although the subject matter and methodologies in social science differ from those in natural science or computer science, several of the approaches used in contemporary social simulation originated from fields such as physics and artificial intelligence.

System Level Simulation

System Level Simulation (SLS) is the oldest level of social simulation. System level simulation looks at the situation as a whole. This theoretical outlook on social situations uses a wide range of information to determine what should happen to society and its members if certain variables are present. Therefore, with specific variables presented, society and its members should have a certain response to the situation. Navigating through this theoretical simulation will allow researchers to develop educated ideas of what will happen under some specific variables (Chira et al, 2008).

Agent Based Modeling

Agent based modeling (ABM) is a system in which a collection of agents independently interact on networks. Each individual agent is responsible for different behaviors that result

in collective behaviors. These behaviors as a whole help to define the workings of the network. ABM focuses on human social interactions and how people work together and communicate with one another without having one, single "group mind". This essentially means that it tends to focus on the consequences of interactions between people (the agents) in a population. Researchers are better able to understand this type of modeling by modeling these dynamics on a smaller, more localized level. Essentially, ABM helps to better understand interactions between people (agents) who, in turn, influence one another (in response to these influences). Simple individual rules or actions can result in coherent group behavior. Changes in these individual acts can affect the collective group in any given population.

Agent-based modeling is simply just an experimental tool for theoretical research. It enables one to deal with more complex individual behaviors, such as adaptation. Overall, through this type of modeling, the creator, or researcher, aims to model behavior of agents and the communication between them in order to better understand how these individual interactions impact an entire population. In essence, ABM is a way of modeling and understanding different global patterns.

Current Research

There are two current research projects that relate directly to modeling and agent-based simulation the following are listed below with a brief overview.

- Agent based simulations of Market and Consumer Behavior, is another research group that is funded by the Unilever Corporate Research. The current research that is being conducted is investigating the usefulness of agent based simulations for modeling consumer behavior and to show the potential value and insights it can add to long-established marketing methods.
- Agent based modeling is most useful in providing a bridge between micro and macro levels, which is part of what sociology studies. Agent based models are most appropriate for studying processes that lack central coordination, including the emergence of institutions that, once established, impose order from the top down. The models focus on how simple and predictable local interactions generate familiar but highly detailed global patterns, such as emergence of norms and participation of collective action. (Macy & Willer, 2002) researched a recent survey of applications and found that there were two main problems with agent based modeling the self-organization of social structure and the emergence of social order. Below is a brief description of each problem Macy and Willer believe there to be;

1. "*Emergent structure*. In these models, agents change location or behavior in response to social influences or selection pressures. Agents may start out undifferentiated and then change location or behavior so as to avoid becoming different or isolated (or in some cases, overcrowded). Rather than producing homogeneity, however, these conformist decisions aggregate to produce global patterns of cultural differentiation, stratification, and hemophilic clustering in local networks. Other studies reverse the process, starting with a heterogeneous population and ending in convergence: the coordination, diffusion, and sudden collapse of norms, conventions, innovations, and technological standards."
2. "*Emergent social order*. These studies show how egoistic adaptation can lead to successful collective action without either altruism or global (top down) imposition of control. A key finding across numerous studies is that the viability of trust, cooperation, and collective action depends decisively on the embeddedness of interaction."

These examples simply show the complexity of our environment and that agent based models are designed to explore the minimal conditions, the simplest set of assumptions about human behavior, required for a given social phenomenon to emerge at a higher level of organization (see Figure 1).

Fig. 1. Social Simulation using agents in an environment related with a wedding.

Researchers working in social simulation might respond that the competing theories from the social sciences are far simpler than those achieved through simulation and therefore suffer the aforementioned drawbacks much more strongly. Theories in social science tend to be linear models that are not dynamic and which are inferred from small laboratory experiments. The behavior of populations of agents under these models is rarely tested or verified against empirical observation.

3. Behaviour in group intelligent techniques – ant colony.

This section describes the principles of any Ant System (AS), a meta-heuristic algorithm based on the ant food-search metaphor. The description starts with the ant metaphor, which is a model of real ants. Then, it follows a discussion of how AS has evolved. The section ends with the explanation of ACS, the most common AS version. ACS Algorithm is the base of the logistic application explained in a later section.

3.1 A metaphor of real ants

The AS was inspired by collective behavior of certain real ants (forager ants). While they are traveling in search of food, they deposit a chemical substance on the traversed path. This substance, called pheromone, is detected with their antennae. The Pheromone communication is an effective way of coordinating the activities of these insects. For this reason, pheromone rapidly influences the behavior of the ants: they will tend to take those paths where there is a larger amount of pheromone.

The behavior followed by real ants is modeled as a probabilistic process. Without any amount of pheromone, the ant explores the neighboring area in a totally random way. In presence of an amount of pheromone, the ant follows a path in a controlled random way. With crossed paths, the ant will follow the trail with the largest amount of pheromone with a higher probability. ***All ants deposit additional pheromone during the*** traveling, then the food-search process evolves with positive feedback. Since the pheromone evaporates, the non-used trails tend to disappear slowly, increasing the positive feedback effect. In this stochastic process, the best ***ant*** receives reward with the highest amount of pheromone, while the worst ***ant*** is ***punished with the lowest amount of pheromone.***

3.2 Artificial ant colony

The AS is inspired in the natural optimization process followed by real ants. The algorithm is a general frame than can be applied to the solution of many combinatorial optimization problems. The artificial society, formed by ants, repeats the food-search process. Each ant builds a solution to the optimization problem. The ants share a pheromone structure, which is a common memory (global information) that can be accessed and updated simultaneously.

The AS is a multi-agent system where low level interactions between single ant-agents result in a complex behavior of the whole ant colony. Each ant-agent has incomplete information or insufficient skill to solve a problem. They need the whole colony to get the final objective. To optimize this kind of collaboration, there is not global control, data is decentralized, and the computation is asynchronous. Besides, the ant colonies exhibit an emergent behavior. This emergent conduct happens because they build their result in an incremental manner. As a consequence, the ant colonies are adaptive, complex and distributed multi-agent systems. The AS was originally proposed to solve the Traveling Salesman Problem (TSP), and the Quadratic Assignment Problem (QAP). Now exist a lot of applications like scheduling, machine learning, data mining (Ponce et al., 2009), and others. There are several variants of AS designed to solve specific problems or to extend the characteristics of the basic algorithm. The next paragraph describes the most important variants in order of appearance. Ant Colony Optimization (ACO) was introduced initially by Dorigo (Dorigo, 1991). ACO use two main characteristics: η_{rs} and τ_{rs}. The heuristic information η_{rs} is used to measure the predilection to travel between a pair of nodes (r,s). The trails of artificial pheromone τ_{rs} is used to compute the learned reference of traveling in a determined arc (r,s). It is formed by three algorithms: Ant-density, Ant-quantity and Ant-Cycle. Ant-density and Ant-quantity use the update of pheromone trails in every step of the ants, while Ant-Cycle makes updates after a complete cycle of the ant. A study of the correct configuration of AS for solving TSP concludes that the main parameter is β, the relative importance of the heuristic information. The study establishes that the optimal number of ants is equivalent to the number of nodes of the problem.

Gambardela and Dorigo (Gambardela, 1995) designed the Ant-Q algorithm, which is based on the principles of Q-learning. It was applied for solving the TSP and Asymmetric TSP (ATSP). Ant-Q uses a table of values Q to indicate how good a determined movement from node r to s is. It applies a rule to choose the next node to be visited and uses reinforcement learning to update Q with the best tour of the ants. Max-Min Ant System algorithm (MMAS) was developed by Stützle and Hoose (Stützle, 1996). It uses the elements of AS. However, it modifies three aspects: updating rule, pheromone values, and the next movement. The updating rule was modified to choose the best tour in every cycle, increasing the probability of early stagnation. Maximum and minimum limits for the pheromone trails were established. These limits avoid repeated movements: bounding the influence of the trail intensity, and leading to a higher degree of exploration.

Other variant of AS, named ASrank, was developed by Bullnheimer, Hartl and Strauss. (Bullnheimer et al., 1997). All solutions are ranked according to their fitness. The amount of deposited pheromone is weighted for each solution, such that the best result deposits more pheromone than bad solutions. This algorithm was tested with TSP and VRP instances. The developed technique is based on the Distributed Q-Leaning algorithm (DQL).

3.3 Ant Colony System (ACS)

Dorigo and Gambardela (Dorigo & Gambardela, 1996) improved AS with an algorithm named Ant Colony System (ACS). It presents three main differences with regard to AS:

transition rule, global updating and local updating. The transition-state rule is modified to establish a balance between the exploration of new arcs and the priority exploitation of a problem. The global updating rule is applied only to the arcs of the best ant tour. A local updating of the pheromone is done while ants build a solution. ACS was applied to TSP and ATSP with the addition of a local search based on a 3-opt scheme. Figure 2 shows a general scheme of the ACS algorithm.

```
Ant_Colony_System ( )
Initialize Data Structures
Do
   For each ant: initialize its solution
   Do
        For each ant:
          Pseudo-random-rule(η_rs, τ_rs)is applied to build a solution
            Local-update(τ_rs)
   Until all ants have completed their solutions
   Global-update(τ_rs)
Until stop criteria is reached
```

Fig. 2. The ACS algorithm.

It is well known that ACS is one of the best ant algorithms. ACS has the best performances and the majority of references (Asmar, 2005). This algorithm was adapted to solve the logistic problem approached in a section 6.1.

4. Behaviour in group intelligent techniques – particle swarm optimization.

The Particle Swarm Optimization (PSO) algorithm is a population-based optimization technique inspired by the motion of a bird flock (Kennedy, J. & Eberhart R., 1995). In the PSO model, every particle flies over a real valued n-dimensional space of decision variables $\vec{X}$. Each particle keeps track of its position $\vec{x}$, velocity $\vec{v}$, and remembers the best position ever visited, P_{Best}. The particle with the best P_{Best} value is called the leader, and its position is called global best, G_{Best}. The next particle's position is computed by adding a velocity term to its current position, as follows:

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1} \tag{1}$$

The velocity term combines the local information of the particle with global information of the flock, in the following way.

$$\vec{v}_{t+1} = w * \vec{v}_t + \phi_1 * (P_{Best} - \vec{x}_t) + \phi_2 * (G_{Best} - \vec{x}_t) \tag{2}$$

The equation above reflects the socially exchanged information. It resumes PSO three main features: distributed control, collective behavior, and local interaction with the environment (Eberhart et al., 1996). The second term is called the cognitive component, while the last term is called the social component. w is the inertia weight, and ϕ_1 and ϕ_2 are called acceleration coefficients. The inertia weight indicates how much of the previous motion we

want to include in the new one. When the flock is split into several neighborhoods the particle's velocity is computed with respect to its neighbors. The best P_{Best} value in the neighborhood is called the local best, L_{Best}.

$$\vec{v}_{t+1} = w * \vec{v}_t + \phi_1 * (P_{Best} - \vec{x}_t) + \phi_2 * (L_{Best} - \vec{x}_t) \quad (3)$$

Neighborhoods can be interconnected in many diferent ways, some of the most popular are shown in Fig. 3. The star topology is, in fact, one big neighborhood where every particle is connected to each other, thus enabling the computation of a global best. The ring topology allows neighborhoods therefore it is commonly used by the PSO with local best.
PSO is a fast algorithm whose natural behavior is to converge to the best explored local optima. However, attainting flock's convergence to the global optimum with high probability implies certain adaptations (Garro, 2009). The approaches range from modifications to the main PSO equation, to the incorporation of reproduction operators. PSO algorithm has been extended in several directions; in a latter section we show a multiobjective optimization approach to solve a logistic problem, the vehicle routing problem with time windows.

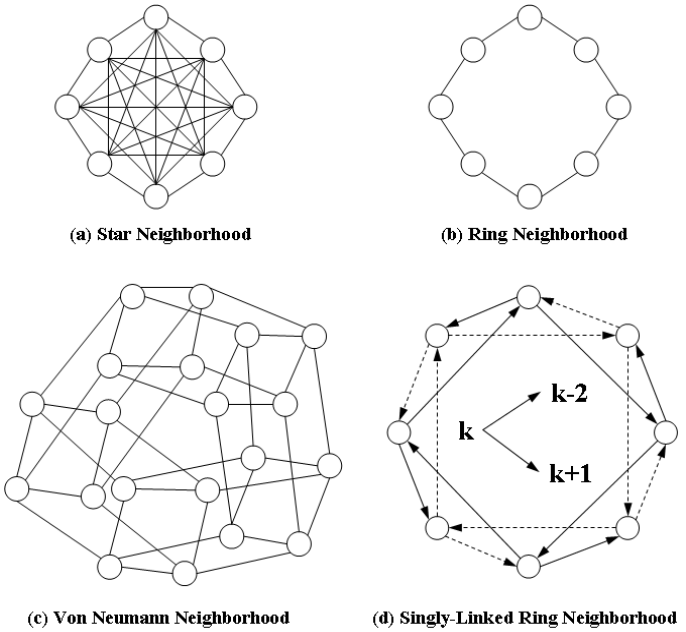


Fig. 3. Neighborhood structures for PSO

5. Behaviour in group intelligent techniques – cultural algorithms.

Cultural algorithms were developed by Robert G. Reynolds in 1994 as a complement to the evolutionary algorithms, this algorithms are bio-inspired in the Cultural Evolution of the Societies, and were focused mainly on genetic and natural selection concepts (Reynolds, 1994). Cultural algorithms operate at two forms: (1) a micro-evolutionary form, which

consists of the genetic material that an offspring inherits from its parents (children, grandsons, greats-grandchild, etc), and (2) a macro-evolutionary level, which consists of the knowledge acquired by the individuals through generations (Culture). This knowledge is used to guide the behavior of the individuals that belong to a certain population. Figure 4 illustrates the basic framework of a cultural algorithm. A cultural algorithm operate in two spaces: Population Space and Belief Space. The most of computational problems found at real World, do not have a definitive (final) solution (Desmond & Moore, 1995). Cultural Algorithms uses the culture like a vehicle to store accessible relevant information to the population's members during many generations, and were developed to model the evolution of the cultural component over time and to demonstrate how this learns and acquires knowledge.

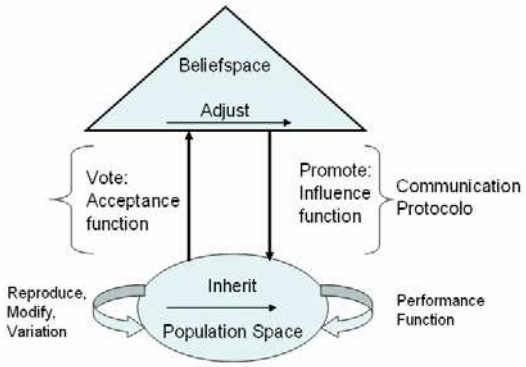


Fig. 4. Conceptual Diagram of Cultural Algorithms.

The Population Space are evaluated with a performance function obj(), and an acceptance function accept() can help to determine which individuals are introduce in the Belief Space. Experiences of those chosen elites will be used to update the knowledge / beliefs of the Belief Space via function update(), this function represents the evolution of beliefs in the population. Next, the beliefs are used to influence the evolution of the population. New individuals are generated under the influence of beliefs in the time that were crates. The two feedback paths of information, one through the accept() and influence() functions, and the other through individual experience and the obj() function create a system of dual inheritance of both population and belief. The population component and the belief space interact with and support each other, in a manner analogous to the evolution of human culture (Wu & Hsiao, 2008).

List of belief space categories

Normative knowledge A collection of desirable value ranges for the individuals in the population eg. acceptable behavior for the agents in population.

Domain specific knowledge Information about the problem domain where cultural algorithms are applied.

Situational knowledge Specific examples of important events - eg. succesful/unsuccesful solutions

Temporal knowledge History of the search space - eg. the temporal patterns of the searching process

Spatial knowledge Information about the topography of the search space

6. Use of collective intelligence techniques in diverse heuristics optimization.

The use of collective intelligence is found in different areas ranging from biology, sociology, and business to computer science. However, a common definition looks for the identification of certain "intelligence" derived from the efforts of joined entities. The aforementioned entities range from bacteria, animals, human to computer processes. Furthermore, the same definition can be applied, in a broad sense, to action selection and evolutionary robotics.

Evolutionary Robotics employs a quasi-optimal approach to develop autonomous controllers for different kinds of robots. The use of genetic algorithms and neural networks are natural candidates, as the preferred methodology, for developing single evolved neural controllers. These controllers are the result of testing populations of adapted individuals during a refinement process through series of computer-program iterations. Next, pairs or groups of individuals can be evolved together. Following this approach a change in the evolution of one individual can be affected by the change of other related individuals in the group. The latter approach has been identified, as its biological counterpart, as co-evolution that can be cooperative or competitive. A cooperative strategy can be developed to achieve a common task (e.g. pushing objects, solving a common task), whereas in a competitive strategy individuals have to struggle to assimilate some scarce resources (e.g. prey and predator, securing of food stashes). In biology diffuse co-evolution has been referred to species evolving in response to a number of other species, which in turn are also evolving in response to a set of species. Consequently, the identification of collective intelligence is more evident when coevolving cooperative groups.

The development of collective behavior in simulation can be achieved by simple scalable control systems as a form of decentralized control. Therefore, the work of (Kube, 1993) exhibited group behavior without the use of explicit communication. A more recent approach by (Marroco & Nolfi, 2007) set a collective task where a group of four robots developed the ability to reach two different target areas and to group in pairs at each different location. Elementary communication skills were evolved alongside the abilities to reach target areas. In this way, coordination is achieved through stigmergy; nevertheless more elaborated communication can be implemented (Mitri, 2006). On the whole, the individual ability to communicate may be one of the requirements for the development of collective intelligence.

The evolutionary approach has been sufficient to solve problems were cooperation and communication skills are necessary for solving a particular task; in contrast communication arises as a deceptive feature within a competitive scenario. In this scenario a common setting is that of the prey and the predator where both individuals are competing for scoring points either for escaping or capturing each other. The experiment can be expanded to add more preys and predators. It is important to notice that the prey/predator sees the other as a source that needs to be secured/avoided making this an instance of the 'action selection' problem (Ochoa et al., 2008).

The action selection problem is related, to the behavior-based approach, and particularly to decision-making when a module takes control of the available actuators until is completed or proves ineffective. In the vertebrate brain, at specific loci, specialized centers of selection can be identified. One of them are the basal ganglia, and recent works support the idea of these nuclei playing an important role in action selection (Prescott et al., 2006). The basal ganglia act as a relay station in the planning and the execution of movements (behavior);

hence gathering information from the cortex and motor cortex. The basal ganglia are able to mediate cognitive and muscular processes. Not only the basal ganglia serves as an important center of action selection; in cooperation with the cerebellum and the sensory cerebrum, all them are able to veto muscular contraction by denying the motor areas sufficient activation. In turn, these individual motor elements form more complex patterns, which can be thought as essential elements in the development of intelligence. The development of intrinsic basal ganglia circuitry with evolvable behavioral modules has already been implemented (Montes et al., 2007). Cooperative individuals not only require a society interaction, but the existence of an internal mechanism (e.g. the basal ganglia) that is able to mediate amongst various sensory processes. Nonetheless, these sensory processes need to be augmented when possible. Therefore, individuals need to build up unified internal perceptions based on their available sensory capabilities in order to produce specialized behavior. The work of (Montes et al., 2008) shows how non-standard avoidance can be achieved by extending sensory information through an evolutionary refinement.
The emergence of collective intelligence based on the behavior-based approach requires stepping out from the modeling of selfish solitary individuals to social organisms. Therefore, we need to group our robots and expose them to numerous interactions to assure complex performances at the level of the group. Here, we have identified some common elements in collective robotics: cooperation, intelligence, communication skills, and the integration of sensory information with action selection. All of those accomplished with the use of the evolutionary robotics approach. As a consequence, we believe that we may contribute to the development of robotic collective intelligence by way of social experiments using the artificial evolutionary method.

7. Cases of studies related to cultural algorithms

Many applications inspired in evolving compute have a great value in Logistics. In this section, we present five in special to compare their contributions, the first is related with Ant Colony in Logistic (6.1 Subsection), another is the use of Particle Swarm Optimization in Logistics (6.2 Subsection), the last three are related with Cultural Algorithms (6.3 Subsection) whih is used to improve Bin Packing Algorithm in a problem of Logistics, another is focused in an interactive game board using the problem of negotiation for obtaining a best situation in the game; in the other way is showed the anlysis of a social networking represented with a Dyoram to determine Six degree of separation in a graph.

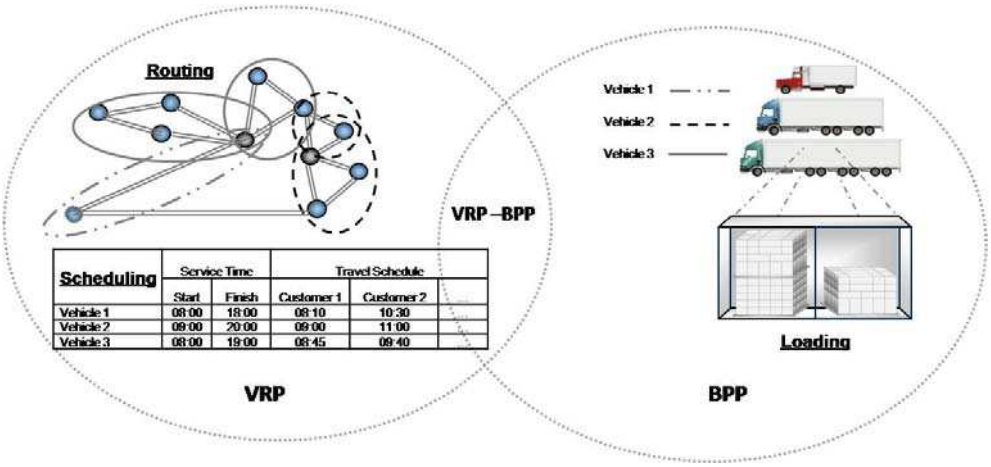
7.1 Ant colony in logistic

Many manufacturing companies need merchandise delivered with the minimum quantity of resources and in due time. An optimal solution to this logistic problem could save between 5 to 20 % of the total cost of the products (Toth, 2002). However, this is a high-level complex problem. This type of word problem, named recently *rich problem*, includes several NP-hard sub-problems with multiple interrelated variants. To contribute to this area we approach the bottled-products distribution in a Mexican company. Our proposal includes an innovative ACS solution.

The Routing-Scheduling-Loading Problem (RoSLoP)

RoSLoP involves three tasks: routing, scheduling and loading. They are formulated with two well-known classical problems: Vehicle Routing Problem (VRP) and Bin Packing

Problem (BPP). The routing and scheduling tasks are defined through VRP, while the loading task is stated through BPP. Fig. 5 shows this formulation.



Scheduling	Service Time		Travel Schedule		
	Start	Finish	Customer 1	Customer 2	
Vehicle 1	08:00	18:00	08:10	10:30	
Vehicle 2	09:00	20:00	09:00	11:00	
Vehicle 3	08:00	19:00	08:45	09:40	

Fig. 5. Routing-Scheduling-Loading Problem (RoSLoP)

RoSLoP includes constrains not considered in VRP and BPP. Previous work approached separately at the most, three variants of BPP (Chan et al., 2005), and five variant of VRP (Pisinger, 2005). Commercial applications have incorporated up to eight variants of VRP without available scientific documentation (OR/MS, 2006). To overcome these limitations, our research tries simultaneously with eleven variants of VRP (Cruz et al., 2008) and five variants of BPP (Cruz et al., 2007).

In order to generalize this problem, RoSLoP is formulated as follows: Given a set of customers with a demand to be satisfied, a set of depots that are able to supply them, and a set of BPP and VRP variants that restrict them, the routes, schedules and loads for vehicles needs to be designed. The Customer demands must be completely satisfied, so the total cost is minimized and the constraints are satisfied.

Methodology of Solution

The assignment of routes and schedules is solved by an ACS algorithm. Three elements complement this algorithm: an auto-adaptive constrained list; an initial search, implemented with the Nearest Neighborhood; and a local search, implemented with 3-opt and Cross-Exchange. The loads are assigned by DiPro algorithm (Cruz et al., 2007). Fig. 6 shows the interaction between the components of ACS and DiPro.

The ACS algorithm in Fig. 7 begins creating an initial solution (line 1). After, it creates an ant colony to minimize the number of vehicles used (lines 3 to 17). In this process, each ant builds a local solution sailing through the adjacent states of the problem. The election of the nodes is done through a pseudo-random selection rule (line 6 to 13). The local solution is compared with respect to the best global solution (line 18) and the updates are done (line 19-20). The stop conditions of lines 17 and 21 are specified with respect to the number of iterations and time of execution respectively. A description of equation and techniques used by ACS are given below.

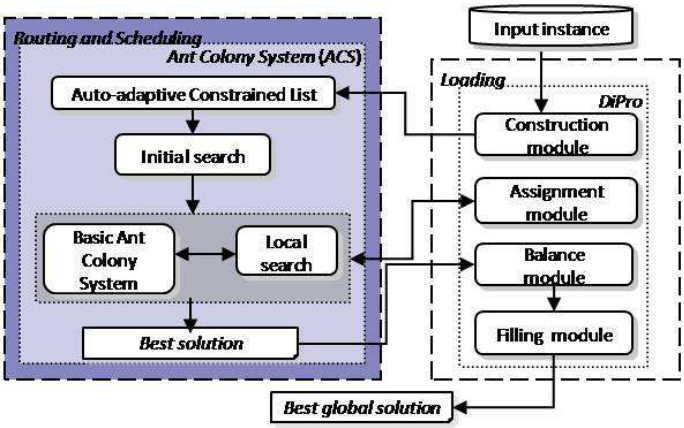


Fig. 6. ACS-DiPro: a solution Methodology of RoSLoP

```
ACS_RoSLoP ( )
1  Create a feasible global solution
2  repeat
3     repeat /*build an initial local solution*/
4        Initialize a local solution and get its number of vehicles t
5        for each ant:
6           repeat /*build a new local solution */
7              Select a vehicle with the Pseudo-random-rule(η_v, τ_v)
8              repeat
9                 Select a customer with the Pseudo-random-rule(η_rs, τ_rs)
10                Customer-Local-update(τ_rs)
11             until no customer can be visited
12             Vehicle-Local-update(τ_v)
13          until the demand is satisfied or t is reached
14          local_search(new local solution)
15          Update the local solution if a better solution is found
16       end for each
17    until stop criteria is reached
18    if the local solution is feasible and better than the global
19       Update the global solution
20       Customer-Global-update(τ_rs)and Vehicle-Global-update(τ_v)
21 until stop criteria is reached
```

Fig. 7. The ACS algorithm for RoSLoP

Pheromone Update: Global update of the pheromone trail for customers is done over the best solution. Eq. 4 evaporates the pheromone in all the edges used for the best ant; the evaporation rate is $\rho \in [0,1]$. It also adds a reward using the increment $\Delta\tau_{rs}$, which is the inverse of the length of the best global solution. Local update (Eq. 5) is applied every time that one ant travels from node *r* to node *s*. It uses τ_0, which is the inverse of the product of the length of the shortest global solution, and the number of visited nodes. Similar equations are used for the vehicle pheromone trail.

$$\tau_{rs} \leftarrow (1-\rho)\tau_{rs} + \rho\Delta\tau_{rs} \tag{4}$$

$$\tau_{rs} \leftarrow (1-\rho)\tau_{rs} + \rho\tau_0 \tag{5}$$

Heuristic Information for Customers: The Eq. 6 determines the heuristic information η_{rs} used in the customer selection. In this equation, Δt_{rs} is the difference between the current time of node r and the arrival time to the node s, ws_s is the remaining size of the time window in s, st_s is the service time in s, and tc_{rs} is the travel cost from r to s.

$$\eta_{rs} = \left(\Delta t_{rs} \cdot (ws_s + st_s) \cdot tc_{rs}\right)^{-1} \tag{6}$$

Heuristic Information for Vehicles: The Eq. 7 calculates the heuristic information η_v used in the selection of the vehicles. In this equation, nv_v is the quantity of trips required by the vehicle v to supply all the demands, $\overline{TM_v}$ is the average of the service time of a vehicle v, $\overline{TR_v}$ is the average travel time of a vehicle v, tr_v is the available service time of the vehicle v, tt_v is the total service time of the vehicle v, and $idpref_v$ is the predilection-of-use grade of a vehicle v.

$$\eta_v = \left(nv_v \cdot (\overline{TM_v} + \overline{TR_v}) \cdot \frac{tr_v}{tt_v} \cdot idpref_v \right)^{-1} \tag{7}$$

Pseudo-random selection rule: An ant k located in node r selects the next node s to move. The selection is made using the pseudo-random selection rule defined in Eq. 8. When a balancing condition is satisfied, the ant selects the best node exploiting the heuristic information and the trails of pheromone. Otherwise, a proportional random exploration is applied. In this equation, q_0 is a parameter of balancing between exploitation and exploration, q is a random value in [0,1], β is the relative importance of the heuristic information, $N_k(r)$ is the set of available nodes for r.

$$\begin{aligned} &\text{if } q \le q_0 \quad s = \arg\max_{s \in N_k(r)} \{\tau_{rs}\eta_{rs}^{\beta}\} \quad \text{Si } s \in N_k(r) \\ &\text{else} \qquad p_{rs}^{x} = \begin{cases} \dfrac{\tau_{rs}\eta_{rs}^{\beta}}{\sum_{s \in N_k(r)} \tau_{rs}\eta_{rs}^{\beta}} & \text{Si } s \in N_k(r) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{8}$$

***Auto-adaptive Constrained List* (*ACL*):** The ACL characterizes the instance graph into subsets that have similar conditions. First, a Minimum Spanning Tree (MST) is generated. When the variability of the cost associated to each path in MST is small, all the nodes form a single conglomerate. Otherwise, it forms conglomerates through a hierarchical grouping. The value of the heuristic information η_{rs} is modified with a factor, which is a ratio of the cardinality of the groups of r and s.

Experimentation

A sample of three real instances and its solution is shown in Table 1. The database contains 312 instances classified by date order, and 356 products. The algorithm was coded in c# and set with 10 ants, 5 colonies, 40 generations, q_0 = 0.9; β = 1; ρ = 0.1. The results show that

ACS_DiPro uses an average of 5.66 vehicles. According with our industrial partner, it represents approximately an average saving of 11% in comparison with the manual solution. Besides, the algorithm execution takes 55.27 s. In contrast, a human expert is dedicated on doing this activity all working day.

Instance	*ORDERS*	ACS_DiPro		
		Distance Traveled	Vehicles Used	Time (s)
13/02/2006	208	1980	5	55.57
06/03/2006	224	1960	5	32.16
09/03/2006	269	2570	6	76.18
…	…	…	…	…
Average	**238.83**	**2245.16**	**5.66**	**55.27**

Table 1. Solution of real instances of RoSLoP

7.2 Logistic application: solving the vehicle routing problem with time window using PSO.

In transportation management, there is a requirement to provide goods and/or services from a supply point to various geographically dispersed points with significant economic implications. The vehicle routing problem (VRP), which was first introduced by (Dantzig & Ramser, 1959), is a well-known combinatorial optimization problem in the field of service operations management and logistics.

The Vehicle Routing Problem concerns the transport of items between de-pots and customers by means of a fleet of vehicles. In the VRP, the decisions to be made define the order of the sequence of visits to the customers; they are a set of routes. A route departs from the depot and it is an ordered sequence of visits to be made by a vehicle to the customers, fulfiing their orders. A solution must be verified to be feasible, checking that it does not violate any constraint, such as the one stating that the sum of the demands of the visited vertices shall not exceed the vehicle capacity.

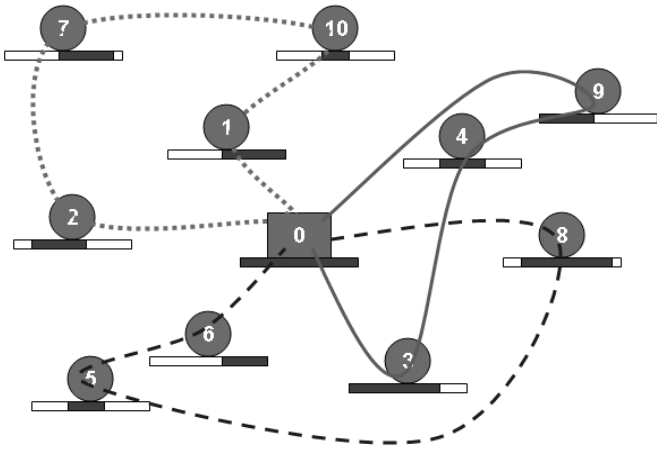


Fig. 8. VRPTW example

The Vehicle Routing Problem (see Figure 8) with Time Windows (VRPTW), is a variant of the VRP which considers the available time window in which either customer has to be supplied. The VRPTW is commonly found in real world applications and is more realistic than the VRP that assumes the complete availability over time of the customers.

Each customer requests a given amount of goods, which must be delivered or collected at the customer location. Time intervals during which the customer is served can be specific. These time windows can be single or multiple. A vehicle can not arrive later than a given time, but it can wait if arriving early. In such a case, the goal of the objective function is to minimize the distance travelled and the waiting time. (Salvelsberg, 1985) proved that even finding a feasible solution to the VRPTW when the number of vehicles is fid is itself a NP-complete problem. An overview on the VRPTW formulation and approaches can be found in (Cordeau et al., 2002).

VRPTW Formulation

The VRPTW is represented by a set of identical vehicles denoted by K, and a directed graph G, which consist of a set of customers and a depot. The nodes 0 represents the depot. The set of n vertices denoting customers is denoted N. The arc set A denotes all possible connections between the nodes (including the node denoting depot). All routes start at node 0 and end at node 0. We associate a cost c_{ij} and a time t_{ij} with each arc (I,j) ε A of the routing network. The travel time t_{ij} includes service time at customer *i*. Each vehicle has a capacity limit q_k and each customer *i*, a demand d_i, *i* ε N. Each customer *i* has a time window, $[a_i, b_i]$, where a_i and b_i are the respective opening time and closing times of *i*.

No vehicle may arrive past the closure of a given time window, b_i. Although a vehicle may arrive early, it must wait until the start of service time a_i is possible. It generates a waiting time w_k for the route. Vehicles must also leave the depot within the depot time window $[a_0, b_0]$ and must return before or, at time b_0. For eliminating any unnecessary waiting time (Russell, 1995), we assume that all routes start just-in-time $w_k = 0$, that is, we adjust the depot departure time of each vehicle $t_k = max(0, a_i - t_{0i})$, $i \varepsilon N$.

Multi-Objective Optimization

In Multi-Objective Optimization (MOO), two or more conflicting objectives contribute to the overall result. These objectives often affect one another in complex, nonlinear ways. The challenge is to find a set of values for them which yields an optimization of the overall problem at hand.

A MOO problem is defined as follows:

$$min_{\vec{x}} \quad \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})] \tag{9}$$

subject to

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{10}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, \ldots, p \tag{11}$$

where $\vec{x} = [x_1, x_2, \ldots, x_d]$ is the vector of decision variables.

The notion of "optimum" in MOO problems difers from that of a single function in global optimization. Having several objective functions the aim is to find good compromises, rather than a single solution.

In 1896, Vilfredo Pareto, an Italian economist, introduced the concept of Pareto dominance (Pareto, 1896). A vector $\vec{x}$ is preferred to (dominates) a vector if each parameter of $\vec{x}$ is no greater than the corresponding parameter of $\vec{y}$ and at least one parameter is less.

Definition 1 *A vector $\vec{x}$ is said to dominate vector $\vec{y}$,*

$$\vec{x} \preceq \vec{y} \tag{12}$$

if and only if $\vec{x}$ is partially less than $\vec{y}$.

$$\forall i \in \{1,2,\ldots,n\}, \vec{x}_i \leq \vec{y}_i \wedge \exists i \in \{1,2,\ldots,n\} : \vec{x}_i < \vec{y}_i \tag{13}$$

Figure 9 shows an example of the Pareto dominance concept between solution vectors *B* and *C*; $B \preceq C$ due because *B* is less than *C* in the two objective functions f_1 and f_2.

A useful concept in MOO, related with Pareto dominance, is the Pareto front. The Pareto front is composed by a set of non-dominated vectors. Figure 9 shows a comparison between two individuals of the Pareto Front, where A is less than B in objective function f_1, but *B* is less than *A* in objective function f_2; therefore both solution vectors are non-dominated.

The Pareto front is particularly useful in engineering: by restricting attention to the set of solutions that are non-dominated, a designer can make tradeoffs within this set, rather than considering the full range of every parameter.

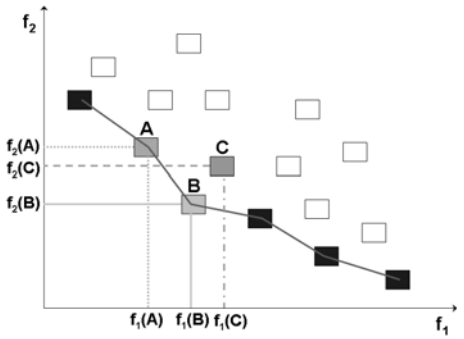


Fig. 9. Pareto Front

Multiobjective approaches to VRPTW

The three objectives of the VRPTW were presented: total distance travelled, number of vehicles, and total waiting time. Many existing VRPTW techniques, however, are single objective-based heuristic methods that incorporate penalty functions, or combine the different criteria via a weighting function (Hasnah, 2002; Li & Lim, 2003). However, in the last five years, some hybrid algorithms have been proposed to solve the VRPTW as a MOO problem. (Tan, 2006), and (Ombuki, 2006), employed the travel distance and the number of vehicles to be minimized. (Chitty & Hernandez, 2004), tried to minimize the total mean transit time and the total variance in transit time. (Murata & Itai, 2005) consider to minimize the number of vehicles and the maximum routing time among the vehicles in order to minimize the active duration of the central depot. (Saadah et al., 2004), minimize the number of vehicles and the total distance travelled. This work proposes a hybrid particle swarm MOO algorithm that incorporates perturbation operators for keeping diversity in the evolutionary search and the concept of Pareto optimality for solving the VRPTW as a MOO problem with 3 objectives: minimize total travel distance, minimize number of vehicles and minimize total waiting time.

Tackling Solomon's logistic problem with MO-PSO

Our Multiobjective PSO (MO-PSO) uses a flock size of 100 members,applies a PSO with parameters $c_1 = 1$ and $c_2 = 1$, and $w = U(0.5, 1)$. Total number of function evaluations is

1,000,000. A PC computer with Windows XP and C++ Builder Compiler, Pentium-4 processor at 3.00GHz, 1.00 GB of RAM was used for all experiments. A well-know problem set proposed by (Solomon, 1987) is used to test our model. In these problems, the travel times are equal to the corresponding Euclidean distances. One problems is chosen for the experiment: problem C101 in dimensions 25.

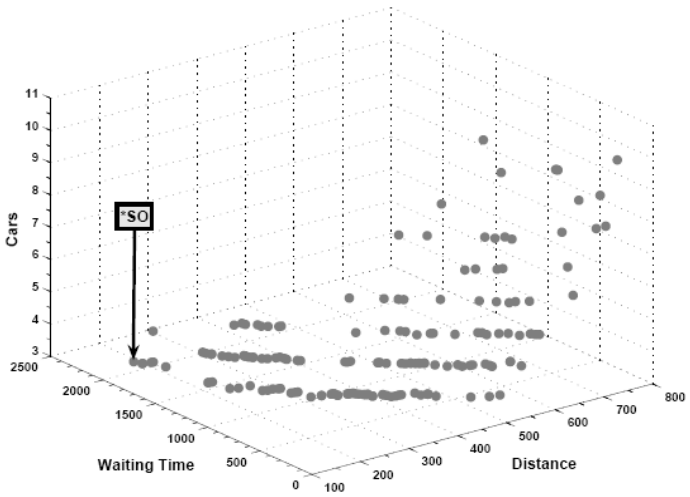


Fig. 10. Pareto front for problem C101 with 25 customers

Figure 10 shows the Pareto front obtained by 10 runs of MO-PSO for the problem C101 with 25 customers. The optimal route obtained by single objective approaches is marked with ***SO** in the figure. The Pareto front obtained by MO-PSO evidenced the huge waiting time accumulated by the optimal solution reported in the literature with single objective (total distance) approaches. The vector reports a total distance of 191.813620 units attained with only 3 cars, and a total waiting time of 2133.144826 units. As we mentioned above, in this problem, the travel times are equal to the corresponding Euclidean distances. Then, the total waiting time is at least 10 times greater than the total travel time performed for all routes. The total waiting time represents the 57.53% of the total available time (3708 units) of the 3 vehicles, if the depot's window time [0; 1236] is taken as reference. Before, we explained that for eliminating any unnecessary waiting time, we assume that all routes start just-in-time $w_k = 0$, that is, we adjust the depot departure time of each vehicle $t_k = \max(0, a_i - t_{0i})$, $i \varepsilon N$. Therefore, there is no waiting times at the beginning of the route.

Table 2 presents some solutions of the Pareto front found by MO-PSO for the problem C101 with 25 costumers. For example, the solution vector (215.703725, 1584.387467, 4) presents a significant reduction in the total waiting time, but, the total distance and number of vehicles increased respect the solution vector (191.813620, 2133.144826, 3). The solution vector (359.85828, 963.12288, 3) has a waiting time of only 25.9% of the total available time (3708 units) of the three vehicles. Finally, the last solution vector (748.901755, 100.252129,10) presents a reduction of 20 times the waiting time of the solution vector (191.813620, 2133.144826, 3), and it represents the 2.7% of the total available time. Although this solution vector increase at least 3 times the total distance and the number of vehicles of the solution vector (191.813620, 2133.144826, 3).