A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks

Sistema de Colonia de Hormigas Autoadaptativo para el Problema de Direccionamiento de Consultas Semánticas en Redes P2P

Claudia Gómez Santillán[1,2], **Laura Cruz Reyes**[2], **Eustorgio Meza Conde**[1], **Elisa Schaeffer**[3] **and Guadalupe Castilla Valdez**[2]

[1]Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada del Instituto Politécnico Nacional (CICATA- IPN). Carretera Tampico-Puerto Industrial Altamira, Km.14.5. Altamira,Tamps.,México.

[2]Instituto Tecnológico de Ciudad Madero (ITCM). 1ro. de Mayo y Sor Juana I. de la Cruz s/n CP. 89440, Tamaulipas, México.

[3] Facultad de Ingeniería Mecánica y Eléctrica (FIME-UANL), Avenida Universidad s/n. Cd. Universitaria, CP. 66450, San Nicolás de los Garza, N.L. México.

cggs71@hotmail.com, lcruzreyes@prodigy.net.mx, emezac@ipn.mx, elisa@yalma.fime.uanl.mx, gpe_cas@yaoo.com.mx.

Article received on July 17, 2009; accepted on November 05, 2009

Abstract

In this paper, we present a new algorithm to route text queries within a P2P network, called Neighboring-Ant Search (NAS) algorithm. The algorithm is based on the Ant Colony System metaheuristic and the SemAnt algorithm. More so, NAS is hybridized with local environment strategies of learning, characterization, and exploration. Two Learning Rules (LR) are used to learn from past performance, these rules are modified by three new Learning Functions (LF). A Degree-Dispersion-Coefficient (DDC) as a local topological metric is used for the structural characterization. A variant of the well-known one-step Lookahead exploration is used to search the nearby environment. These local strategies make NAS self-adaptive and improve the performance of the distributed search. Our results show the contribution of each proposed strategy to the performance of the NAS algorithm. The results reveal that NAS algorithm outperforms methods proposed in the literature, such as Random-Walk and SemAnt.

Keywords: Search Process, Internet, Complex Network, Ant Colony System, Local Environment, Neighbor.

Resumen

En este documento, proponemos un nuevo algoritmo para ruteo de consultas textuales dentro de una red P2P, llamado Neighboring-Ant Search (NAS). El algoritmo está basado en la metaheurística Ant Colony System (ACS) y el algoritmo SemAnt. Además, NAS está hibridizado con estrategias del ambiente local de aprendizaje, caracterización y exploración. Dos reglas de aprendizaje (LR) son usadas para aprender del rendimiento pasado, esas reglas son modificadas por tres Funciones de Aprendizaje (LF). Un Coeficiente de Dispersión del Grado (DDC) es usado como una métrica topológica local para la caracterización estructural. Una adaptación del bien conocido método de exploración de adelanto (one-step Lookahead) es usado para explorar el ambiente cercano. Estas estrategias locales proveen a NAS una capacidad auto-adaptativa que mejora el rendimiento de la búsqueda distribuida. Los resultados experimentales muestran la contribución de cada estrategia propuesta para el rendimiento del algoritmo NAS. Estos resultados revelan que el algoritmo NAS obtiene mejores resultados que los algoritmos propuestos en la literatura existente tales como Random-Walk y SemAnt.

Palabras Clave: Proceso de Búsqueda, Internet, Redes Complejas, Sistema de Colonia de Hormigas, Ambiente Local, Vecindad.

1 Introduction

The popularity of peer-to-peer (P2P) systems is motivated by the benefits offered to the end user. In contrast to the traditional Web, a P2P system does not need to rely on any dedicated centralized servers, which makes P2P networks reliable and fault tolerant. Hence a user can easily join a network and leave when necessary, giving rise to

unstructured self-organizing networks. Due to the unstructured nature, these applications often employ a flooding-based data search mechanism, which generates severe communication overhead and limits the growth of P2P systems. These systems together with the underlying Internet are considered complex dynamic distributed networks for their size and constantly evolving interconnectivity. In complex dynamic distributed networks, global knowledge collection is not a feasible approach to handle queries on shared resources. In these circumstances, each query needs to determine locally its behavior, without resorting to a global control mechanism.

Digital technologies and new standards make it possible to produce music, movies, pictures, images, and textual information in a digital form with reasonable quality. Internet is the essential, cheapest and most convenient way to manage digital files, to sell, buy, and share digital content. Such a popular application like the World Wide Web (WWW) has not been convenient enough to share files. Sharing content on the WWW requires infrastructure (a HTTP server) and makes it difficult for individual users to share their files in an easy and independent way. The files published on the WWW are available for search only after their respective sites are crawled and indexed by existing centralized search engines. Since such operations may take a significant amount of time, users have no direct control over the published files to make them available for immediate search. These disadvantages make the use of traditional applications for file sharing complicated [13].

In 1999 the P2P systems arose as a response to the increased demand for file sharing. These systems are formed by interconnected peers that offer their resources to other peers within the network. The participants connect and disconnect constantly, producing changes in the structure of the network. Due to the unstructured nature, applications mainly employ flooding-based data search mechanisms. Flooding-based search generates vast amounts of Internet traffic that limits the growth of peer-to-peer systems. The obvious problems that appeared with the growing popularity of peer-to-peer file sharing systems are two: a) the poor accuracy of the information search and b) the traffic caused by the flooding-based search. Measurements have shown that peer-to-peer systems are the main source of Internet traffic [13],[10],[11], making the development of new approaches to avoid flooding an important research challenge.

The Semantic Query Routing Problem consists in each peer deciding, based on a keyword in the query, to which neighboring peer to resend the text query. To avoid flooding, the goal is to maximize the number and the quality of query results, while minimizing the use of the resources of the network. Existing approaches for query routing in P2P networks range from simple broadcasting techniques to sophisticated methods [13],[10],[11]. Due to the fact that P2P networks are based on non-central authorities and high-growing dimension, the challenge for query routing is the development of methods that adapt themselves to dynamic environments. Such intelligent adaptation must be based only on the local knowledge of each peer. Among the intelligent mechanisms successfully applied to several problems in distributed systems, lie the ant-colony methods. The metaheuristic of Ant Colony System, proposed by Dorigo [12], solves optimization problems based on graphs. Many ant algorithms have been specifically designed for handling routing tables in telecommunications. However, there are very few ant algorithms for handling routing tables in the Semantic Query Routing Problem [10].

In this paper, we present a novel algorithm for distributed text query routing. The algorithm, called the Neighboring-Ant Search (NAS), is based on two well-known ant algorithms: the Ant Colony System [7] and the SemAnt [10]. Additionally, NAS is hybridized with local strategies of learning, characterization, and exploration. Three functions are employed to learn from past performance. The first function is used to evaluate the NAS performance based on the found and expected results. The second function qualifies the NAS performance based on the available time for the searching. The third function qualifies each peer depending on the distance towards a previously found resource. A topological metric based on the number of connections of each peer is used for the structural characterization. An adaptation of the well-known one-step Lookahead search is used to explore the neighbor peers of the nearby environment [11]. These three strategies contribute with the main goal of the application which is to find a greater amount of resources in the least amount of time.

2 Background

In order to place the research in context, this section is divided in four parts. The first part defines, explains, and models the peer-to-peer complex networks. The second part explains and formally defines the Semantic Query Routing Problem. The third part explains and defines the Lookahead exploration and the Degree-Dispersion-Coefficient local metric. The last part refers to the Random-Walk algorithm.

2.1 Peer-to-Peer Complex Networks

P2P systems are formed by interconnected peers that offer their resources to other peers within the network. Hence a P2P network is a distributed system that can be modeled as a *graph*. Each peer in the network is represented by a *node* (also called a *vertex*) of the graph. The interactions among the peers are represented by the *connections* (also called the *edges*) of the graph. In P2P networks, the nodes are capable of self-organization for the purpose of sharing resources, without requiring the mediation or support of a server or centralized authority [2].

More so a P2P system, together with the underlying communication network (typically Internet), forms a complex system that requires autonomous operation through mechanisms of intelligent search [10]. Amaral [1] published a classification for different types of systems and categorized them into simple, complicated, and complex systems. *Complex systems* are those systems that have (typically) a very large number of components, the connections among them may evolve over time, and the roles of the components may vary. In many studies, complex systems are modeled as networks, giving rise to the concept of *complex networks* and, within this context, the term P2P Complex Networks.

One of the main motivations for modeling systems as P2P complex networks is the flexibility and generality of the abstract representation that allows handling properties such as dynamic topology in a natural way. A recent methodology for modeling complex systems, called Autonomous Oriented Computing (AOC), was proposed by Liu [12]. AOC consists in the formulation of a tuple that represents the general model of the system, i.e.: $<\{e_1, e_2,\ldots,e_i,\ldots,e_N\}, E,\Phi\}>$, where $\{e_1,e_2,\ldots,e_i,\ldots,e_N\}$ is a subset of size N of autonomous *entities*, E is the *environment* in which the entities reside, and Φ is the *objective function* of the global system. Each entity is a basic element with a well-defined goal within the complex system. To achieve its goal, it has attributes that describe its behavior rules, current state, and an evaluation function. We use the AOC notation to model P2P systems as follows: *i*) the entities are the *agents* that surf in the network with the objective of finding resources; *ii*) the environment is the P2P network and *iii*) the objective function is to find the maximal set of resources in the shortest possible time. A more detailed description of our querying system based on intelligent agents is given in Section 4.

2.2 The Semantic Query Routing Problem

The problem of locating textual information in a P2P network over the Internet is known as *Semantic Query Routing Problem* (SQRP). The goal of SQRP is to determine the shortest paths from a node that issues a query to nodes that can appropriately answer it (by providing the requested information). The query traverses the network moving from the initiating node to a neighboring node and then to a neighbor of the neighbor and so forth until it locates the requested resource (or gives up in its absence). This type of propagation is known as *flooding* and it is the most common search strategy in P2P networks. Algorithms for SQRP must consider several factors, ranging from hardware and software characteristics to user behavior. Due to its complexity [10],[1],[12],[6], solutions proposed to SQRP typically limit to special cases. Yang et al. [6] propose AntSearch that controls the quantity of flooding using a simple learning technique, whereas Michlmayr [10] proposes the SemAnt algorithm for learning from user behavior.

Formally, SQRP is defined with the description of an *Instance* and an *Objective* that must be satisfied by a solution algorithm such as the ant-based algorithm proposed in this work. ***Instance***: given a P2P network represented by a graph T, a set of *contents* distributed in the nodes called repositories R, and a set of semantic *queries* Q launched by the nodes. Each query can be launched from any node in the time T_0, $\forall T_0 \in \mathbf{Z}$, assuming a discrete-time process. The node that originally launches a query (or receives a query from other node) in time T_0+i, $\forall i \in \mathbf{Z}^+ \cup \{0\}$, can locally process the query and/or forward a copy of the query to a set of nearby nodes at time $T_0+(i+1)$. The query processing finishes when a stop condition has been satisfied, whether either the maximal quantity of

resources has been found or th time-to-live value specified for the query is reached. ***Objective***: find a set of paths among the nodes launching the queries and the nodes containing the resources, such that the quantity of found resources is maximized and the quantity of steps given to find the resources is minimized.

2.3 Lookahead and the Degree-Dispersion-Coefficient

A node *i* is a *neighbor* of a node *j* if the two nodes *i* and *j* are connected by an edge (*i*, *j*) in the graph that models the system. The set of all neighbors of a node *i* is denoted by $\Gamma(i)$. In an undirected simple graph, that is, a graph in which the edges are considered bidirectional communication channels and each pair of nodes may be connected by at most one edge, the *degree* of a node is the number of neighbors it has.

A well-known strategy based on local information is the one-step *Lookahead* exploration method [11]. Lookahead is employed in algorithms to examine neighboring resources up to a certain level before deciding how to proceed with the search. In this work, we assume that each node knows the resources of the first-level neighbors.

In order to locally focus the exploration strategy we use the *Degree-Dispersion-Coefficient* (DDC) function [14]. DDC is based on local information that through the dispersion of the degree of a node measures the differences between the degree of a node and the degrees of its neighbors, this is:

$$DDC(i) = \frac{\sigma(i)}{\mu(i)}, \qquad (1)$$

Where the degree variation of the set $\{i \cup \Gamma(i)\}$ $\sigma(i) = \sqrt{\frac{\sum_{j \in \Gamma(i)} \left[k_j - \mu_i\right]^2 + \left[k_i - \mu_i\right]^2}{N_i}}$, the average degree found in the set $\{i \cup \Gamma(i)\}$ $\mu(i) = \frac{\sum_{j \in \Gamma(i)} \left[k_j\right] + k_i}{N_i}$, N_i is the number of nodes in $\{i \cup \Gamma(i)\}$, and, k_i and k_j are the degree of nodes *i* and *j* respectively.

2.4 Random-Walk Algorithm

The Random-Walk (RW) search algorithm is a *blind* search technique where the nodes of the network possess no information on the location or contents of the requested resource, unless the resource resides in the node itself . Let *G* be a graph that models the network and *v* a node in *G*. A *T-hop Random-Walk* from *v* in *G* is a sequence of dependent random variables $X_0,\ldots, X_T$ defined as follows: $X_0 = v$ with probability 1 and for each $i = 1,\ldots,T$, the value for X_i is selected uniformly at random among the nodes in $\Gamma(X_{i-1})$, that is, among the neighbors of the node of the preceding step. In others words, a Random-Walk begins at a node and on each step moves to a neighbor of the current node, until it arrives to a node that meets the goal. In our P2P model, the goal is met when a node contains the requested resource [3]. Optionally, one could include the DCC function into the Random-Walk algorithm. A simple modification to include such a structural preferentiality is to choose uniformly at random two neighbors, calculate their DCC values, and move on to the neighbor with higher DCC.

3 Related Work

The success of SQR algorithms in P2P file-sharing networks lies on search mechanisms that have received special attention [10][6]. We summarize here some of the most relevant proposals for semantic query routing using Ant-Colony Algorithms. These algorithms are based mainly on a learning structure named *pheromone*. This structure is used for establishing indirect communication between ants about their past performance. The pheromone table (τ) is used as a query routing table.

Michlmayr [10] proposes a distributed SQRP algorithm for P2P networks called SemAnt and includes an evaluation of the parameter configuration that affects the performance of the algorithm. Michlmayr aims for an

optimal ratio between network traffic and quantity of results and compares the performance of SemAnt with the Random-Walk algorithm using the following metrics: *i*) *Resource Usage* define as the number of links traveled for each query within a given period of time, *ii*) *Hit Rate* defines as the number of resource found for each query within a given period of time, and *iii*) *Efficiency* defined as the ratio of resource usage to hit rate. Dividing the number of links traveled by the number of resources found, gives the average number of links traveled to find one resource.

Yang et al. [6] propose an algorithm called *AntSearch* for non-structured P2P networks. In AntSearch, each pair of nodes stores information on the level of success of past queries as well as on the pheromone levels of the immediate neighbors. The work of Yang et al. was motivated by the need to improve search performance in terms of the traffic in the network and the level of information recovery. Yang et al. use three metrics to measure the performance of the AntSearch. One is the *number of searched files* for a query with a required number of results, given *N as*: a good search algorithm should retrieve the number of results over but close to *N*. The second one is the *per result costs* that defines the total amount of query messages divided by the number of searched results; this metric measures how many average query messages are generated to gain a result. Finally, *search latency* is defined as the total time taken by the algorithm.

Di Caro and Dorigo [4] propose *AntNet* that is designed for packet-switched networks. The ants collaborate in building routing tables that adapt to current traffic in the network, with the aim of optimizing the performance of the entire network. AntNet uses global information on the nodes of the network in order to choose the destination nodes. Di Caro and Dorigo focused on standard metrics for performance evaluation, considering only sessions with equal costs, benefits and priority and without the possibility of requests for special services like real-time. In AntNet framework, the main measures are: *i*) *throughput* correctly delivered bits/sec, *ii*) *delay distribution for data packets* (sec), and *iii*) *network capacity usage* for data and routing packets, expressed as the sum of the used link capacities divided by the total available link capacity.

Most relevant aspects of former works have been incorporated into the proposed NAS algorithm. The framework of AntNet algorithm is modified to correspond to the problem conditions: in AntNet the final addresses are known, while NAS algorithm does not know a priori the nodes where the resources are located. On the other hand and different to AntSearch, the SemAnt algorithm and NAS are focused on the same problem conditions, and both use algorithms based on AntNet algorithm. However, the difference between the SemAnt and NAS is that SemAnt only learns from past experience, whereas NAS takes advantage of the local environment. This means that the search in NAS takes place in terms of the classic local exploration method of Lookahead [11], the local structural metric DDC, and three local functions of the past algorithm performance. These three performance functions are described in the following section.

4 Classical Learning Rules Modified by the Proposed Learning Functions

The classic ACS algorithm is formed by two rules – selection and update – that allow the convergence of the system towards better results. Modifications of these rules were made with the goal of adapting the ACS algorithm to the SQRP. Also, new functions were added such as the DDC topological metric, the Lookahead method, and three Learning Functions: hit importance, time-to-live of the agent, and distance towards a resource that improve the performance of the system in terms of the objective of the problem. The additions of these functions improved the system performance and are explained in detail in this section.

4.1 Learning Functions

This section describes the new proposed Learning Functions (LF) and so is divided into three parts. The first function defines the hit importance, the second function defines the time-to-live importance, and the last function defines the distance importance. With these three LF, introduced in the classical learning rules, the agents search the resources.

Hit Importance Function (HIT), shown in Eq. (2), qualifies the performance of the search agent *k* (that is, an ant that represents a query), based on the found result ($results_k$), and the expected result (*maxResults*):

$$HIT = \frac{results_k}{maxResults}, \quad (2)$$

where $results_k$ represents the amount of results found by the ant k and *maxResults* is the amount of results requested. The value of the *HIT* function is applied in the global updating function, Eq. (8), with the purpose of guiding the queries toward routes that provide the greatest possible amount of found resources. This *HIT* value is registered in each node of the path traversed by the search ant, as shown in Figure 1.

Importance of Time-to-live Function (ITL_HOP), qualifies the performance of the search agent *k*, based on the time-to-live used to find one resource (TTL_k) and the maximum time-to-live originally assigned:

$$ITL\_HOP = \frac{maxTTL}{2 \times TTL_k}, \quad (3)$$

where TTL_k is the partial time-to-live of the ant k until the moment, and *maxTTL* is the maximum time-to-live assigned to an ant for a query. These time measures are given in terms of the number of hops and correspond respectively with the given steps and the maximum steps allowed to each ant. The result is applied into the global updating rule, Eq. (8), with the overall goal of decreasing the time-to-live necessary to find a set of resources. The *ITL_HOP* value is registered in each node of the path traversed by the ant until the node with the last resource is found, as shown in Figure 2.

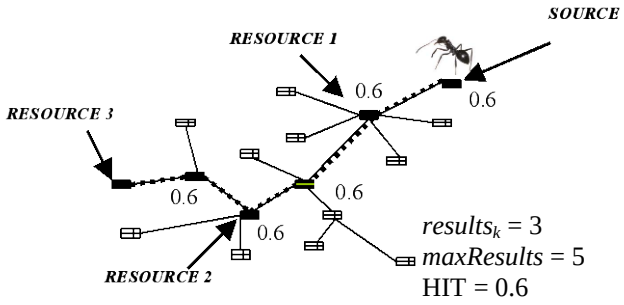


Fig. 1. Example of the calculation of the HIT function

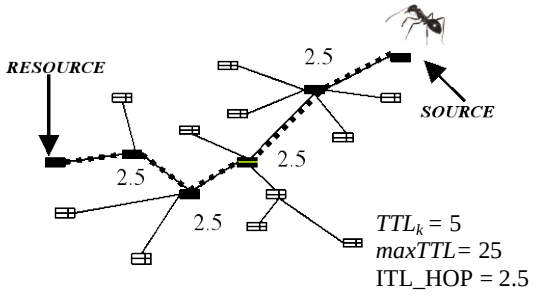


Fig. 2. Example of the calculation of the ITL_HOP function

Importance of Distance Function (ID_HOP), for the agent *k*, shown in Eq. (4), qualifies each node *s*, which is a neighbor of the current node *r*, depending on the distance towards a previously found resource *t*:

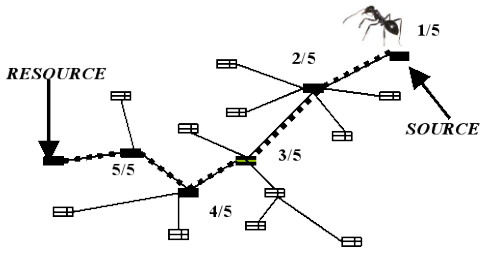
$$ID\_HOP_{r,s,t} = \left(\frac{h_k}{h_{r,s,t}} \right)^{-1} \quad \forall r, s \in route \text{ of } k \quad (4)$$

when r is the current node, s is the evaluated node, t is the found resource, k is the ant agent, h_k is the total number of steps taken by the agent, and $h_{r,s,t}$ is the number of hops from the source to an evaluated resource node. Once an ant k generates a route to a resource t, the function $ID\_HOP_{r,s,t}$ is applied to each node belonging to this route to increase its importance in terms of the distance to a found resource node with respect to the length of the route. This function is obtained through the inverse of the total number of hops h_k made by the ant on the route to the resource found,

divided by the relative number of hops $h_{r,s,t}$ from the source to the node evaluated *s*, which is a neighbor of the current node *r*, as is shown in Figure 3.

4.2 Classical Learning Rules (LR)

This section describes the modifications made to the two classical learning rules, originally proposed by Dorigo [7] in the ACS algorithm. The first rule is called *state transition;* with this rule the next node to be visited is selected. The transition rule uses two strategies: *exploitation* and *exploration*. The second rule is called *updating*; with this rule the ant updates the nodes in the traversed path. The updating rule uses two strategies: local and global updating of the pheromone. All strategies are used to feedback the system on successful routes.



$h_k = 5$
$h_{r,s,t}$= 1, for *r*=1 and *s* =2
$h_{r,s,t}$= 3, for *r*=3 and *s* =4
ID_HOP$_{r,s,t}$ = 1/5, for *r*=1 and *s* =2
ID_HOP$_{r,s,t}$ = 3/5, for *r*=3 and *s* =4

Fig. 3. Example of the calculation of the *ID_HOP* function

The modified *state transition* rule of NAS is formulated by Eqs. (5) and (6). This:

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right] \cdot \left[DDC_{L_n} + ID\_HOP_{r,L_n,t} \right]^{\beta} \right\}, & \text{if } q \le q_0 \text{ (exploitation)} \\ S, & \text{otherwise (biased exploration),} \end{cases} \quad (5)$$

where *r* is the current node where the ant *k* is located, *u* belongs to the set of neighboring nodes of *r*, V_k is the set of nodes visited by the ant *k*, τ is the pheromone table, *t* is the searched resource, β is the parameter that determines the relative importance between the pheromone and the DDC with *ID_HOP*, *q* is a random number, and q_0 determines the relative importance of exploitation versus exploration. In case that $q \le q_0$, the exploitation strategy is selected: it selects a node that provides a greater amount of pheromone and better connectivity with smaller numbers of hops toward a resource. Otherwise the exploration strategy, Eq. (6), is selected:

$$S = f(p_{r,u,t}), \quad p_{r,u,t} = \frac{\left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta}}{\sum_{\forall i \in \Gamma(r) \wedge i \notin V_k} \left[\tau_{r,i,t} \right] \cdot \left[DDC_i + ID\_HOP_{r,i,t} \right]^{\beta}}, \quad (6)$$

where *S* selects a node applying a random selection function *f* based on the well-known roulette-wheel selection to favor nodes with higher connectivity, stronger pheromone trail, and a shorter distance to a requested resource. This exploration strategy stimulates the ants to search for new paths. Note that the pseudorandom variable is modified by the DDC topological metric and the *ID_HOP* learning function.

The ACS *updating* rule is composed of both local and global updating. Hence the modified *updating* rule in this work is given in Eqs. (7) and (8). The *local update* strategy of NAS is formulated by:

$$\tau_{r,s,t} \leftarrow (1-\rho) \cdot \tau_{r,s,t} + \rho \cdot \tau_0, \quad (7)$$

where *r* is the current node where the ant *k* is located, *s* is the current neighboring node where the ant is going to move, *t* is the searched resource, τ is the pheromone table where the ant does the local updating, τ_0 is the initialization value of the pheromone, and ρ is the local evaporation factor of the pheromone. Each time an ant decides to move towards a node by the state transition rule, some pheromone is deposited at each node that has been visited to establish a trend towards the most frequently visited nodes.
On the other hand, the modified *global update* strategy, given by Eq. (8), is computed each time that a resource is found and is applied to each node belonging to the route that lead to the discovery of the resource:

$$\tau_{r,s,t} \leftarrow (1-\alpha)\cdot\tau_{r,s,t} + \alpha\cdot\left[w\cdot HIT + (1-w)\cdot ITL\_HOP\right] \quad \forall r,s \in path \text{ of } k, \tag{8}$$

where α is the global evaporation factor of the pheromone, and *w* is a weight factor that controls the relative importance between the resource found (*HIT* function) and the time-to-live (*ITL_HOP* function). The amount of pheromone deposited depends on the quality of the solution obtained, the amount of resources found, and the time-to-live of the ant at time of discovery.

5 Multi-agent Architecture and Parallel Pseudocode

In this section we describe the NAS algorithm for the Semantic Querying Routing Problem. We first present an agent-based architecture and then a parallel pseudocode.

Multi-Agent System Architecture. The overall system architecture is shown in Figure 4. It comprises two elements: *i*) the *environment* (*E*) which is a static P2P complex network, with a probability distribution for the network topology (*T*), understood as a set of linked nodes with local information about their neighboring nodes, *ii*) the *agents* {{e_1}, {e_2}, {e_3}} that can be of three kinds, depending on their role: query, search, and retrieval. In the NAS algorithm the agents are represented as *ants*, each agent is either an ant that carries a query (*Q*) or a node of the network that launches the query; each node also has its own repository (*R*).

The *query agents* {e_1} represent stationary ants located in the nodes that launch queries and their role is to create *search agents*. The *search agents* {e_2} represent ants born in a node that launches a query. Their role is to move through network following a set of rules. These agents operate through the state transition that chooses between exploratory or exploitatory movements through Eqs. (5) and (6). Each time that an action is activated the local update module conducts local evaporation from the pheromones table through Eq. (7). In order to evaluate its performance, an agent records the routes that have been selected, and each time it finds a resource, it creates a *retrieval agent*. The Lookahead strategy verifies if a resource exists in the current node or in its neighborhood. The time-to-live of a search agent is set at *maxTTL*, but the agent could also ceases to operate upon reaching the expected amount of results *maxResult*. All modules rely on control parameters that must be configured properly to ensure good performance of the system. The *retrieval agents* {e_3} are ant created whenever a result is found in a node. These agents are responsible for evaluating the performance of the search agents and updating with feedback the pheromone table. With this feedback, done through Eq. 8, the route that was traversed by the search agent up the resource is updated on the pheromone table and returned to the end user.

NAS Algorithm Parallel Pseudocode, is a metaheuristic algorithm, where a set of independent agents called ants cooperate indirectly and sporadically to achieve a common goal. The algorithm has two objectives: it seeks to maximize the number of resources found by the ants and to minimize the number of steps taken by the ants. NAS guides the queries toward nodes that have better connectivity using the local structural metric DDC [14]. Since the DDC, in order to minimize the hop count, measures the differences between the degree of a node and the degrees of its neighbors, the more frequent that a query is carried towards a resource a better path is selected. This is, the rate of optimization of a query depends directly on its popularity.

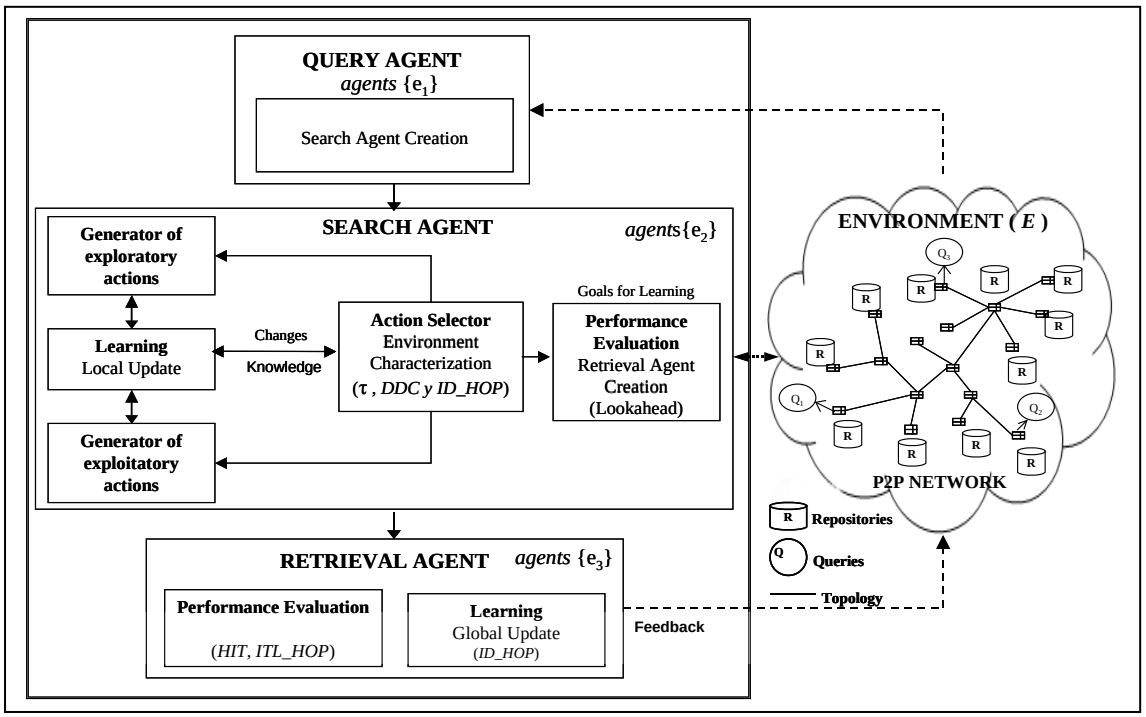


Fig. 4. NAS Architecture

Table 1. NAS algorithm pseudo code

01	**parallel** // Concurrent activity of **query agents**
02	**for each** query in r_k create a search agent k with $TTL_k = maxTTL$ and $Hits_k=0$
03	**while** $Hits_k < maxResults$ and $TTL_k > 0$ // Concurrent activity of **search agents**
04	// Phase 1: The evaluation of results
05	**if** the unvisited $s_k \in \{ r_k \cup \Gamma(r_k)\}$ has the searched resource // Lookahead strategy
06	r_k = append s_k to $path_k$
07	$Hits_k = Hits_k + 1$
09	Local Pheromone Update (Eq. 7)
10	Global Pheromone Update (Eq. 8) // Concurrent activity of **retrieval agents**
11	**else** // Phase 2: The state transition
12	**if** r_k is a leaf node or does not have an unvisited neighbor ,
14	remove the last node from $path_k$
15	**else**
	s_k = apply the *transition* rule with the DDC function (Eqs. 5 and 6)
16	r_k = append s_k to $path_k$
18	Local Pheromone Update (Eq. 7)
	$TTL_k = TTL_k - 1$
19	kill the search agent

The NAS algorithm performs in parallel all the queries using **query agents**. The role of each query agent is to create a **search agent** when a query is launched in a corresponding node. The activity of each search agent consists of two main phases. The first phase, the **evaluation of results** (lines 04-10 of the pseudocode in Table 1) implements the classical Lookahead technique. That is, an ant k, called search agent and located in a node r_k, verifies if the

resource exists in an unvisited node s_k that belongs to its neighborhood, including itself. If the resource is found, the ant adds the node s_k to its path, updates the number of occurrences of the queried resource $Hits_k$, reduces the ant time TTL_k by one hop, performs the local pheromone update according to Eq. (7), and performs the global pheromone update according to Eq. (8). The global updating of the pheromone is a concurrent activity of the ants called **retrieval agents**; the route to the resource is updated on the pheromone table and returned to the end user. In the case that the evaluation phase fails, a second phase, the **state transition** (lines 11-18 of the pseudocode in Table 1) is carried out. This phase selects through random number q, Eq. (5), a neighbor node s. In the case that there is no new node towards which to move, that is to say, the node is a leaf or all neighbor nodes have been visited, a hop backwards is carried out on the path. Otherwise the ant adds the selected node s_k to its path, updates locally the pheromone, Eq. (7), and reduces TTL_k by one hop. The query process ends when the expected number of results has been found or TTL_k reaches zero. In both cases the search agent is killed, indicating the end of the query.

6 Experimental Analysis of the NAS Algorithm

In this section, we describe two experiments carried out on the NAS algorithm. The objective of the first experiment is study the performance of NAS in comparison with algorithms proposed in the literature, SemAnt and Random-Walk. The objective of the second experiment is to examine the contribution of each local environment strategy to the performance of the NAS algorithm.

6.1 Experiments Setup

The NAS algorithm was implemented to solve SQRP instances. The application of the NAS algorithm requires the specification of the problem instance to solve and the definition of the control parameters of the algorithm. In our implementation, an SQRP instance is determinated by three separate files: topology, repositories, and queries. We generated the experimental instances as follows.

The generation of the *topology* (*T*) is based on the method of Barabási et al. [4] to create a non-uniform network with a *scale-free distribution*. In the scale-free or *power law* distribution, a reduced set of nodes has a very high degree and the rest of the nodes have a small degree. All networks generated have 1,024 nodes and bi-directional edges. The number of nodes was selected based on recommendations by Michlmayr [10] and Di Caro [6].

In the P2P model, each peer manages a *local repository* (*R*) of resources and offers its resources to other peers. We generated these repositories using "topics" obtained from ACM Computing Classification System taxonomy (ACMCCS). This database contains a total of 910 distinct topics. Also the content distribution is a power law: few nodes contain many topics in their repositories and the rest of the nodes contain few topics.

For the generation of the *queries* (*Q*), each node was assigned a list of possible topics to search. This list is limited by the total amount of topics of the ACMCCS. During each step of the experiment, each node has a probability of 0.1 to launch a query, selecting the topic uniformly at random within the list of possible topics of the node. The probability distribution of *Q* determines how often the query will be repeated in the network. When the distribution is uniform, each query is duplicated 100 times on average.

The topology and the repositories were created static, whereas the queries were launched randomly during the simulation. Each simulation was run for 20,000 time units (queries). The average performance was studied by computing three performance measures each 100 units of time:
Average hops, defined as the average amount of links travelled by a search agent until its death (that is, reaching either *maxResults* = 10 or *maxTTL* = 25). *Average hit-rate*, defined as the average number of resources found by each search agent until its death and *Average efficiency*, defined as the *average hit-rate* divided by the *average hops*.

The control parameters of the algorithm are specified in a file containing a *global static configuration* of the NAS algorithm parameters. The configuration of the NAS algorithm used in the experimentation is shown in Table 2. In the first column is the parameter value and the in second column is given a description of the parameter. These parameter values were based on recommendations done by Michlmayr [10] and Dorigo [7].

6.2 Comparative study of the NAS algorithm

In this experiment, the performance of the NAS algorithm is compared against the SemAnt [10] and Random-Walk [3] algorithms. For the experimentation with three algorithms we use the general specifications described in Section 6.1.

To carry out the experiment with the SemAnt algorithm with the same conditions, we only changed the parameter *number of links*. For SemAnt [10], the number of connections range between 4,000 and 10,000 which does not affect the performance of the algorithm. Hence, we fixed the number of the links to 7,000 links, choosing an intermediate value in that range.

Table 2. Configuration parameter of the NAS algorithm

Parameter	Definition
ρ= 0.07	Local pheromone evaporation factor
α= 0.07	Global pheromone evaporation factor
τ_0 = 0.009	Pheromone table initialization
ID_HOP_0 = 0.001	Initial value of the table of distances to previously encountered resources
β = 2	Relative importance of DDC and *ID_HOP* with respect to the pheromone
q_0 = 0.9	Relative importance between exploration and exploitation
maxResults =10	Maximum number of results to retrieve
maxTTL = 15	Time-to-live of the search agents
w = 0.5	Relative importance of the resources found and the time-to-live

The experimental values for the SemAnt algorithm were obtained from [10]. For the *average hit-rate* variable the SemAnt algorithm shows values from 0.8 to 2.1 hits per query. However, for the NAS algorithm, the average hit-rate ranges from 9.5 to 10 hits per query. On the other hand, the *average hop* count of the SemAnt algorithm starts out at 23 and lowers to 16 hops per query during the operation of the algorithm, whereas the NAS algorithm, starts at the average hop of 12.5 and then diminishing to 12 hops per query. Finally, the *average efficiency* of the SemAnt algorithm improves from 0.034 to 0.13 hits per hop, while for the NAS algorithm, as shown in Figure 5, it increases from an initial value of 0.76 up to 0.84 hits per hop.

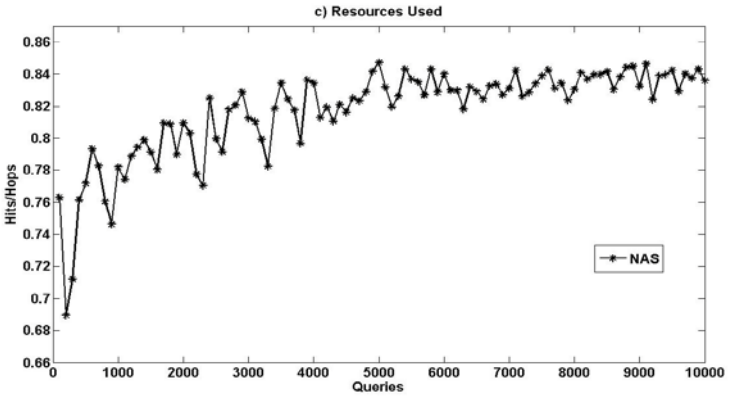


Fig. 5. Performance of NAS Algorithm *average efficiency*

For the experiment with the Random-Walk (RW) algorithm, we also used the general specifications described in Section 6.1. Figure 6(a) shows the *average hit-rate* for both RW and NAS algorithms. The behavior changes slightly over time. That is, the average hit-rate in RW varies between 1 to 0.5 hits per query, while in NAS, the average hit-rate varies between 9.6 to 10 hits per query. Similarly Figure 6(b) shows the *average hop* count for both RW and

NAS algorithms. The behavior of the RW does not evolve; the average hop count keeps around 15 hops per query, from the beginning to the end. However in NAS the behavior evolves, starting at 12.5 and lowering down to 12 hops on average. Finally, Figure 6(c) shows the *average efficiency* for both RW and NAS algorithms. In the RW algorithm, the behavior does not evolve; the average efficiency keeps around 0.5 hits per hop. For NAS the behavior does evolve; so that the average efficiency increases from 0.76 to 0.84 hits per hop.

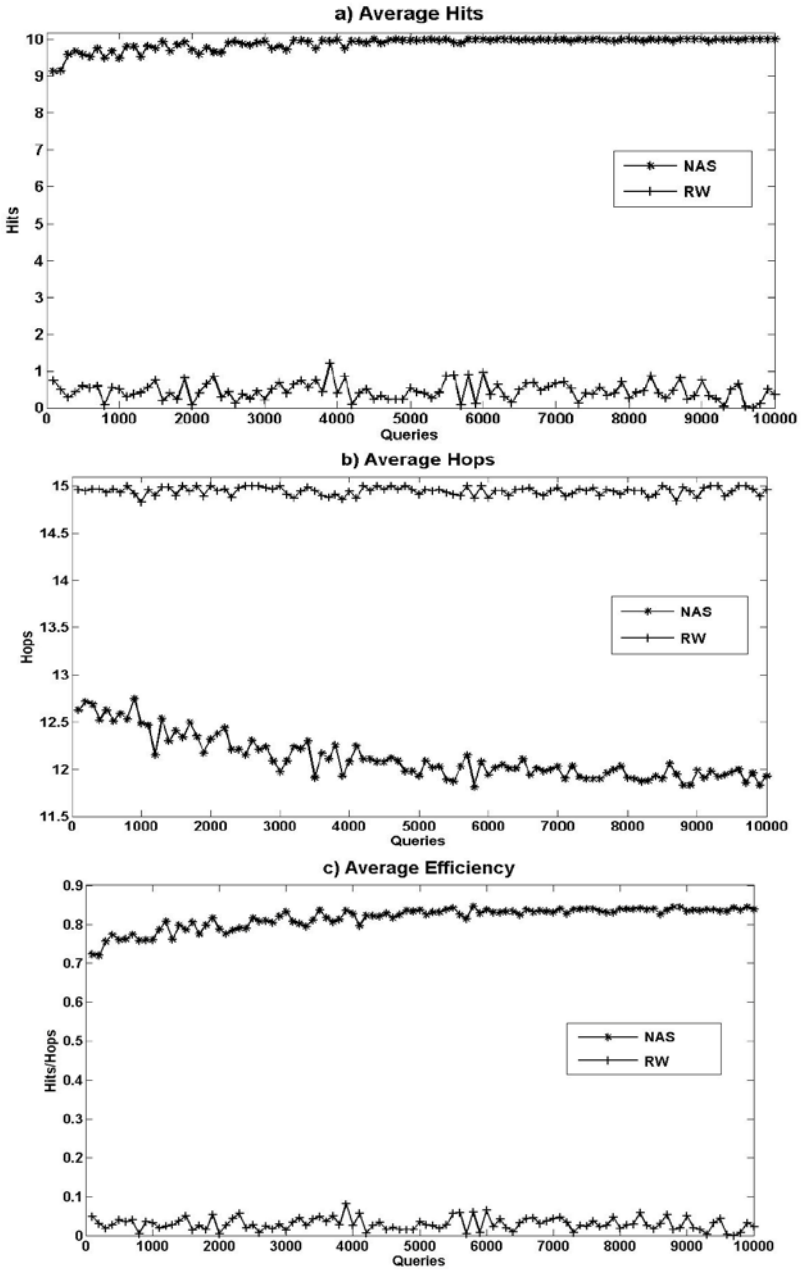


Fig. 6. Comparison of NAS Algorithm against Random-Walk Algorithm. a) *average hits-rate*, b) *Average hops* and c) *average efficiency*

6.3 Experiments for the Analysis of the Local Environment Strategies of NAS Algorithm

In this experiment, the performance of the NAS algorithm is analyzed experimentally in order to determine the contribution of each of the three used local strategies – modified LR, DDC, and Lookahead – to increase the performance of the NAS algorithm.

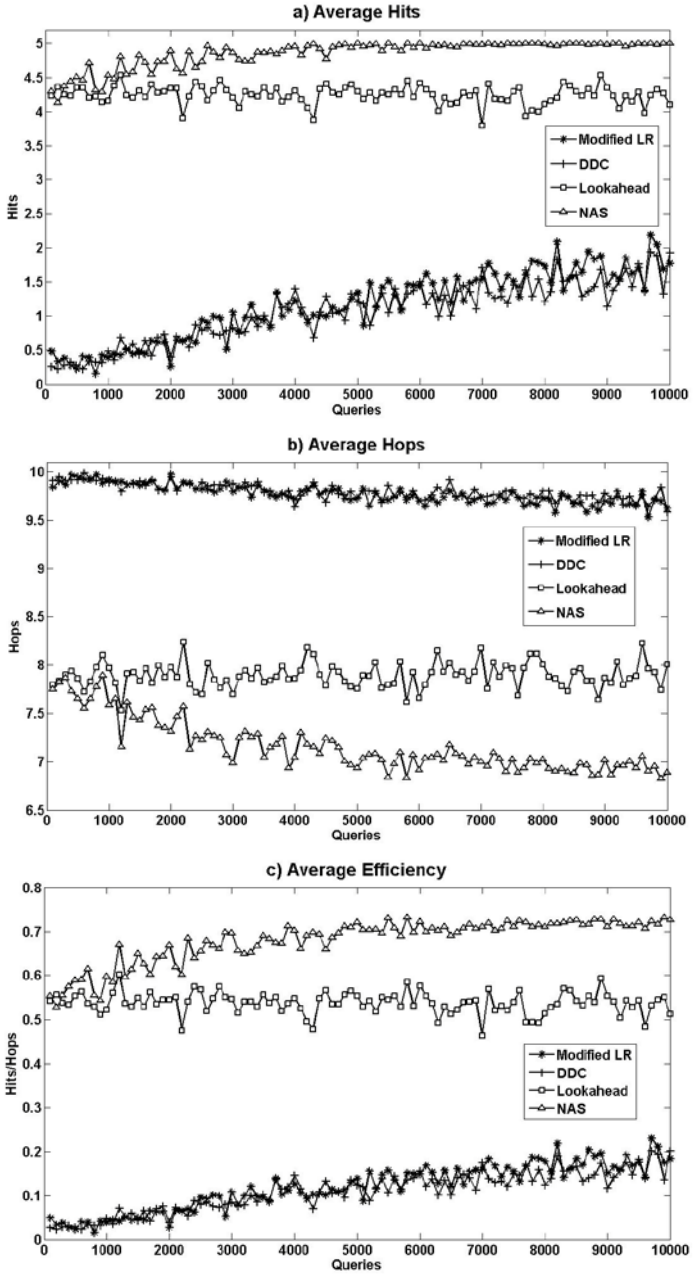


Fig. 7. Comparisons of NAS Local Environment Strategies. a) Average HITS, b) Average HOPS and c) Average Efficiency

For evaluating the contribution of modified LR, DDC and Lookahead strategies were eliminated. The contribution of DDC was evaluated eliminating only Lookahead; modified LR was kept because DDC is included in it. Similarly, Lookahead was evaluated without modified LR and DDC. For the experiment with the NAS algorithm, we used the general specification described in Section 6.1, with the exception of three parameters: *maxResult* =5, *maxTTL*=10, and $q_0 = 0.90$.

Figure 7(a) shows the *average hit-rate* performed during a set of queries with NAS and three different configurations of NAS: modified LR, DDC and Lookahead. For the modified LR and DDC, the algorithms start approximately at 0.5 hits per query; at the end, the average hit-rate increases to 2 hits per query. For Lookahead and NAS the average hit-rate starts at 4.3 and after 1,000 queries the behavior changes; for Lookahead the average hit-rate ends at 4 and for NAS ends at 5. On the other hand, Figure 7(b) shows the *average hops* performed during in a set of queries with NAS and three different configurations. For modified LR and DDC, the behavior is approximately the same; the average hop count starts at 10 and ends at 9.5 hops per query. However, the Lookahead and NAS start at 7.8 and after 1500 queries, the behavior changes; for Lookahead the average hop count ends at 8 and for NAS ends at 6.9 hops per query. Finally, Figure 7(c) shows the *average efficiency*. For the modified LR and DDC, the behavior is approximately the same; at the beginning the efficiency is around 0.05, at the end the efficiency increases to 0.2 hits per hop. However, for the Lookahead and NAS, the behavior evolves such that the average efficiency stars at 0.55 and after 1,500 queries, the behavior changes; for Lookahead, the average efficiency ends at 0.5, for NAS, the average efficiency ends at 0.7. Hence, the best performance was obtained with the combination of these three strategies.

Besides, the results reveal that for the specified configuration, the Lookahead method shown the biggest contribution to the final performance of NAS, giving an efficiency of 0.5 hits per hop. While the DDC and modified LR had a similar impact of 0.2 hits per hop. In experimentations with other configurations [13], the analyzed strategies have shown different contributions. Due to this result, it becomes relevant to study further the relations that exist between the problem characteristic and the algorithm parameter configuration in order to yield a bigger benefit of each one local strategy proposed for NAS.

7 Conclusions and future work

For the solution of the Semantic Query Routing Problem (SQRP), we proposed a novel algorithm called NAS that is based on existing ant colony algorithms but incorporating local environment strategies: modified LR, DDC, and Lookahead. Three functions are used to learn from past performance: importance of hits (HIT), importance of time-to-live (ITL_HOP), and importance of distance (ID_HOP). This combination results in a lower hop count and an upper hit count, outperforming two algorithms proposed in related work, Random-Walk and SemAnt.

Our analysis and simulations confirm that the proposed techniques are more effective at improving search efficiency. Specifically the NAS algorithm in the efficiency shows six times better performance efficiency, than the SemAnt, and seventeen times better performance than the Random-Walk. We observe that upon including learning and characterization with modified LR and DDC respectively, the algorithm evolves to reach an average of 2 hits with 9.5 hops per query. Adding only exploration with Lookahead, the algorithm keeps a constant performance of 4 hits with 8 hops. Combining all the three strategies the NAS algorithm evolves to yield 5 hits per 6.9 hops. As observed, the best results were obtained in the combination of proposed strategies.

We plan to study more profoundly the relation among SQRP characteristics, the configuration of NAS algorithm and the local environment strategies employed in the learning curve of ant-colony algorithms, as well as their effect on the performance of hop and hit count measures.

References

1. **Amaral, L. A. N., & Ottino, J. M. (2004).** Complex Systems and Networks: Challenges and Opportunities for Chemical and Biological Engineers, *Chemical Engineering Scientist*, 59(1), 1653–1666.

2. **Androutsellis-Theotokis Stephanos & Diomidis Spinellis (2004).** A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4), 335–371.
3. **Arora, S., & Barak, B. (2009).** *Complexity Theory: A Modern Approach*. New York: Cambridge University Press.
4. **Barabasi, A., Albert & R., Jeong, H. (1999).** Mean-Field theory for Scale-free Random Networks. *Physical A*, 272(1), 173–189.
5. **Cruz-Reyes, L., Gómez S. C., Aguirre L. M., Schaeffer E., Turrubiates L.T., Ortega I. R., & Fraire H. H. (2008).** NAS Algorithm for Semantic Query Routing System for Complex Network. *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), Advances in Soft Computing*, 50, 284-292.
6. **Di Caro, G. & Dorigo, M. (1998).** AntNet: Distributed Stigmergy Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9(1), 317-365.
7. **Dorigo, M. & Gambardella, L. M. (1997).** Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, 1(1), 53-66.
8. **Erdős, P. & Rényi, A. (1960).** On the Evolution of Random Graphs. *Publications of the Mathematical Institute of the Hungarian Academic of Sciences.* 5(1), 17-61.
9. **Liu, J., XiaoLong, J. & Kwok, C.T. (2005).** *Autonomy Oriented Computing /From Problem Solving to Complex System Modeling*. New York: Kluwer Academic Publisher.
10. **Michlmayr, E. (2007).** *Ant Algorithms for Self-Organization in Social Networks*. PhD Thesis, Vienna University of Technology. Austria, Vienna.
11. **Mihail, M., Saberi A. & Tetali P. (2006).** Random Walks with Lookahead in Power Law Random graphs. *Internet Mathematics*, 3(2), 147-152.
12. **Ortega R., Meza E., Gómez C., Cruz L., & Turrubiates T. (2007).** Impact of Dynamic Growing on the Internet Degree Distribution. *Polish Journal of Environmental Studies*, 16(1), 117-120.
13. **Sakaryan G. (2004).** *A Content-Oriented Approach to Topology Evolution and Search in Peer-to-Peer Systems*, PhD Thesis, University of Rostock. Rostock, Germany.
14. **Yang, K. & Wu, C., Ho, J. (2006).** AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks. *IEICE Transactions on Communications*, 89(9), 2300-2308.

Claudia Gómez S*. was born in Mexico in 1971. She is a doctoral student at National Polytechnic Institute, Mexico. She received her MS degree in Computer Science from the Leon Institute of Technology, Mexico, in 2000. Her research interests are optimization Techniques, complex network and autonomous agents.*

Laura Cruz-Reyes *was born in Mexico in 1959. She received the PhD (Computer Science) degree from National Center of Research and Technological Development, Mexico, in 2004. She is a professor at Madero City Institute of Technology, Mexico. Her research interests include optimization techniques, complex networks, autonomous agents and algorithm performance explanation.*

Eustorgio Meza *received the PhD (Oceanic Engineering) deegree from Texas A&M University, College Station, U.S.A. He received the MS degree in Computer Science (AI) from Institute of Technology and Advanced Studies of Monterrey, Mexico. He is a Professor at Research Center in Applied Science and Advanced Technology from National Polytechnic Institute. His research interests are oceanology, complex Network.*

Elisa Schaeffer *was born in Finland in 1976. She received the PhD (Science in Technology) degree from Helsinki University of Technology, Finland, in 2006. She is a teaching researcher at the Graduate Program in Systems Engineering (PISIS), at the Faculty of Mechanical and Electrical Engineering (FIME), Universidad Autonoma de Nuevo Leon, Mexico. Her research interests include nonuniform networks, graph clustering and optimization of network operation*

Guadalupe Castilla*. is a doctoral student at Madero Institute of Technology, Mexico. She received her MS degree in Computer Science from the Leon Institute of Technology, Mexico, in 2002. Her research interests are optimization Techniques.*

Accepted Manuscript

Ant colony system with characterization-based Heuristics for a bottled-products distribution logistics system

Claudia G. Gómez S., Laura Cruz-Reyes, Juan J. González B., Héctor Fraire H., Rodolfo A. Pazos R., Juan J. Martínez P.

PII: S0377-0427(13)00588-8
DOI: http://dx.doi.org/10.1016/j.cam.2013.10.035
Reference: CAM 9429

To appear in: *Journal of Computational and Applied Mathematics*

Received date: 15 February 2013
Revised date: 18 October 2013

Please cite this article as: C.G. Gómez S., L. Cruz-Reyes, J.J. González B., H. Fraire H., R.A. Pazos R., J.J. Martínez P., Ant colony system with characterization-based Heuristics for a bottled-products distribution logistics system, *Journal of Computational and Applied Mathematics* (2013), http://dx.doi.org/10.1016/j.cam.2013.10.035

Ant Colony System with Characterization-Based Heuristics for a Bottled-Products Distribution Logistics System

Claudia G. Gómez S., Laura Cruz-Reyes, Juan J. González B., Héctor Fraire H., Rodolfo A. Pazos R., Juan J. Martínez P.

Instituto Tecnológico de Ciudad Madero, Av. 1o. de Mayo s/No Col. Los Mangos, Cd. Madero, Tamaulipas, Mexico.

Abstract

The aim of this paper is to show the solution of the Vehicle Routing Problem with Time Windows (VRPTW) as a key factor to solve a logistics system for the distribution of bottled products. We made a hybridization between an Ant Colony System algorithm (ACS) and a set of heuristics focused on instance characterization and performance learning. We mainly propose a method to make a constrained list of candidate customers called Extended Constrained List (ECL) heuristics. Such list is built based on the characterization of the time-window and the geographical distribution of customers. This list gives priority to the nearest customers with a smaller time window. The ECL heuristics is complemented by the Learning Levels (LL) heuristics, that allows the ants to use the pheromone matrix in two phases: local and global. In order to validate the benefits of each heuristics, a series of computational experiments were conducted using the standard Solomon's benchmark. The experimental results show that, when the ECL heuristics is incorporated in the basic ACS algorithm, the number of required vehicles is reduced by 28.16%. When the LL heuristics is incorporated, this reduction increases to 36.83%. The experimentation reveals that, by a suitable characterization, preexisting conditions in the instances are identified in order to take advantage from both of the ECL and LL.

Keywords: Heuristics hybridization, Vehicle routing problem, Ant colony system algorithm.

1. Introduction

Logistics systems applied to transport systems are a current problem in productive and service sectors. Distributors should effectively and efficiently stock customers with products, which constitutes a major challenge for a logistics system, since resources are limited and transportation costs constitute a high percentage of the value added to goods: from 5% to 20% [1]. Solving real-life world transportation problems requires the development of robust methods that support the complexity of these kind of problems. In order to contribute to this area, we have a methodological solution developed by integrating the main tasks (routing, scheduling and loading) involved in a generic transportation problem, called RoSLoP [2, 3].

RoSLoP focuses mainly on routing vehicles applied to the distribution of bottled products in a company located in northeast Mexico [2, 6]. The objective of the routing task is to define routes for vehicles so as to minimizing the total cost subject to twelve real world constraints [2]. Considering the difficulty of handling several realistic constraints at the same time, this problem is often called Rich VRP [4, 3].

This work extends the solution presented in [2] for RoSLoP, particularly for the VRPTW routing task. The proposal of this research is based on a heuristics hybridization of an ACS algorithm. The purpose of the hybridization is to improve the optimization process by characterizing the problem according to two aspects of customers: geographic distribution (topology) and service time distribution (time-window types). The heuristics are: Extended Constrained List (ECL) and Learning Levels (LL) [5, 3]. The first heuristics constitutes the main contribution of this work, it provides a higher probability of selection of those customers which are closer and have smaller time-window types, whereas the second heuristics slightly increases the probability for movements which have previously contributed to improve the solution.

Like our work, researchers have presented since the 1960's their proposals related to VRPTW in order to solve this problem. The main work includes Pisinger and Ropke [7], who present an extension of Large Neighborhood Search (LNS) proposed by [8]. Their algorithm, called ALNS, is an adaptation of the LNS algorithm by building partial neighborhoods which compete to iteratively modify the current solution.

Another researcher, Mester [9], proposed a heuristic algorithm based on (1+1) multi-parametric evolutionary strategies. Three operators are used for

the removal of customers (purely random removals, removal of one customer from each route, and random ejections from rings generated from two circles centered on the depot with random radiuses), while a cheapest insertion heuristics is applied for re-insertion. Each offspring is further improved using Or-Opt, Exchange, 2-Opt local moves within a parameterized dynamic environment "Adaptive Variable Neighborhood" (AVN). In an effort to prune the neighborhood space, a strategy for selecting neighboring routes, called "Dichotomous Route Combinations" (DRC), is also used to take advantage of the geographical division and topology of the vehicle routes.

A different strategy, but also based on Local Search, is proposed by Prescott-Gagnon et al. [10]; they present a hybridization of a branch-and-price heuristic technique, a LNS method composed by operators related to customer selection known as: Proximity, SMART, and Longest Detour. Another local-search based heuristics was developed by Hoshino et al. [11]; such heuristics is controlled by chaotic dynamics exploiting principles of neural networks. The chaotic search applies by exchanging and relocating local moves. The neurons are updated asynchronously while a single iteration is performed, and finally the route elimination heuristics of [12] is applied periodically for further improvement.

After reviewing state-of-the-art papers, it is concluded that the ECL and LL heuristics are original. Table 4 in Subsection 5 presents comparative results obtained by our ACS algorithm including the proposed heuristics versus other works here presented.

2. Vehicle Routing Problem with Time Windows (VRPTW)

Vehicle routing has been of great interest for the scientific community over the past fifty years. However, open questions remain due to its complexity [13]. VRP, defined by Dantzig in [14], is a classic combinatorial optimization problem. It consists of trying to service a set of customers using a fleet of vehicles, respecting constraints on the vehicles, customers, drivers, and so on. Among the most important variants of VRP is the VRPTW defined by Kallehauge in [15]. This problem consists of finding the optimal routing of a fleet of vehicles from a depot to a number of customers that must be visited within a specified time interval, called time window. A time and capacity constrained digraph $G = (V, A, c, t, a, b, d, q)$ is defined with the following elements:

- a node set $V = V_* \cup \{0, n+1\}$, where $V_* = \{1, \ldots, n\}$ is the set of customer nodes, and the nodes 0 and $n+1$ are the starting depot and the returning depot respectively,
- an arc set $A = A_* \cup \delta^+(0) \cup \delta^-(n+1)$, where $A_* = A(V_*)$, is the set of arcs (i, j) such that $(i, j) \in V_*$, additionally $\delta^+(0) = \{(0, i) | i \in V_*\}$ is the set of arcs leaving the start depot node, and $\delta^-(n+1) = \{(i, n+1) | i \in V_*\}$ is the set of arcs entering the destination depot node,
- durations (traveling time) on arcs $t \in \mathbb{N}^{|A|}$, such that $t_{ij} \leq t_{ik} + t_{kj}$, for $i, j, k \in V$,
- start and end service time on nodes $a, b \in \{\mathbb{Z}_+ \cup \{+\infty\}\}^{|V|}$, where $a_0 = a_{n+1} = 0$, $b_0 = b_{n+1} = +\infty$, $a_i \geq t_{0i}$, $b_i \geq a_i$ for $i \in V_*$ and $b_j \geq a_i + t_{ij}$ for $(i, j) \in A_*$,
- demands on nodes $d \in \mathbb{Z}_+^{|V|}$, where $d_0 = d_{n+1} = 0$,
- and a vehicle capacity $q \in \mathbb{Z}_+$ where $q \geq d_i$ for $i \in V_*$ and $q \geq d_i + d_j$ for $(i, j) \in A_*$.

For any path $P = (v_1, \ldots, v_k)$ in G, the arrival times of the set of nodes $V(P)$ of the path is the vector $s \in \mathbb{Z}_+^{|V(P)|}$, whose elements are defined as follows: $s_{v_1} = a_{v_1}$, and $s_{v_i} = \max_{i=2,\ldots,k}\{s_{v_{i-1}} + t_{v_{i-1}v_i}, a_{v_i}\}$. The demand of the path is $d(V(P))$.

A path $P = (v_1, \ldots, v_k)$ in G is feasible if $s_{v_i} \leq b_{v_i}$ for $i \in V(P)$ and $d(V(P)) \leq q$. A feasible route R from 0 to $n+1$ in G is defined as $R = (0, v_2, \ldots, v_{k-1}, n+1)$. We denote by $\Re$ the set of all feasible routes from 0 to $n+1$ in G. For each feasible route a vehicle is required to service the customers included in the route.

Given a time and capacity constrained digraph G, the vehicle routing problem with time windows consists of minimizing the number of required vehicles and the total time required to service all the customers demand.

As VRPTW is a multiobjective problem, in this work a hierarchical technique is applied; i.e., the objectives of the problem are achieved consecutively: minimizing the number of vehicles required, and minimizing the traveling and waiting times required to service all customers.

3. Hybridization of the Ant Colony System Algorithm

Ant algorithms were inspired by the behavior of ants in search for food, because in performing the search, each ant drops a chemical called pheromone,

which provides an indirect communication among the ants. Every algorithm based on ant colony includes two main features: a heuristic measure η_{ij}, which provides information related to the problem used to compute preference of travel; a larger η_{ij} value implies a larger probability of selecting node j. The second feature is trails of artificial pheromone τ_{ij}, which is a table that keeps learning over time and is used in the selection rule of the next customer to visit [16]. These heuristic measures are included within the $ACS-VEI()$ and $ACS-TIME()$ functions and are explained in Subsection 4.3.

To build feasible solutions for RoSLoP, three modules were created: routing, scheduling and loading. The routing module is an Ant Colony System algorithm [2]. The first action of Algorithm 1 consists of applying the Extended Constrained List (ECL) heuristics, which aims at characterizing the topology and time windows (see construction details in Subsections 4.1 and 4.2). Next, an initial solution is calculated, using the nearest-neighbors strategy [18].

Algorithm 1 ACS algorithm for RoSLoP routing

Multiple Ant Colony System $(\rho, \beta, q0)$
$ECL()$ / $*$ *Construction of the constrained list*
$\psi^{gs} \leftarrow InitialSolution()$
repeat
  $vehicles \leftarrow active\_vehicle(\psi^{gs})$/$*$ *Number of vehicles used by the best solution*
  $vehiclesbeta \leftarrow vehicles - 1$
  //*begin* $ACS-VEI$/$*$ *vehicle minimization process*
    $LL\ (\psi^{ls})$
    $LocalSearch\ (\psi^{ls})$
    textbfif $(\psi^{ls} isfeasible)$
      $\psi^{gs} = \psi^{ls}$
    end if
until *stop criterion is satisfied*
repeat
  $\psi^{ls} \leftarrow \psi^{gs}$
  //*begin* $ACS-TIME$/$*$ *vehicle minimization process*
    $LL\ (\psi^{ls})$
    $LocalSearch\ (\psi^{ls})$
    if $(\psi^{ls} < \psi^{gs})$
      $\psi^{gs} = \psi^{ls}$
    end if
until *stop criterion is satisfied*

In the following lines Algorithm 1 creates two ant colonies: the first for

minimizing the number of vehicles and the second for minimizing travel time. Each colony is constituted by a set of ants and each step builds a local solution (ψ^{ls}), which will be evaluated to identify the best local solution (ψ^{ls}); therefore, it is considered as the global solution (ψ^{gs}). The value of the pheromone is modified to stimulate finding both popular and better routes. At each step, the result from the ECL heuristics is used to select the next customer to be visited, which favors choosing the nearest customers and those that have limited opening hours (see details in Subsection 4.3). The Learning Levels (LL) heuristics takes control of the pheromone on two levels: the lowest level is for each ant and the highest for each colony [5, 3] (see details in Subsection 4.4).

4. Heuristics for selecting customers

This section describes two heuristic methods proposed to build a customer list grouped by proximity and availability of service, another heuristics keeps a table with information from the best solutions so far obtained. Their aim is to limit the selection of customers to those with characteristics suitable for the route under construction. These heuristics collaborate hybridizing Algorithm 1 to improve its performance.

4.1. Construction of the constrained list using topological characterization

The first part of the heuristic method, proposed for customer selection, is based on a clustering technique. A list of clusters of candidates is generated considering feasibility conditions and the location of the customers in the graph. Particularly, a clustering technique based on topological characterization is applied to limit the global population into subsets that fulfill feasibility and closeness. The use of this clustering technique by the ants in the constructive process is very advantageous; while the hybridization reduces the search space in the creation of feasible solutions due to the incorporation of instance information about the distribution of the customers and the depot. The constrained list organized by clusters is created as follows:

1. A minimum spanning tree T is generated including all the customers and the depot of the instance with Algorithm 2. This tree T is a subgraph of A that connects all customers at minimal cost. Figure 1 shows the result of calculating T for some Solomon's instances [17].

Algorithm 2 Construction of the Minimun Spanning Tree T

$T = \{\emptyset\}$, $L = A$, $Sort(L)$
for each $(i, j) \in L$
  if i and j belong to T
    $discard(i, j)$
  else
    $T = T \cup (i, j)$

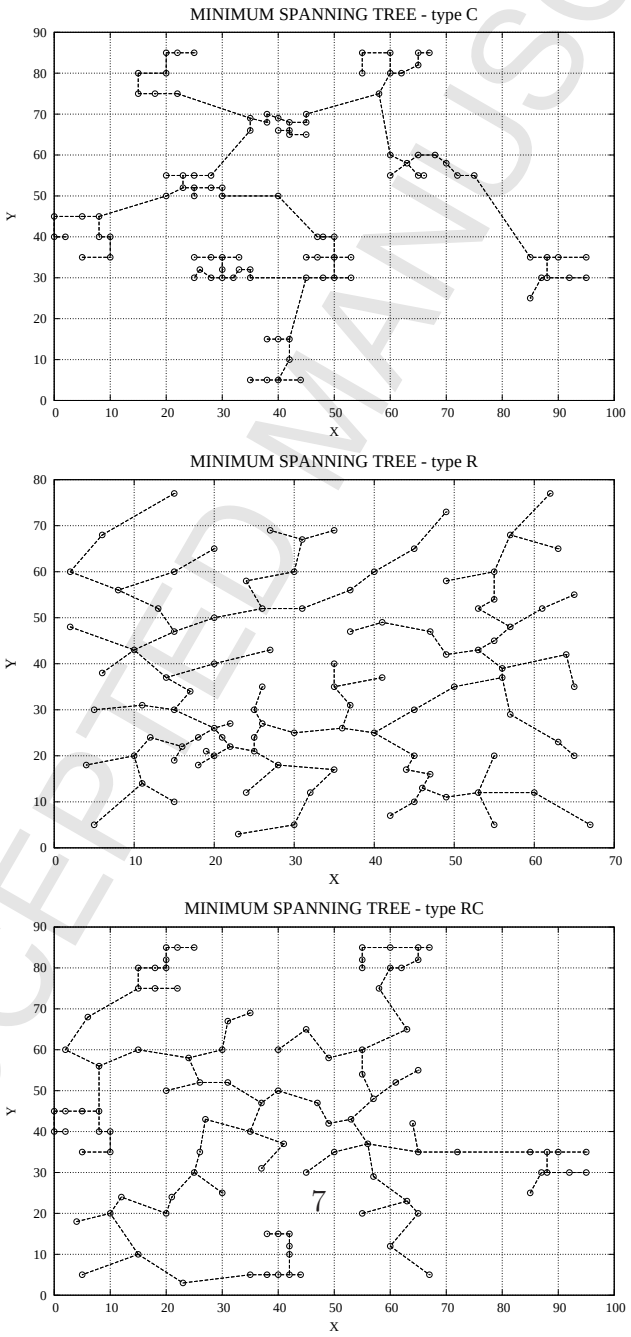


Figure 1: Example of a T for each type of instance: C, R and RC

2. Parameters μ and σ are calculated from expressions (1) and (2), where each tc_{ij} is the distance from customer i to customer j. These data will be used for defining the percentage of variability.

$$\mu = \sum_{i,j \ \epsilon T, \ i \neq j} tc_{ij} \tag{1}$$

$$\sigma = \sqrt{\frac{1}{|V|} \sum_{i,j \epsilon \ T} (tc_{ij} - \mu)^2} \tag{2}$$

3. The percentage of variability θ of the associated costs of each arc belonging to T is computed by expression (3), notice that θ normalizes σ.

$$\theta = \frac{\sigma}{2(\max_{(i,j)\in T}\{tc_{ij}\} - \min_{(i,j)\in T}\{tc_{ij}\})} \tag{3}$$

   In case $\theta < 0.1$ (very small variability), the location of the customers in the instance approaches a uniform distribution; therefore, the entire population forms a single cluster. Otherwise, a value of $\theta \geq 0.1$ reveals the possible existence of regions with different density.

4. The formation of the set of clusters H is carried out by performing a hierarchical clustering when the rule "if $\theta \geq 0.1$" is satisfied; otherwise, customers form a single cluster $|H| = 1$. Figure 2 shows clusters formed under variability percentage as defined by expression (3).

5. In the proposed hierarchical clustering, each customer starts in its own cluster, and pairs of clusters are merged iteratively in order to find an appropriate set of clusters H, based on an acceptance threshold ω. The initial value for ω is calculated by expression (4).

$$\omega = 2 \ \max_{(i,j)\in T}\{tc_{ij}\} \tag{4}$$

   For each pair of clusters (h_i, h_j), where $h_i, \ h_j \in H$, the well know Mahalanobis Distance (dM) is used with the rule "if $dM(h_i, h_j) < \omega$ and $h_i \neq h_j$ then $h_i \cup h_j$" for determining if both clusters are merged in one cluster.

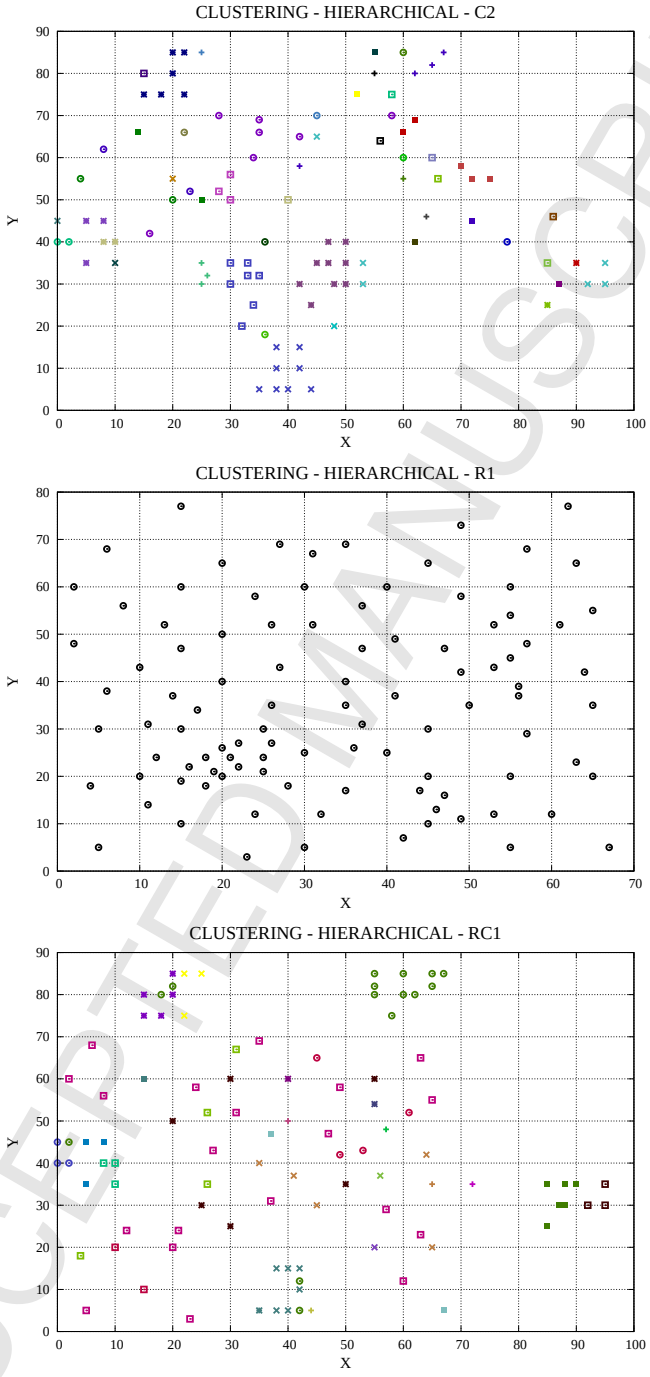


Figure 2: Examples of different cluster formations

Cluster formation continues as long as merge operations exist with the same value of ω, otherwise ω is modified by $\omega = \omega(1 + \theta)$ and ends if there is no cluster reduction with the new ω.

4.2. Extension of the constrained list using time-window characterization

After applying a clustering technique to characterize the topology, we formulate another characterization for the time-window length, which extracts additional information that helps to characterize the problem instances. In a preliminary experiment, it was found that it is possible to characterize the behavior of the customer time windows in a given Solomon's instance. We identified three types of distributions based on the shape of the time-window graph: Linear, Curved and Phased.

The objective of this new strategy is to identify those costumers that have a short length of time window, only when the distribution of the time window of a given instance has a phased form. For this type of instances a classification of its customers is carried out by detecting those customers with a short length of time window in order to give them a larger visit preference when the ants perform a search for a new solution.

The extension of the constrained list is created in five steps:

1. Using expression (5), calculate the time-window length for each customer of the instance, where tw_i is the time-window length of customer i, where a_i and b_i are the start and end of the time window.

$$tw_i = b_i - a_i \tag{5}$$

2. Define four time intervals: $Int_k = ((k-1)[IntLength], k[IntLength])$, $k = 1, ..., 4$; where $IntLength$ denotes an interval length, given by expression (6).

$$IntLength = \frac{\max(tw_i)}{4} \tag{6}$$

3. For each interval Int_k, determine the number of customers O_k whose time window lies within the interval, given by expressions (7) and (8), in order to classify an instance based on its time-windows distribution.

$$O_k = \sum_{i=1}^{|V|} g(i,k) \tag{7}$$

where

$$g(i,k) = \begin{cases} 1 & \text{if } tw_i \in Int_k \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

4. Determine the number of non-empty intervals I using expression (9), then fill the priority table P of size $n \times n$, when $I = 2$ and some visiting preferences between customers are satisfied.

$$I = \sum_{k=1}^{4} g(k) \text{ ; where}$$

$$g(k) = \begin{cases} 1 & \text{if } O_k > 0 \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

5. Determine the priority of $(i,\ j)$ (p_{ij} in expression (10) for each pair $(i,j) \in A$, where i is the current customer, j and k are yet unassigned to the path).

$$p_{ij} = \begin{cases} 3 & \text{if } I = 2 \wedge (tw_i, tw_j) \in Int_{minSet} \wedge wait^1_{ij} < \lambda_1, \\ 2 & \text{if } I = 2 \wedge wait^1_{ij} \geq \lambda_1 \wedge wait^2_{ij} < \lambda_2 \wedge de_{ik} < \lambda_2, \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $minSet = \min_{i \in [0,4]}\{i | O_i \geq 1\}$, $wait^2_{ij} = a_j - (a_i + s_i + de_{ik} + s_k)$ and $wait^1_{ij} = (a_j - a_i - s_i) - de_{ij}$. Parameters a_i and a_j are the time-window start for customers i and j, s_i is the service time of customer i, de_{ij} is the Euclidian distance and λ_1 and λ_2 are waiting thresholds. Priority levels two and three are only for phased instances $I = 2$). Level three is for a pair of customers (i,j), both customers with short time-window length (they belong to the first enabled interval) and with an acceptable waiting time between customers i and j bounded by λ_1. Pairs of customers with level two do not have acceptable waiting time between them, but they have an intermediate customer to meet this condition.

As a result of applying the ECL in the Solomon's set [17], the distributions permit identifying three types of time windows:

Type 1: Linear Distribution

In Figure 3-a we show an example of linear distribution, where time-window lengths are constant for all the customers in the instance. We found that there were 12 instances out of the entire set of 56, which have a linear distribution; this constitutes approximately 21% of the total.

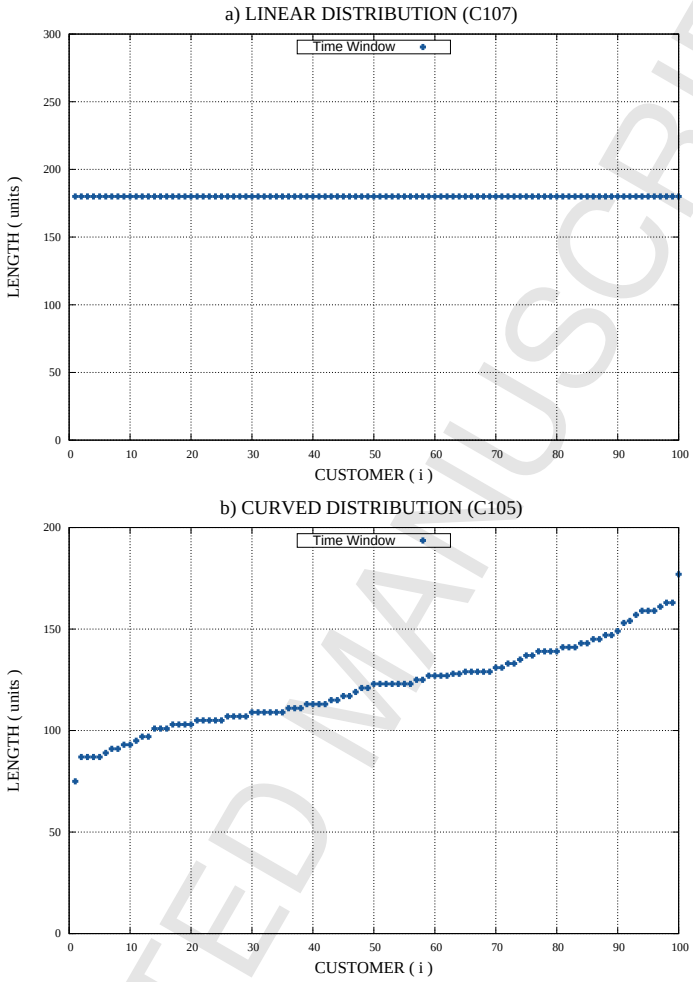


Figure 3: Two types of distribution: a) Linear Distribution, b) Curved Distribution

Type 2: Curved Distribution

We defined curved behavior when costumers have uniform differences in length from each other in a certain instance as can be seen in Figure 3-b. We found that 33% of the instances show a curved behavior in their time-window distribution.

Type 3: Phased Distribution

In Figures 4 and 5, we present an example for phased distribution for the time-window length. We found that phased distribution occurs in 26

instances out of the total of 56; this constitutes 46% for the total instances. In the graph of phased distribution two groups of costumers are clearly discerned, those who have a small time-window length and those that have large time-window length. A particular feature in this type of distribution is the percentage of customers who have long lengths, which can constitute 25%, 50% or 75% of all the costumers. The most interesting structure of the instances is the phased distribution that constitutes about 50% of the instances in the Solomon's benchmark. Since this distribution makes the distinction between two groups of costumers (those with short length and long length), it is convenient to use this information to improve the ACS algorithm.

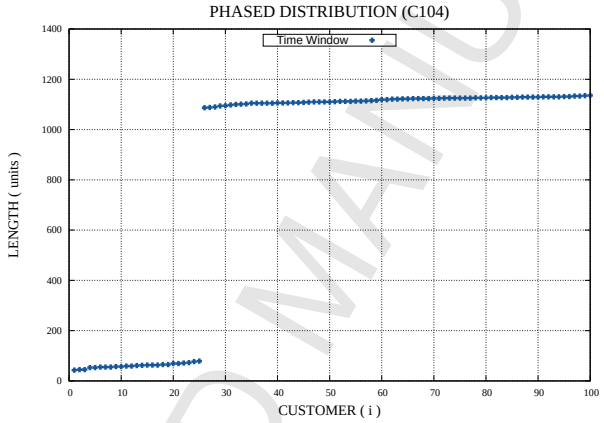


Figure 4: Another type of distribution: Phased Distribution 25%

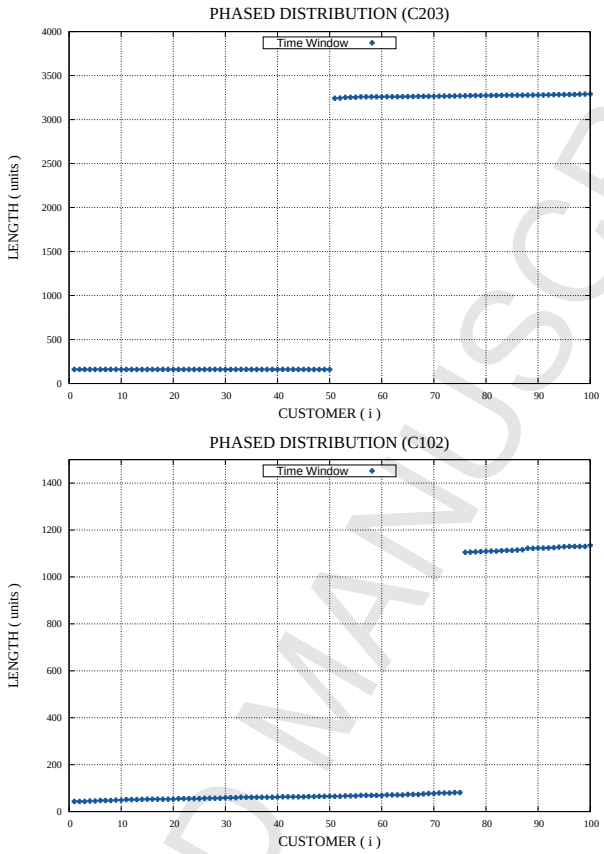


Figure 5: Another type of distribution: Phased Distribution 50% and 75%

4.3. Selection of candidates for incorporation into routes

Once the membership into a group based on topological distribution (Subsection 4.1) and priority based on time-window length distribution (Subsection 4.2) of each customer has been defined, the candidate selection is performed according to the selection rule used primarily to update the pheromone and heuristic information:

- Information of artificial pheromone trails τ, which determines the movement preference from one customer to another; this preference is modified during the execution of the algorithm depending on the solutions as they are being obtained.

- Heuristic information η, which measures the preference for the path, including information from the constrained list using topological characterization and the characterization of the constrained list extension by using time-window length.

The selection rule shown in expression (11), is controlled by a balance parameter $q_0 \in [0, 1]$ and a random value q. When $q \leq q_0$, it exploits the knowledge available, choosing the best option according to the heuristic information and the pheromone trails. However, if $q > q_0$, it applies an exploration for new movements from customer i to customers j.

$$s = \begin{cases} \arg\max_{j \in N(i)} \{\tau_{ij}[\eta_{ij}]^{\beta}\} & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{11}$$

$$S = f(prob(i,j)) \text{ where } prob(i,j) = \frac{\tau_{ij}[\eta_{ij}]^{\beta}}{\sum_{k \in N(i)} \tau_{ik}[\eta_{ik}]^{\beta}} \tag{12}$$

In expression (11) β is the relative importance of the heuristic information, $N(i)$ is the set of available neighbor customers and S is a random function that selects customer j according to a probability distribution given by expression (12).

The value for η is calculated as follows: if the source customer v_i and the destination customer v_j belong to different clusters ($h_l \neq h_m$) a correction factor to the heuristic information is applied (see expression (13)), which depends on the size of the set of costumers V and the size of the set of clusters H; otherwise, the priority setting established by the time-window length is used.

$$\eta_{ij} = \begin{cases} \eta_{ij} \frac{|H|}{|V|} & \text{if } h_l \neq h_m \wedge v_i \in h_l \wedge v_j \in h_m \\ \eta_{ij}\ p_{ij} & \text{otherwise} \end{cases} \tag{13}$$

where $h_l, h_m \in H$ (cluster set), $v_i, v_j \in V$ (customers) and p_{ij} denotes time-window information.

4.4. Learning Levels

Learning levels (LL) defines two levels of knowledge: global and local. The first level consists of the values of the original pheromone table τ, which

only contains the information of the best solution (ψ^{gs}) obtained by the ants and it is modified only in the global updating process.

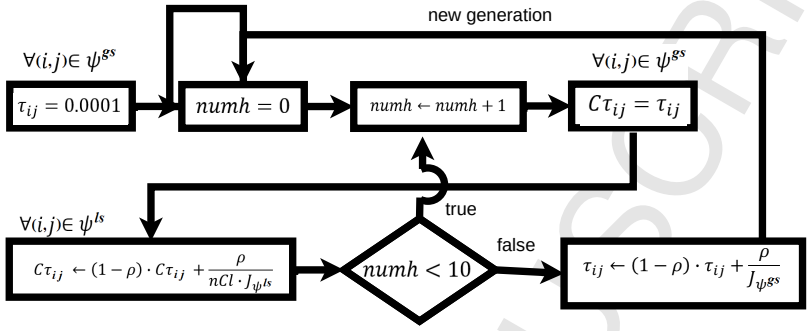


Figure 6: Allocation process for learning levels

The second level consists of the copy of the pheromone table $C\tau$, which contains the local values of the pheromone and it is used by the ants as a guide when searching for better solutions. During a local update process, each ant updates its copy of the pheromone table $C\tau_{ij}$. The global pheromone table takes the best local pheromone table. The allocation process for learning levels is described in Figure 6 [4].

5. Computational Results

The computational experiments were done on a computer with an Intel Xeon 5120 with 2 cores at 1.86 GHz with 3.0 GB of RAM. The operating system was Windows XP Service Pack 3 (32bits), with the IDE of Microsoft Visual C# Express 2010. For the hypothesis test the SPSS software was used. The ACS algorithm parameter values used were $n = 10$ for artificial ants and $\alpha = 1$, $q_0 = 0.65$, $\beta = 6$ and $\rho = 0.1$. The stopping condition was 1800 seconds for each Solomon's instance [17]. In the VRPTW context, the most common way to compare the performance of heuristic algorithms is solving the instances of the Solomon's benchmark [17].

These problem instances include a central depot and a variety of: customers, capacity constraints, demand constraints, and time-window distributions. The objective for real applications of VRPTW is to minimize the number of required vehicles. Therefore, we carried out a series of experiments

to solve the Solomon's instances, which are characterized as shown in Table 1.

The main goal of the experiments was to determine the impact on the performance of the ACS algorithm when each heuristics is individually incorporated and when both heuristics are jointly applied. Tables 2 and 3 show the results obtained in the experiments. The first column contains the identifier of the solved instance. The next four columns show the algorithm performance measured in number of vehicles, for the algorithms: basic ACS, ACS with constrained list (ACS+ECL), ACS with learning levels (ACS+LL), and ACS with both heuristics (ACS+ECL+LL). Additionally, columns 3-5 show the reduction levels reached, in the vehicles number for each instance type and for all the instances. As we can see, when the two heuristics are individually applied to the ACS algorithm, ACS+ECL consistently outperforms ACS+LL, except for the R1 instance type. This behavior can be explained because the ECL heuristics was designed to exploit the customers clustering and the time-window length. As we can observe in Table 1, the R1 instance type does not meet these two conditions. It is remarkable that for the types where ACS+ECL outperforms ACS+LL, the percentage reduction reached with ECL is significantly larger than the reduction attained with LL.

Instances with Linear, Curved and Phased time-window distributions were included in the experiment, see Subsection 4.2.

Moreover, the best performance is achieved when both heuristics are incorporated (ACS+ECL+LL). As we can see this hybrid algorithm outperforms ACS, ACS+ECL and ACS+LL; the percentage reduction is 36.83% over all the instance types.

To determine if the observed differences were statistically significant, a Wilcoxon hypothesis test was carried out. In Table 3, the results of the tests for each pair of the assessed algorithms are shown. The first two columns indicate the algorithms considered; in the test the differences considered were (Algorithm 2 - Algorithm 1). The next four columns contain the number of measures used N, the number of ties T, the value of the normal estimation Z, and the corresponding p-value. As we can see, for all the algorithm pairs the p-value is smaller than 0.05. Therefore, the observed differences between the performances of each algorithm pair are statistically significant with a reliability level of 95%.

Table 4 shows a comparison of the results of our approach versus those of the state of the art. The first column indicates instance types, the next two columns contain our results showing vehicles average and traveled distance

Table 1: Characterization of Solomon's instances

Type	Description
C1	Customers are distributed by clusters with tight time windows and short scheduling horizons.
C2	Customers are distributed by clusters with wide time windows and long scheduling horizons.
R1	Customers are randomly generated with a uniform distribution with tight time windows and short scheduling horizons.
R2	Customers are randomly generated with a uniform distribution with wide time windows and long scheduling horizons.
RC1	Customers are semi-clustered (i. e., combination of both clusters and randomly distributed) with tight time windows and long scheduling horizons.
RC2	Customers are semi-clustered (i. e., combination of both clusters and randomly distributed) with wide time windows and long scheduling horizons.

average. From the fourth to the sixth column, state of the art results are shown; these columns contain respectively vehicles average, distance average and the best approach found for the corresponding instance type. As observed, the best results are not obtained by the same approach. This implies that state of the art algorithms do not consistently obtain the best results for all instance types.

The last row of Table 4 shows that our algorithm uses only 1.17 vehicles and 228.63 distance units more than a hypothetical algorithm (i.e., one that yields the best solution obtained by each individual state-of-the-art algorithm). Therefore, it is concluded that our proposal is competitive and functions adequately for most of the instances.

Table 2: Performance achieved when the heuristics are individually and jointly applied to the basic ACS on all the instances

Instances	Vehicles			
	ACS	ACS+ECL	ACS+LL	ACS+ECL +LL
C1	157	91	149	90
Reduction %		42.04%	5.10%	42.68%
C2	66	28	65	24
Reduction %		57.58%	1.52%	63.64%
R1	182	172	165	150
Reduction %		5.49%	9.34%	17.58%
R2	44	37	43	31
Reduction %		15.91%	2.27%	29.55%
RC1	149	112	135	95
Reduction %		24.83%	9.40%	36.24%
RC2	59	32	52	25
Reduction %		46%	12%	57.63%
Total	657	472	609	415
Reduction %		28.16%	7.31%	36.83%

Table 3: Results of the Wilcoxon hypothesis test for each pair of algorithms assessed

Algorithm 1	Algorithm 2	N	T	Z	p-value
ACS+ECL	ACS	56	7	-6.126	0.0001
ACS+LL	ACS	56	25	-4.994	0.0001
ACS+ECL+LL	ACS	56	2	-6.414	0.0001
ACS+LL	ACS+ECL	56	13	-5.127	0.0001
ACS+ECL+LL	ACS+ECL	56	32	-4.668	0.0001
ACS+ECL+LL	ACS+LL	56	7	-5.949	0.0001

6. Conclusions and Outlook

Some optimization algorithms use reinforcement learning (RL) techniques to learn how to make good decisions about the way to perform the search.

Table 4: Comparative results of state-of-the-art approaches versus our proposed approach

Instance type	Our approach		State of the art		
	Vehicles	Distance	Vehicles	Distance	Work
C1	10	829.59	10	828.38	[17]
C2	3	593.88	3	589.86	[17]
R1	12.5	1234.88	11.92	1210.3	[10]
R2	2.82	1057.42	2.73	954.27	[9]
RC1	11.88	1441.89	11.5	1384.2	[9]
RC2	3.38	1146.5	325	1108.52	[17]
TOTAL	43.56	6304.16	42.4	6075.53	–

The RL search methods fall into two main categories [22, 21, 20]: model-based and model-free.

Model-based search (MBS) builds a probabilistic model of the search space and then uses this model to build new solutions to the problem under consideration. The reinforcement feedback from the previously seen solutions is used to improve the model, which is subsequently used to generate new feasible solutions, in such a way that the search will concentrate on the regions containing high quality solutions. The reuse of information stored in its internal model can facilitate decisions and learning in searching of actions that give the largest rewards. However, as the numbers of available actions, possible outcomes, and steps to reach the goal increase, the search requires a large amount of execution time and working memory.

Model-free search (MFS) is an approach that learns directly from experience and searches the solution space without relying on a predictive model of performance to guide the search. An action can be selected greedily or stochastically by comparing the action values of the candidate actions. Model-free search is computationally fast for making estimations. However, it is very slow in terms of its convergence to the optimal solution.

Ant colony optimization (ACO) belongs to the class of model-based search algorithms [21]. In general, the ACO approach attempts to solve an optimization problem by repeating the following two steps: candidate solutions are constructed using a pheromone model, which provides positive feedback;

and the candidate solutions are used to modify the pheromone values. ACO algorithms can easily converge to suboptimal solutions if the positive feedback is not carefully controlled [19]. The Ant Colony System (ACS) is one of the most successful ACO variants in practice.

In this work the solution of the Vehicle Routing Problem with Time Windows (VRPTW) using ACS is approached. A state-of-the-art implementation of ACS uses a constrained list to incorporate geographic information of the customers. We propose an extended constrained list (ECL) to include information of time-window lengths, for identifying those customers with a small length of time window in order to give them a larger visit priority when the ants search for a new solution.

ECL is a model-free method proposed to control the pheromone feedback by increasing the selective pressure toward a solution with desired characteristics. This selective pressure reduces the execution time required by ACS and increases its accuracy. In this sense, the model-free ECL provides complementary estimates for guiding the preferences of the model-based ACS. To the best of our knowledge, the integration of model-based and model-free for VRPTW has not been reported previously. We believe that a wide variety of routing approaches may benefit from the integration of these types of models.

A series of experiments were conducted to determine the impact on the performance of the ACS algorithm when the ECL heuristics is incorporated alone, and when ECL is complemented by the Learning Levels (LL) heuristics. The computational results show that when the two heuristics are individually applied on the ACS algorithm, ECL consistently outperforms LL except for the R1 instance type. Also we can see that the hybrid algorithm ACS+ECL+LL outperforms algorithms ACS, ACS+ECL and ACS+LL over all the instance types and all the instances. For all the instances the percentage reduction attained is 36.83%. The experimental results also reveal that our proposal is competitive with state-of-the-art algorithms.

The hybridization ACS+ECL+LL has only been analyzed with the ant colony system proposed by the authors for VRPTW. Future work is needed to determine the generality of our results. First, it would be interesting to incorporate ECL+LL in other state-of-the-art ACO algorithms. Second, we believe it would be of great importance the study of the performance of each heuristics applied to other algorithms and problems. Since ECL is associated with forming groups in a graph, ECL could complement the selection of the next action in search algorithms for graph problems. Besides, we believe

that if we change our grouping method by another general-purpose method, ECL could also be applied to other problems to complement the selection of actions. Regarding the second heuristics, since LL is associated with the use in two phases of a probabilistic model, this simple heuristics could be exploited by model-based search algorithms.

References

[1] P. Toth and D. Vigo, The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, USA, 2002.

[2] L. Cruz, J. Delgado, J. González, J. Torres, H. Fraire and B. Arrañaga, An ant colony system to solve routing problems applied to the delivery of bottled products, Lecture Notes in Computer Science 4994 (2008) 329-338.

[3] L. Cruz, J. González, J. Delgado, B. Arrañaga and H. Fraire, A new approach to improve the ant colony system performance: learning levels, Lecture Notes in Computer Science 5572 (2009) 670-677.

[4] J. González, J. Delgado, L. Cruz, H. Fraire and A. Ramirez, Comparative analysis of hybrid techniques for an ant colony system algorithm applied to solve a real-world transportation problem, P. Melin, J. Kacprzyk, W. Pedrycz (Eds.), Soft Computing for Recognition Based on Biometrics Studies in Computational Intelligence, 2010, pp. 365-385.

[5] E. Mariano and F. Morales, DQL: A new updating strategy for reinforcement learning based on q-learning, Lecture Notes in Computer Science 2167 (2001), 324-335.

[6] C. Gómez, L. Cruz, M. Morales, J. González, O. Castillo, G. Rivera and P. Hernández, Variants of VRP to optimize logistics management problems, C. Ortiz Zezzatti, C. Chira, A. Hernandez, M. Basurto (Eds.), Logistics Management and Optimization through Hybrid Artificial Intelligence Systems, 2012, pp. 207-237.

[7] D. Pisinger and S. Ropke, A general heuristic for vehicle routing problems, Computers and Operations Research 34 (2007), 2403-2435.

[8] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, M. Maher, J. Puget (Eds.), Lecture Notes in Computer Science, Germany, 1998, pp. 417-431.

[9] D. Mester, O. Bräysy and W. Dullaert, A multi-parametric evolution strategies algorithm for vehicle routing problems, Expert Systems with Applications 32 (2007), 508-517.

[10] E. Prescott-Gagnon, G. Desaulniers, and L.M. Rousseau, A branch-and-price based large neighborhood search algorithm for the vehicle routing problem with time windows, Networks 54 (2009), 190-204.

[11] T. Hoshino, T. Kimura and T. Ikeguchi, Two simple local searches controlled by chaotic dynamics for the vehicle routing problems with time windows, In Abstract Proceedings of The Seventh Metaheuristics International Conference MIC 2007, Canada, 2007, pp.25-29.

[12] O. Bräysy, A reactive varaible neighborhood search for the vehicle routing problem with time windows, INFORMS Journal on Computing 15 (2003), 347-368.

[13] S. Thangiah, A site dependent vehicle routing problem with complex road constraints, Proceedings of International Conference on Metaheuristics for Optimization (2003), 1-17.

[14] G. Dantzig and J. Ramser, The truck dispatching problem, Management Science 6 (1959), 80-91.

[15] B. Kallehauge, Formulations and exact algorithms for the vehicle routing problem with time windows, Computers & Operations Research 35(7) (2008), 2307-2330.

[16] M. Dorigo and C. Blum, Ant colony optimization theory: a survey, Theoretical Computer Science 344 (2005), 243-278.

[17] M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Operations Research 35 (1987), 254-265.

[18] N. Park, H. Okano and H. Imai, A path-exchange-type local search algorithm for vehicle routing and its efficient search strategy, Journal of the Operations Research Society of Japan 43 (2000), 197-208.

[19] J. Dowling, E. Curran, R. Cunningham, V. Cahill, Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing, IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 35 (2005) 360-372.

[20] A. Haith, J. Krakauer, Model-based and model-free mechanisms of human motor learning, M. Richardson, M. Riley, K. Shockley (Eds.), Progress in Motor Control, Advances in Experimental Medicine and Biology 782 (2013) 1-21.

[21] H. Ping, A stochastic approximation interpretation for model-based optimization algorithms, Ph.D. thesis, Stony Brook University, USA, 2012.

[22] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo. Model-based search for combinatorial optimization: a critical survey, Annals of Operations Research 131 (2004) 373-395.

Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores

Laura Cruz-Reyes[1], Marcela Quiroz C.[1], Adriana C. F. Alvim[2], Héctor J. Fraire Huacuja[1], Claudia Gómez S.[1] y José Torres-Jiménez[3]

[1] Instituto Tecnológico de Ciudad Madero, México

[2] Universidad Federal do Estado do Rio de Janeiro, Brasil

[3] Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México

lauracruzreyes@itcm.edu.mx, qc.marcela@gmail.com, adriana@uniriotec.br, automatas2002@yahoo.com.mx, cggs71@hotmail.com, jtj@tamps.cinvestav.mx

Resumen. *En este artículo se aborda un problema clásico muy conocido por su aplicabilidad y complejidad: el empacado de objetos en contenedores (Bin Packing Problem, BPP). Para la solución de BPP se propone un algoritmo genético híbrido de agrupación denominado HGGA-BP. El algoritmo propuesto está inspirado en el esquema de representación de grupos de Falkenauer, el cual aplica operadores evolutivos a nivel de contenedores. HGGA-BP incluye heurísticas eficientes para generar la población inicial y realizar mutación y cruzamiento de grupos; así como estrategias híbridas para el acomodo de objetos que quedaron libres al aplicar los operadores grupales. La efectividad del algoritmo es comparable con la de los mejores del estado del arte, superando los resultados publicados para el conjunto de instancias hard28, el cual ha mostrado el mayor grado de dificultad para los algoritmos de solución de BPP.*

Palabras clave. *Metodologías computacionales, inteligencia artificial, solución de problemas, problema de empacado de objeto en contenedores, algoritmo genético híbrido.*

Efficient Hybrid Grouping Heuristics for the Bin Packing Problem

Abstract. *This article addresses a classical problem known for its applicability and complexity: the Bin Packing Problem (BPP). A hybrid grouping genetic algorithm called HGGA-BP is proposed to solve BPP. The proposed algorithm is inspired by the Falkenauer grouping encoding scheme, which applies evolutionary operators at the bin level. HGGA-BP includes efficient heuristics to generate the initial population and performs mutation and crossover for groups as well as hybrid strategies for the arrangement of objects that were released by the group operators. The effectiveness of the algorithm is comparable with the best state-of-the-art algorithms, outperforming the published results for the class of instances hard28, which has shown the highest difficulty for algorithms that solve BPP.*

Keywords. *Computer methodologies, artificial intelligence, problem solving, bin packing problem, hybrid genetic algorithm.*

1 Introducción

El problema de empacado de objetos en contenedores en una dimensión (one-dimensional Bin Packing Problem BPP), consiste en almacenar un conjunto de objetos de diferentes tamaños $N = \{1,\ldots,n\}$, o pesos, en el menor número de contenedores de tamaño fijo sin violar la capacidad de ningún contenedor.

BPP es un problema de optimización combinatoria NP-duro, considerado intratable debido a que demanda una gran cantidad de recursos para su solución [2,16]. La importancia de BPP radica en que tiene un extenso número de aplicaciones industriales y logísticas, y gran cantidad de problemas prácticos pueden ser modelados como problemas de empacado [8, 9, 10].

A lo largo de los últimos veinte años, en la búsqueda de buenas y mejores soluciones para BPP se han diseñado una gran variedad de algoritmos. Los resultados más destacados han sido obtenidos mediante el uso de algoritmos híbridos y metaheurísticos.

Martello y Toth [24] proponen un procedimiento de ramificación y poda (MTP) que incluye un criterio de dominación [23] para reducir el espacio de búsqueda. El criterio de dominación establece que: si es posible intercambiar un conjunto de objetos de un contenedor por un solo objeto del mismo peso que el conjunto, la solución puede ser mejorada debido a que será más fácil distribuir el conjunto de objetos pequeños en el resto de los contenedores, a encontrar un lugar para el objeto grande, debido a que su dominio es mayor. E. Falkenauer [13] propone un algoritmo genético híbrido de agrupación (HGGA) que utiliza: un esquema de representación de la solución por grupos de objetos y un método de optimización local inspirado en el criterio de dominación de Martello y Toth.

Scholl *et al.* [28] desarrollan un procedimiento híbrido (BISON) que combina una búsqueda tabú con un método de ramificación y poda, usando una estrategia dual que consiste en minimizar el llenado de los contenedores dado un número fijo de éstos. BISON incluye un nuevo esquema de ramificación basado en límites inferiores del valor de la solución óptima. Coffman *et al.* [7] presentan un estudio exhaustivo de las principales heurísticas deterministas para Bin Packing, incluyendo los análisis del peor caso de los algoritmos FFD y BFD [23]. Schwerin y Wäscher [29] proponen un nuevo límite inferior para BPP basado en problemas de corte y lo integran al método MTP de Martello y Toth [24], obteniendo resultados de calidad con su nuevo algoritmo (MTPCS). Fleszar y Hindi [14] introducen un nuevo algoritmo (Pertubation-MBS´) que incorpora una versión modificada de la heurística MBS de Gupta y Ho [18], una búsqueda de vecindad variable y límites inferiores, obteniendo buenos resultados. Levine y Ducatelle [20] desarrollan un método híbrido (HACO-BP) que implementa la metaheurística de optimización basada en colonia de hormigas; este algoritmo incluye una estrategia de búsqueda local y se apoya en el criterio de dominación de Martello y Toth [23]. Bhatia y Basu [5] presentan un algoritmo genético de agrupación multi-cromosómico (MGGA) y una nueva heurística de empacado (better-fit).

Alvim *et al*. [1] proponen una heurística de mejora híbrida (HI_BP) en la que retoman la estrategia dual de Scholl *et al*. [28], así como las técnicas de reducción de Martello y Toth [23], obteniendo los mejores resultados del estado del arte, hasta ese momento.

Singh y Gupta [31] desarrollan un enfoque evolutivo híbrido (C_BP) que combina un algoritmo genético de agrupación con una versión mejorada de la heurística Perturbation-MBS' de Fleszar y Hindi [14]. El desempeño obtenido es comparable con la estrategia HI_BP de Alvim *et al.* [1], sin embargo, el algoritmo C_BP es menos robusto.

A. Stawowy [32] propone una nueva estrategia evolutiva no especializada (ES) que incluye mutaciones inteligentes, una técnica de reducción del tamaño del problema y un esquema de representación basado en permutaciones con separadores de grupos. Los resultados obtenidos por el metaheurístico no hibridizado son comparables con los de algoritmos mucho más complicados. Rohlfshagen y Bullinaria [26] desarrollan un algoritmo genético inspirado en el proceso de formación de proteínas (ESGA) que obtiene resultados prometedores al ser comparado con otras estrategias.

Loh *et al.* [22] desarrollan un procedimiento (WA) que hace uso del concepto de recocido de pesos, el algoritmo propuesto es sencillo y fácil de implementar y obtiene un desempeño de alta calidad superando al mejor algoritmo del estado del arte, HI_BP [1].

Gómez-Meneses y Randall [17] presentan un procedimiento evolutivo híbrido de optimización extrema (HEO) que incorpora una búsqueda local para mejorar el empacado de los contenedores. R. Lewis [21] propone un algoritmo de escalado de colina (HC) que utiliza un esquema sencillo de mejora basado en el criterio de dominación y obtiene buenas soluciones superando, en algunos casos, el desempeño de algoritmos complejos.

El último trabajo en esta área fue presentado por Fleszar y Charalambous [15], los autores

proponen una modificación al procedimiento Perturbation-MBS' [14], que utiliza una estrategia para controlar el peso promedio de los objetos que son empacados en cada contenedor. La nueva heurística (Perturbation-SAWMBS) logra superar el desempeño reportado para los mejores algoritmos del estado del arte HI_BP, C_BP y WA [1, 22, 31]. Así mismo, los autores publican nuevos resultados para la heurística WA, los cuales contradicen el desempeño presentado inicialmente por sus autores.

En este artículo se presenta un algoritmo genético híbrido de agrupación para el problema de empacado de objetos en contenedores (*Hybrid Grouping Genetic Algorithm for Bin Packing*, *HGGA-BP*). Este algoritmo está inspirado en el esquema de representación de grupos propuesto por Falkenauer y Delchambre [12] para representar las soluciones como grupos de objetos asociados a contenedores y manipularlos con operadores grupales. HGGA-BP incorpora estrategias heurísticas eficientes, deterministas y aleatorias, para generar soluciones de calidad. Resultados experimentales muestran que HGGA-BP obtiene buenas soluciones en tiempos cortos, superando la efectividad de los mejores algoritmos del estado del arte en los casos de prueba más difíciles la clase hard28 [4].

2 Heurísticas híbridas de agrupación

De manera general, un algoritmo genético es una estrategia evolutiva poblacional donde un conjunto de soluciones son sometidas a un proceso de evolución que involucra: selección de individuos, cruzamiento y mutación, dando como resultado soluciones de mayor calidad. En esta sección se describen las heurísticas híbridas de agrupación propuestas para la creación y evolución de individuos. Las heurísticas propuestas incluyen diferentes algoritmos simples de empacado; su incorporación al algoritmo genético HGGA-BP se presenta en la siguiente sección.

2.1 Heurísticas de empacado

Las heurísticas de empacado son conocidos algoritmos deterministas, muy simples y rápidos, que han mostrado resultados satisfactorios en la solución de BPP [7, 19]. Se diferencian por la manera en que los objetos son tratados antes de ser acomodados y por la forma en que se elige el contenedor que almacenará cada objeto.

Primer Ajuste (First Fit, FF) [19]: Cada objeto en consideración es colocado en el primer contenedor que tenga suficiente capacidad disponible. En caso de que ningún contenedor parcialmente lleno pueda almacenarlo, el objeto es colocado dentro de un nuevo contenedor (vacío). Una variación a este método se establece cuando los objetos son tomados según el orden decreciente de sus pesos dicha variante es conocida como Primer Ajuste Decreciente (First Fit Decreasing, FFD).

Mejor Ajuste (Best Fit, BF) [19]: Cada objeto es acomodado en el contenedor más lleno que lo pueda almacenar, agregando nuevos contenedores cuando sea necesario. De igual manera que con FF, existe una variación, llamada Mejor Ajuste Decreciente (Best Fit Decreasing, BFD), que considera los objetos en orden decreciente de sus pesos.

Peor Ajuste (Worst Fit, WF) [19]: Contrario a Mejor Ajuste, cada objeto en consideración es almacenado en el contenedor menos lleno con capacidad residual suficiente para contenerlo. La variante que toma los objetos según el orden decreciente de sus pesos es conocida como Peor Ajuste Decreciente (Worst Fit Decreasing, WFD).

Best 3-Fit (B3F) [1]: Inicialmente se abre un número límite de contenedores. Luego, si existe un contenedor vacío, se selecciona y se coloca el objeto actual, de otro modo, se intenta llenar cada contenedor con objetos que no han sido seleccionados y que por pares suman la capacidad residual del contenedor. Para el resto de los objetos, el elemento actual es insertado en el contenedor más lleno en el que ajuste (como en BF). Si no existe un contenedor con capacidad suficiente un nuevo contenedor es agregado a la solución.

Heurísticas aleatorias: Las heurísticas FF, BF, WF y B3F son estrategias deterministas y son la base del proceso de generación de individuos en el algoritmo HGGA-BP pues las soluciones obtenidas con ellas son individuos de buena calidad [1, 7,19]. Para obtener un grupo de individuos diversos, soluciones distintas son

obtenidas al manejar órdenes aleatorios del conjunto de objetos a acomodar. Las versiones aleatorias de las heurísticas de empacado han sido denominadas: *FF_Aleatorio*, *BF_Aleatorio*, *WF_Aleatorio* y *B3F_Aleatorio*, respectivamente.

2.2 Construcción de individuos

Todas las estrategias de construcción de individuos inician con el cálculo de un *límite inferior* del número de contenedores LM, que es igual al número de objetos de tamaño mayor que la mitad de la capacidad del contenedor. LM representa el número de objetos "grandes" que no podrían combinarse entre sí. Una vez calculado dicho límite, los LM objetos más pesados son almacenados cada uno en un contenedor y el resto de los objetos son acomodados con alguna de las técnicas para empacado de objetos descritas en la sección anterior.

2.3 Generación de la población

Con el método *PI_ D-A* la población inicial se forma con dos tipos de individuos: deterministas y aleatorios. La parte determinista se conforma de cuatro individuos creados con FFD, BFD, WFD, y B3FD. La población aleatoria se compone de individuos creados con las estrategias FF_Aleatorio, BF_Aleatorio, WF_Aleatorio y B3F_Aleatorio.

El cálculo de la *Aptitud* de cada individuo de la población es realizado utilizando la función de costo introducida por Falkenauer y Delchambre, que evalúa el promedio de llenado de los contendores que conforman una solución [12]. Dado que, según esta función, las soluciones que poseen un mayor promedio de llenado son mejores, el objetivo del algoritmo HGGA-BP es maximizar el valor de las aptitudes de los individuos que conforman la población. La Ecuación 1 define el cálculo de la aptitud, donde m es el número de contenedores utilizado en la solución, S_i es la suma de los tamaños de los objetos en el contenedor i y c es la capacidad del contenedor.

$$Aptitud = \frac{\sum_{i=1}^{m} \left(S_i / c\right)^2}{m} \quad (1)$$

2.4 Operadores genéticos para grupos

Un esquema de codificación por agrupación para BPP fue propuesto por Falkenauer y Delchambre [12]. En este tipo de estructura cromosómica los genes del individuo están representados por contenedores y no por objetos. Para manipular grupos de objetos en el algoritmo HGGA-BP se propone utilizar operadores genéticos especiales y heurísticas de reacomodo de objetos que quedan libres al aplicar dichos operadores.

2.4.1 Heurísticas de reacomodo de objetos

En el proceso de solución, los operadores genéticos generan y modifican individuos, producto de este proceso algunos contenedores son vaciados y existen objetos libres que necesitan ser reinsertados en la solución. En este trabajo se proponen dos nuevos procedimientos de reacomodo.

Reacomodo_Voraz: Este procedimiento recorre cada contenedor intentando intercambiar uno de sus objetos por un objeto libre de mayor tamaño, sin violar su capacidad. Después de recorrer todos los contenedores, los objetos libres se reacomodan utilizando la heurística de empacado BFD.

Reacomodo_por_Pares: Esta estrategia consta de dos etapas: Generar una permutación aleatoria del orden de los contenedores y recorrer por pares todos los objetos de cada contenedor intentando intercambiar el par de objetos por una mejor opción. Las alternativas de sustitución son dos: a) Sustituir el par de objetos del contenedor por un objeto libre de peso igual o mayor que no sobrepase la capacidad del contenedor y b) Sustituir el par de objetos del contenedor por un par de objetos diferentes que sumen lo mismo o que llenen más el contenedor. Finalmente, si existen objetos libres, la heurística BF_Aleatorio es aplicada para completar la solución.

2.4.2 Cruzamiento, mutación y eliminación

Para el algoritmo HGGA-BP, se implementa una modificación del operador de cruzamiento propuesto por Falkenauer y Delchambre [12] y se proponen un operador de cruzamiento y dos operadores de mutación que tratan de mejorar la aptitud de los individuos. Adicionalmente, se propone un operador de sustitución para diversificar la población al sustituir individuos con aptitudes repetidas.

Cruzamiento_BFD: En HGGA-BP se implementa el cruzamiento para grupos propuesto por Falkenauer y Delchambre [12], con la diferencia de que los objetos libres son acomodados con la heurística BFD en lugar de FFD. Cruzar individuos implica determinar los contenedores que deben ser heredados de padres a hijos. Aquellos contenedores que sobreviven el proceso evolutivo se caracterizan por ser dominantes sobre otros. En esta técnica, se crean, de manera aleatoria, dos puntos de corte en cada individuo a cruzar, que dividen cada solución en tres segmentos. El nuevo individuo (denominado hijo) está formado por el primer segmento de contenedores del primer padre, el segundo segmento de contenedores del segundo padre y el resto de los contenedores del primer padre, eliminándose contenedores con objetos duplicados y reacomodando los objetos libres con la heurística BFD.

Mutación_RV: Este operador de mutación elimina un porcentaje de los contenedores menos llenos e intenta reacomodar los objetos libres utilizando la heurística Reacomodo_Voraz. Para cada individuo a mutar, el número de contenedores a vaciar se calcula en relación al tamaño de la solución, dependiendo de qué tan grande o pequeño sea el individuo.

Cruzamiento_RpP: Este nuevo método de cruzamiento incorpora explotación de buenas soluciones. En esta estrategia, los individuos a cruzar son seleccionados de manera aleatoria tomando en cuenta sólo soluciones de buena calidad (la mejor mitad de la población). A diferencia del Cruzamiento_BFD, en ambos padres, cada contenedor es considerado como un punto de corte. En cada punto de corte se procesan un contenedor del primer padre y un contendor del segundo padre. El contenedor que se encuentra lleno es el primero en ser heredado a la nueva solución, para luego heredar el otro contenedor; si ambos o ningún contenedor se encuentra lleno, se da preferencia al primer padre. Los contenedores que incluyen objetos duplicados son eliminados y los objetos libres son reinsertados aplicando Reacomodo_por_Pares.

Mutación_RpP: Un segundo método de mutación es incluido para individuos con buenas aptitudes y consiste principalmente en: calcular el número de contenedores a vaciar de acuerdo al tamaño de la solución y al número de contenedores incompletos y reacomodar los objetos libres utilizando Reacomodo_por_Pares.

Eliminación_por_Sustitución: Este nuevo operador es utilizado para eliminar individuos con aptitudes duplicadas en la población, siguiendo el enfoque de substitución. El operador previene la convergencia prematura del algoritmo a regiones sub-óptimas, al mismo tiempo que conserva la variabilidad en la población. En cada generación, los individuos que presentan un valor de aptitud repetido son sustituidos por nuevos individuos: n_{b3f} generados con la heurística de empacado B3F_Aleatorio y el resto con FF_Aleatorio.

3 Algoritmo genético HGGA-BP

La Tabla 1 presenta el algoritmo HGGA-BP propuesto en este trabajo. El proceso comienza generando una población inicial con el método *PI_D-A*, el cual crea un conjunto de individuos (soluciones) con heurísticas deterministas y aleatorias (Línea 3). En cada iteración, dado un porcentaje de cruzamiento y con base en la aptitud, se selecciona un conjunto de individuos para aplicarles el operador *Cruzamiento_BFD* (Líneas 5-7). Posteriormente, según el porcentaje de mutación, los peores individuos de la población sufren pequeñas alteraciones genéticas usando *Mutación_RV*; el objetivo es mejorar su aptitud (Línea 8).

En una segunda etapa se procede a intensificar la búsqueda, con el objetivo de perfeccionar el empacado de las mejores soluciones. Primero, los individuos más aptos son clonados para obtener nuevos individuos mediante el operador *Mutación_RpP* (Líneas 10-12). Luego, los individuos con aptitud repetida

Tabla 1. Algoritmo genético híbrido de agrupación para BPP: HGGA-BP

```
1   Inicio
2      Inicializar parámetros: max_gen, L2, gen1, gen2
3      Generar la población inicial con PI_D-A
4      mientras generación < max_gen y mejor_solución > L2
5         Seleccionar individuos a cruzar en proporción a su aptitud
6         Aplicar Cruzamiento_BFD y generar nuevos_individuos
7         Sustituir los peores individuos por los nuevos_individuos
8         Aplicar Mutación_RV a los peores individuos
9         si generación > gen1
10           Clonar los mejores individuos de la población
11           Aplicar Mutación_RpP a los individuos clonados
12           Sustituir los peores individuos por los individuos clonados
13        fin si
14        Aplicar Eliminación_por_Sustitución a individuos repetidos
15        si generación > gen2
16           Seleccionar los mejores individuos para cruzar
17           Aplicar Cruzamiento_RpP para generar nuevos_individuos
18           Sustituir los peores individuos por los nuevos_individuos
19        fin si
20        Registrar la mejor_solución
21     fin mientras
22  fin procedimiento
```

son reemplazados por nuevos individuos al aplicar el operador *Eliminación_por_Sustitución* (Línea 14). Finalmente, los mejores individuos de la población son apareados para generar individuos de alta calidad producto del operador *Cruzamiento_RpP* (Líneas 16-18).

Al término de cada generación la mejor solución de la población es registrada (Línea 20). El resultado final del algoritmo es el individuo más apto de todo el proceso evolutivo. El algoritmo iterará un máximo número generaciones *max_gen*, y se detendrá antes en caso de encontrar una solución cuyo tamaño coincidacon el límite L_2 de Martello y Toth [23].

En el cálculo de L_2, la idea principal es encontrar un valor ε, entre cero y la mitad de la capacidad del contenedor, que maximice el resultado obtenido al dividir el conjunto de pesos en objetos grandes (con peso mayor que $1 - \varepsilon$) y pequeños (con peso menor que ε). Para cada valor de ε, se asume que los objetos grandes son almacenados en diferentes contenedores y los objetos pequeños son utilizados para terminar de llenar los contenedores donde se almacenaron los objetos grandes.

4 Experimentos computacionales

HGGA-BP fue desarrollado en lenguaje C++ y las experimentaciones se ejecutaron en una computadora personal con procesador Intel Xenon a 1.866 Ghz y 4GB de RAM, sobre el sistema operativo Windows XP profesional SP2.

Con el fin de demostrar la robustez del algoritmo, para cada caso de prueba se realizaron 30 ejecuciones del algoritmo con diferentes semillas de números aleatorios. Los valores reportados como soluciones finales de cada instancia representan los promedios de las 30 soluciones generadas.

El algoritmo se configuró mediante un estudio experimental sobre el desempeño de las heurísticas propuestas [25]. El tamaño de la población es de 150 individuos. La población inicial se crea con diferentes heurísticas (1 con *FFD*, 1 con *BFD*, 1 con *WFD*, 1 con *B3FD*, 1 con *B3F_Aleatorio*, 56 con *FF_Aleatorio*, 60 con *BF_Aleatorio* y 30 con *WF_Aleatorio*). El máximo

número de generaciones max_gen es 100. En cada generación se aplican los operadores *Cruzamiento_BFD* y *Mutación_RV* al 30% de los individuos de la población. En *Mutación_RV*, el intervalo de contenedores a vaciar es 10-35%. Después de la décima generación (gen_1=10), el operador *Mutación_RpP* es aplicado, en cada generación, para generar 15 nuevos individuos que sustituyen a las 15 peores soluciones y para introducir pequeñas modificaciones en 15 buenos individuos. *Cruzamiento_RpP* se utiliza para generar 35 nuevos individuos que sustituyen a las 35 peores soluciones y es aplicado después de la generación número 20 (gen_2=20). El operador *Eliminación_por_Sustitución* es aplicado en cada generación para generar n_{b3f}=3 individuos con la heurística de empacado *B3F_Aleatorio* y el resto con la estrategia *FF_Aleatorio*.

4.1 Instancias de prueba

El desempeño de los algoritmos para BPP ha sido evaluado con diferentes clases de instancias de referencia consideradas retadoras. Los casos de prueba, elegidos por diferentes investigadores para comparar las cualidades de sus estrategias con las de otros algoritmos del estado del arte, conforman un conjunto de 1615 casos estándar reconocidos por la comunidad científica. Las 1615 instancias de prueba se encuentran en sitios de Internet reconocidos [3, 6, 11, 27], sus soluciones óptimas son conocidas [1, 4, 27]. Las instancias consideradas se han dividido en cuatro grupos, tomando en cuenta su origen y los sitios de los que fueron obtenidas.

El primer grupo consiste en dos conjuntos de instancias propuestas por E. Falkenauer [13].

***Uniform* (*u*)** [3]: Conjunto de 80 instancias identificadas con la letra *u* debido a que su principal característica es que los pesos de los objetos están uniformemente distribuidos entre 20 y 100. La capacidad del contenedor *c* es de 150 y existen cuatro clases de casos cada uno con *n* = 120, 250, 500 y 1000 objetos. Cada clase posee 20 instancias identificadas respectivamente por *u*_120, *u*_250, *u*_500 y *u*_1000. El valor de la solución óptima para cada una de estas instancias es conocido [1].

***Triplets* (*t*)** [3]: Conjunto de 80 instancias difíciles, identificadas con la letra *t*. Su nombre se debe a que las instancias fueron construidas con una solución óptima conocida de *n*/3 contenedores, de tal forma que cada contenedor de la solución óptima debe almacenar exactamente tres objetos que lo llenan completamente. El tamaño de las instancias es de *n* = 60, 120, 249 y 501, definiéndose así cuatro clases, el tamaño del contenedor *c* es de 100, mientras que los pesos están distribuidos entre 25 y 50. Cada clase posee 20 instancias identificadas respectivamente por *t*_60, *t*_120, *t*_249 y *t*_501.

El segundo grupo está formado por tres conjuntos de casos de prueba introducidos por Scholl *et al.* [28]. En cada conjunto, las diferentes clases de problemas fueron creados variándose el número de objetos *n*, la capacidad del contenedor *c* y los posibles pesos de los objetos.

***Data Set 1* (*set_1*)** [27]: Construidas de forma similar que algunas instancias propuestas por Martello y Toth [24] que resultaron difíciles. Son un conjunto de 720 instancias denotadas con *n*-*c*-*w* por los datos que manejan: *n* = 50, 100, 200 y 500, capacidad del contenedor *c* = 100, 120, 150 y los pesos *w* generados uniformemente en intervalos de [1,100], [20,100] y [30,100]. La combinación de los diferentes parámetros resulta en 36 clases, cada clase contiene 20 instancias.

***Data Set 2* (*set_2*)** [27]: Incluye 480 instancias con pesos generados con una distribución uniforme, denotadas por *n*-*w*-*b*. Cada sigla representa la configuración de un parámetro de entrada: número de objetos *n* con valores de 50, 100, 200 y 500, la capacidad de los contenedores *c* es 1000. Con el objetivo de generar instancias cuyo número medio de objetos por contenedor variara entre tres y nueve, se consideraron otros dos parámetros: el peso medio deseado *w* con valores c/3, c/5, c/7, c/9, y una desviación máxima de dicho peso *b* = 20%, 50%, 90%. Por ejemplo cuando *w* = c/5 y *b*= 50%, los pesos de los objetos fueron generados de manera aleatoria con una distribución uniforme en el intervalo discreto [100, 300]. Al Combinar las características anteriores se cuenta con 48 clases, cada una con 10 instancias.

***Data Set 3* (*set_3*)** [27]: Conjunto formado por una única clase con diez instancias consideradas difíciles. Cada instancia posee 200 objetos, capacidad de contenedor 100000 y los pesos

están distribuidos uniformemente de manera dispersa entre 20000 y 35000.

El tercer grupo incluye tres clases de instancias sugeridas por Schwerin, Wäscher y Gau. Las primeras dos clases se han definido como ffd-hard (difíciles de resolver por la heurística FFD) [29,30]. La tercera clase también ha sido considerada difícil [33].

Was_1 [11]: Formado por 100 instancias de capacidad *c* = 1000. Cada instancia posee *n* = 100 objetos. Los pesos de los objetos varían entre 150 y 200.

Was_2 [11]: Incluye 100 instancias con capacidad del contenedor *c* = 1000, cada instancia contiene 120 objetos con pesos entre 150 y 200.

Gau_1 [11]: Son 17 instancias con características variadas. La capacidad de los contenedores es igual a 10000, el número de objetos *n* varía de 57 a 239 y los pesos se encuentran distribuidos entre 2 y 7332.

El grupo cuatro contiene 28 instancias difíciles que han sido consideradas en problemas de corte de una dimensión (one-dimensional Cutting Stock Problem).

Hard28 [6]: Incluye las 28 instancias más difíciles usadas por G. Belov [4], dichas instancias no pudieron ser resueltas por algoritmos de corte ni por métodos de reducción. El número de objetos *n* varía entre 160 y 200, los pesos de los objetos varían entre 1 y 800 y la capacidad del contenedor *c* es 1000.

Los últimos dos conjuntos de instancias, gau_1 y hard28, han mostrado poseer casos de prueba de alto grado de dificultad [4, 15]. Destacándose el conjunto hard28, para el que existe un mayor número de instancias que los algoritmos no logran solucionar de manera óptima.

4.2 Resultados experimentales

Para investigar la efectividad del algoritmo HGGA-BP, se realizó un estudio comparativo de los resultados obtenidos por este algoritmo con aquellos obtenidos por los mejores algoritmos reportados en la literatura: HI_BP [1], WA [22] y Perturbation-SAWMBS [15].

La Tabla 2 detalla, para cada clase de instancias, el número de casos de prueba y, para cada procedimiento, el número de instancias para las que el algoritmo alcanza la solución óptima conocida (columna ópt.), el radio teórico promedio de las soluciones obtenidas (columna

Tabla 2. Resultados obtenidos por los mejores algoritmos heurísticos aplicados a BPP (tiempo en seg.)

Clase	no. inst.	HI_BP [1]			WA [22]			Pert.-SAWMBS [15]			HGGA-BP [Este trabajo]		
		ópt.	radio	tiempo	ópt.	radio	tiempo	ópt.	radio	tiempo	ópt.	radio	tiempo
u	80	80	1	0.03	71	1.0011	0.28	79	1.0001	0.00	79	1.0001	1.38
t	80	80	1	0.98	0	1.0232	0.15	80	1	0.00	80	1	3.42
set_1	720	720	1	0.19	703	1.0003	0.25	720	1	0.01	718	1.0000	2.08
set_2	480	480	1	0.01	468	1.0006	0.04	480	1	0.00	480	1	0.67
set_3	10	10	1	4.60	9	1.0017	0.08	10	1	0.16	9	1.0017	6.56
was_1	100	100	1	0.02	100	1	0.00	100	1	0.00	100	1	0.02
was_2	100	100	1	0.02	100	1	0.02	100	1	0.01	100	1	0.52
gau_1	17	12	1.0122	0.60	13	1.0122	0.02	16	1.0025	0.04	15	1.0047	1.10
hard28	28	5	1.0119	**≥** 0.48	5	1.0119	**≥** 0.59	5	1.0119	**≥** 0.24	**8***	1.0106	4.31
Total	**1615**	**1587**	**1.0026**	**0.77**	**1469**	**1.0057**	**0.16**	**1590**	**1.0016**	**0.05**	**1589**	**1.0019**	**2.22**

HGGA-BP	1.866 GHz Intel Xenon
HI_BP	1.7 GHz Pentium IV
WA	2.33 GHz Core2
Pert.-SAWMBS	2.33 GHz Core2

≥ Un incremento significativo en el tiempo de ejecución no mejoró los resultados

*** HGGA-BP supera a los mejores algoritmos en el conjunto Hard28**

radio), así como el tiempo promedio de ejecución medido en segundos (columna tiempo). El *radio teórico* es una medida de calidad, y representa la razón entre el número de contenedores de la solución encontrada por el algoritmo y el número de contenedores utilizados en la solución óptima, cuando la solución obtenida es óptima el radio teórico será igual a 1 y será mayor en otro caso.

Los resultados del algoritmo HI_BP fueron obtenidos al ejecutar el código en lenguaje C proporcionados por la Dra. Adriana C. F. Alvim, los cuales coinciden con lo publicado en su artículo [1]. Los resultados de los procedimientos WA y Perturbation-SAWMBS fueron obtenidos de la publicación de Fleszar y Charalambous [15], donde se destaca que los resultados anunciados por Loh *et al.* [22] para la heurística WA son incorrectos. Con la finalidad de verificar los resultados de la heurística WA se realizó la codificación de dicho algoritmo, obteniendo resultados similares a los reportados por Fleszar y Charalambous [15], por lo que coincidimos con dichos autores en que los resultados publicados por Loh *et al.* [22] son incorrectos.

En todos los casos, cuando los algoritmos no logran encontrar la solución óptima, la solución encontrada posee sólo un contenedor extra respecto a la solución óptima.

En la misma Tabla 2 se han resaltado los resultados obtenidos para el conjunto de instancias hard28, el cual contiene las instancias hasta ahora más difíciles, para los algoritmos conocidos. Puede observarse que la estrategia propuesta en este trabajo, HGGA-BP, supera la efectividad de los mejores algoritmos del estado del arte, al resolver un mayor número de instancias. Para este conjunto de instancias, Fleszar y Charalambous [15] señalan que incluso incrementando el número de iteraciones de su algoritmo Perturbation-SAWMBS de 2,000 a 100,000 no es posible mejorar las soluciones encontradas, destacando la dificultad de estas instancias y recomendando su uso para futuros estudios de heurísticas de BPP. Resultados similares fueron obtenidos al incrementar el tiempo de búsqueda del procedimiento HI_BP.

La Tabla 3 presenta los resultados detallados del algoritmo HGGA-BP sobre el conjunto hard28, en esta tabla se incluye, para cada instancia, el valor del límite inferior de contenedores L_2, la solución óptima y las soluciones mínima, máxima y promedio obtenida por el algoritmo HGGA-BP en 30 corridas.

Los resultados más relevantes de la Tabla 3 se resaltaron con color de fondo en escala de grises y letras en negritas.

Color gris obscuro: resalta las cinco instancias de prueba (hBPP814, hBPP359, hBPP716, hBPP119 y hBPP175) que son solucionadas de manera óptima por los mejores algoritmos del estado del arte.

Color gris claro: resalta los tres casos de prueba (hBPP640, hBPP531 y hBPP814) en los que HGGA-BP supera a dichos algoritmos.

Sólo letras negritas: resalta tres casos (hBPP360, hBPP742 y hBPP47) para los cuales HGGA-BP obtiene la solución óptima en alguna de las 30 corridas.

Las 17 instancias restantes son consideradas muy difíciles pues el algoritmo HGGA-BP no logra obtener la solución óptima en ninguna corrida. Sería interesante conocer las características que marcan la diferencia en el grado de dificultad dentro de este conjunto de instancias para poder definir estrategias adecuadas que permitan obtener su solución óptima.

Para fundamentar las conclusiones obtenidas al comparar los resultados de los mejores algoritmos del estado del arte con HGGA-BP se efectuó un estudio estadístico comparativo de resultados usando la prueba no paramétrica *T* de Wilcoxon (Wilcoxon signed rank test) con un nivel de significancia de 0.05. El estudio fue realizado con el objetivo de comparar el desempeño de HGGA-BP con los mejores algoritmos: HI_BP [1] y Perturbation-SAWMBS [15]. Las pruebas fueron realizadas con el procedimiento UNIVARIATE de la herramienta SAS 9.2 para Windows.

Para el conjunto total de instancias la prueba mostró que no existen diferencias significativas en el desempeño de los tres algoritmos. Así mismo, al analizar los conjuntos de instancias u, t, set_1, set_2, set_3, was_1, was_2 y gau_1 tampoco se observaron diferencias significativas en el desempeño de los tres algoritmos. Sin embargo, para el conjunto hard28 la prueba *T* de Wilcoxon mostró que HGGA-BP es superior a los

Tabla 3. Resultados de HGGA-BP con hard28

Instancias	L_2	Óptimo	Min	Max	Prom
hBPP14	61	62	62	62	62
hBPP832	60	60	61	61	61
hBPP40	59	59	60	60	60
hBPP360	**62**	**62**	**62**	**63**	**63**
hBPP645	58	58	59	59	59
hBPP742	**64**	**64**	**64**	**65**	**65**
hBPP766	62	62	63	63	63
hBPP60	63	63	64	64	64
hBPP13	67	67	68	68	68
hBPP195	64	64	65	65	65
hBPP709	67	67	68	68	68
hBPP785	68	68	69	69	69
hBPP47	**71**	**71**	**71**	**72**	**72**
hBPP181	72	72	73	73	73
hBPP359	75	76	76	76	76
hBPP485	71	71	72	72	72
hBPP640	74	74	74	75	74
hBPP716	75	76	76	76	76
hBPP119	76	77	77	77	77
hBPP144	73	73	74	74	74
hBPP561	72	72	73	73	73
hBPP781	71	71	72	72	72
hBPP900	75	75	76	76	76
hBPP175	83	84	84	84	84
hBPP178	80	80	81	81	81
hBPP419	80	80	81	81	81
hBPP531	83	83	83	83	83
hBPP814	81	81	81	82	81

otros procedimientos, al mejorar la solución en seis instancias: hBPP640, hBPP531, hBPP814, hBPP360, hBPP742 y hBPP47 (ver Tabla 3).

5 Conclusiones y trabajo futuro

Desarrollamos un nuevo algoritmo genético híbrido, denominado HGGA-BP, para resolver el problema clásico de empacado de objetos en contenedores. HGGA-BP está inspirado en el esquema de representación propuesto por Falkenauer [13] para problemas de agrupación como BPP.

El metaheurístico híbrido HGGA-BP conjunta diferentes heurísticas de solución de BPP y crea un balance entre exploración y explotación del espacio de búsqueda con el fin de obtener las mejores soluciones. El tiempo de ejecución del algoritmo propuesto es muy corto al ser comparado con el requerido por otras estrategias poblacionales [13, 20, 32]. La calidad de las soluciones encontradas por HGGA-BP es similar a la obtenida por los mejores algoritmos del estado del arte, superando los resultados mostrados por las mejores estrategias de BPP sobre el conjunto de instancias hasta ahora más difíciles (hard28).

La revisión de los resultados obtenidos por los mejores algoritmos de solución de BPP reveló que aún existen instancias de la literatura que presentan un alto grado de dificultad para las cuales las estrategias incluidas en los algoritmos no parecen conducir a mejores soluciones. Al realizar el análisis de la literatura, se observó que ninguno de los algoritmos del estado del arte ha sido analizado para explicar el porqué de su buen o mal desempeño. Por otro lado, tampoco existe un análisis general sobre la estructura y dificultad de las instancias de BPP. Es importante identificar cuáles son las características que distinguen a las instancias de BPP y que pueden ser causa de su grado de dificultad. Así mismo, es necesario entender el comportamiento de los algoritmos y evaluar las estrategias que les permiten alcanzar su desempeño.

Referencias

1. **Alvim, A.C.F., Ribeiro, C.C., Glover, F., & Aloise, D.J. (2004).** A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2), 205–229.
2. **Baase, S. & Gelder, A.V. (2000).** *Computer Algorithms, Introduction to Design and Analysis* (3rd ed.). Reading, Mass.: Addison-Wesley Longman.
3. **Beasley, J.E. (1990).** OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11), 1069–1072. Retrieved from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html.
4. **Belov, G. (1992).** Problems, Models and Algorithms in One and Two Dimensional Cutting. Ph.D. Thesis, Technischen Universitat Dresden, St. Petersburg, Russland.
5. **Bhatia, A.K. &Basu, S.K. (2004).** Packing Bins Using Multi-chromosomal Genetic Representation and Better-Fit Heuristic. *Neural Information*

Processing, Lecture Notes in Computer Science, 3316, 181–186.

6. **CaPaD.** Cutting and Packing at Dresden University: Test instances & results. Retrieved fromhttp://www.math.tu-dresden.de/~capad/cpd-ti.html#pmp%2024.
7. **Coffman, E.G., Garey, M.R., & Johnson D.S. (1997).** Approximation algorithms for bin packing: a survey. In Hochbaum D.S. (Ed.), *Approximation algorithms for NP-hard problems* (46–93). Boston: PWS Publishing.
8. **Crainic, T.G., Perboli, G., Rei, W., & Tadei, R. (2011).** Efficient Lower Bounds and Heuristics for the Variable Cost and Size Bin Packing Problem. *Computers and Operations Research*, 38(11), 1474–1482.
9. **Cruz, L., Nieto-Yáñez, D.M., Rangel-Valdez, N., Herrera, J.A., González, J., Castilla, G., & Delgado-Orta, J.F. (2007).** DiPro: An Algorithm for the Packing in Product Transportation Problems with Multiple Loading and Routing Variants. *6th Mexican International Conference on artificial Intelligence (MICAI2007:Advances in Artificial Intelligence)*, Lecture Notes in Artificial Intelligence, 4827, 1078–1088.
10. **Di Natale, M. & Bini, E. (2007).** Optimizing the FPGA implementation of HRT systems. *13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07),* Bellevue, Washington, USA, 22–31.
11. **ESICUP.** Euro Especial Interest Group on Cutting and Packing. One Dimensional Cutting and Packing Data Sets. Retrieved from http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1%2023
12. **Falkenauer, E. & Delchambre, A. (1992).** A Genetic Algorithm for Bin Packing and Line Balancing. *1992 IEEE International Conference on Robotics and Automation*, Nice, France, 2, 1186–1192.
13. **Falkenauer, E. (1996).** A Hybrid Grouping Genetic Algorithm for Bin Packing. *Journal of Heuristics*, 2(1), 5–30.
14. **Fleszar, K. & Hindi, K. (2002).** New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, 29(7), 821–839.
15. **Fleszar, K. & Charalambous, C. (2011).** Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European Journal of Operational Research*. 210(2), 176–184.
16. **Garey, M.R. & Johnson, D.S. (1979).** *Computers and Intractability: A Guide to the Theory of NP Completeness*. San Francisco: W. H. Freeman
17. **Gómez-Meneses, P. & Randall, M.A. (2009).** Hybrid Extremal Optimisation Approach for the Bin Packing Problem. *Artificial Life: Borrowing from Biology, 4th Australian Conference, ACAL 2009, Lecture Notes in Computer Science*, 5865, 242–251.
18. **Gupta, J.N.D. & Ho, J.C. (1999).** A new heuristic algorithm for the one-dimensional bin-packing problem. *Production Planning & Control: The Management of Operations*, 10(6), 598–603. Retrieved from http://www.math.tu-dresden.de/~capad/cpd-ti.html#pmp%2024
19. **Johnson, D.S. (1974).** Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3), 272–314.
20. **Levine, J. & Ducatelle, F. (2004).** Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7), 705–716.
21. **Lewis, R. (2009).** A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Computers & Operations Research*, 36(7), 2295–2310.
22. **Loh, K.H., Golden, B., &Wasil, E. (2008).** Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7), 2283–2291.
23. **Martello, S. &Toth, P. (1990).** *Knapsack Problems: Algorithms and Computer Implementations*. New York: J. Wiley & Sons.
24. **Martello, S. & Toth, P. (1990).** Lower Bounds and Reduction Procedures for the Bin Packing Problem. *Discrete Applied Mathematics,* 28(1), 59–70.
25. **Quiroz, M. (2009).** Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP. Tesis de maestría, Instituto Tecnológico de Cd. Madero, Tamaulipas, México.
26. **Rohlfshagen, P. & Bullinaria, J.A. (2010).**Nature inspired genetic algorithms for hard packing problems. *Annals of Operations Research*, 179(1), 393–419.
27. **Scholl, A. & Klein, R. (s.f.).** Bin Packing. Retrieved from http://www.wiwi.uni-jena.de/Entscheidung/binpp/
28. **Scholl, A., Klein, R., & Jürgens, C. (1997).** BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers and Operations Research*, 24(7), 627–645.

29. **Schwerin, P. & Wäscher, G. (1999).** A new lower bound for the bin-packing problem and its integration to MTP. *Pesquisa Operacional*, 19(2), 111–130.
30. **Schwerin, P. & Wäscher, G. (1997).**The bin packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4(5-6), 337–389.
31. **Singh, A. & Gupta, A.K. (2007).** Two heuristics for the one-dimensional bin-packing problem. *OR Spectrum*, 29(4), 765–781.
32. **Stawowy, A. (2008).** Evolutionary based heuristic for bin packing problem. *Computers & Industrial Engineering*, 55(2), 465–474.
33. **Wäscher, G. & Gau, T. (1996).** Heuristics for the integer one-dimensional cutting stock problem: A computational study. *OR Spektrum*, 18(3), 131–144.

Laura Cruz-Reyes. Recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico y de Estudios Superiores de Monterrey, México, en 1999 y el grado de doctora en ciencias de la computación del Centro Nacional de Investigación y Desarrollo Tecnológico, México, en el 2004. Es una profesora de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen redes complejas, aprendizaje automático y técnicas de optimización.

Marcela Quiroz C. Nació en México en 1984. En el 2009 recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico de Ciudad Madero, México. Actualmente es una estudiante de doctorado en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen técnicas de optimización, aprendizaje automático, algoritmia experimental y explicaciones del desempeño algorítmico.

Adriana C. F. Alvim. Profesora de tiempo completo en la Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Brasil. Recibió el grado de doctora en ciencias de la computación de la Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) en el 2003. Sus intereses de investigación incluyen heurísticas y metaheurísticas para problema de optimización combinatoria duros.

Héctor J. Fraire Huacuja. En 1988 recibió el grado de maestro en ciencias de la Información de la Universidad Autónoma de Nuevo León, México, y el grado de doctor en ciencias de la computación del Centro Nacional de Investigación y Desarrollo Tecnológico, México en el 2005. Actualmente es un profesor de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen optimización heurística y aprendizaje automático.

Claudia Gómez S. En el 2000 recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico de León, México y en el 2010 recibió el grado de doctora en ciencias de la computación del Instituto Politécnico Nacional, México. Actualmente es una profesora de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen redes complejas, optimización y agentes inteligentes.

José Torres-Jiménez. Estudió ingeniería electrónica en el Instituto Tecnológico de Nuevo Laredo y recibió los grados de maestro en ciencias de la computación y doctor en ciencias de la computación del Instituto Tecnológico y de Estudios Superiores de Monterrey, México. Actualmente es un profesor de tiempo completo en el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México. Sus áreas de interés incluyen optimización combinatoria, sistemas de bases de datos, ingeniería de software e inteligencia artificial.

Artículo recibido el 03/01/2011; aceptado el 08/11/2011.

Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080405

Many-Objective Portfolio Optimization of Interdependent Projects with 'a Priori' Incorporation of Decision-Maker Preferences

Laura Cruz[1], *Eduardo Fernandez*[2], *Claudia Gomez*[1], *Gilberto Rivera*[1,*] *and Fatima Perez*[3]

[1] Postgraduate & Research Division, Madero Institute of Technology, 89440, Tamaulipas, Mexico
[2] Faculty of Civil Engineering, Autonomous University of Sinaloa, 80040, Sinaloa, Mexico
[3] Department of Applied Economics (Mathematics), University of Malaga, 29071, Malaga, Spain

Received: 6 Jul. 2013, Revised: 8 Oct. 2013, Accepted: 9 Oct. 2013
Published online: 1 Jul. 2014

Abstract: Project portfolio selection is one of the most important problems faced by any organization. The decision process involves multiple conflicting criteria, and has been commonly addressed by implementing a two-phase procedure. The first step identifies the efficient solution set; the second step supports the decision maker in selecting only one portfolio solution from the efficient set. However, several recent studies show the advantages gained by optimizing towards a region of interest (according to the decision maker's preferences) instead of approximating the complete Pareto set. However, these works have not faced synergism and its variants, such as cannibalization and redundancy. In this paper we introduce a new approach called *Non-Outranked Ant Colony Optimization*, which optimizes interdependent project portfolios with *a priori* articulation of decision-maker preferences based on an outranking model. Several experimental tests show the advantages of our proposal over the two-phase approach, providing reasonable evidence of its potential for solving real-world high-scale problems with many objectives.

Keywords: portfolio selection, interdependent projects, multiobjective metaheuristic optimization, preference incorporation, multicriteria decision

1 Introduction

Portfolio problems are ubiquitous in business and government organizations. Usually, there are more good ideas for projects or programmes than there are resources (funds, capacity, time, etc.) to support them [1]. Manufacturing enterprises recognize that success depends on the selection of research and development (R&D) project portfolios, expecting that these projects will permit them to develop new products that generate growing benefits. Local governments allocate public funds to projects and programmes that improve social and educational services. Environmental regulations and alternative policy measures attempt to mitigate the harmful consequences of human activity [2]. To fight poverty, governments in underdeveloped countries fund many helpful social programmes. Portfolio consequences are usually described by multiple attributes related to the organizational strategy. A vector $z(x) = \langle z_1(x), z_2(x), \ldots, z_p(x) \rangle$ is associated with the consequences of a portfolio x considering p criteria. This is a vector representation of the portfolio's impact. In the simplest case, $z(x)$ is obtained from the cumulative sum of the benefits of the selected projects, but under interacting project conditions, it is necessary to consider the contribution of interdependent project groups. Without loss of generality, we can assume that higher criterion values are preferred to lower values. The best portfolio is obtained by solving the following problem:

$$\max_{x \in R_F} \{ \langle z_1(x), z_2(x), \ldots, z_p(x) \rangle \}, \quad (1)$$

where R_F is the space of feasible portfolios, and is usually determined by the available budget, and by constraints for the kind of projects, social roles and geographic zones. Solving Problem (1) means finding the best compromise solution according to the system of preferences and values of the Decision Maker (DM).

In the scientific literature, the problem expressed by (1) has received great interest in the management of R&D

* Corresponding author e-mail: riveragil@gmail.com

by manufacturing and industrial enterprises (e.g. [3,4,5, 6,7,8]). Most of these approaches can also be applied in the public sector. Perhaps what best characterizes the portfolio problems in non-profit organizations are the emphasis on intangible criteria and, probably, a higher number of project proposals and objectives to optimize. Many-objective problems are frequent in project portfolio optimization. For example, in socially responsible organizations, the number of criteria used for capital investment may be about a dozen (see [9]). Even more objective functions should be considered in basic research project management (cf. [10]). A high number of project proposals can apply for public support in a simple call for projects. For instance, in 2012 the US state of Georgia had a list of over 1600 applicant projects at the State Department of Transportation alone [11,12,13,14], with many potential interdependencies. There should be a large set of Pareto-efficient solutions to (1). However, the DM has to select only one portfolio according to her/his preferences for the consequences expressed by $z(x)$.

The specificity of such project portfolio problems with many objectives has been scarcely approached by the scientific literature. This paper is a contribution in this sense. It is structured as follows. Section 2 summarizes the most-widely accepted optimization model of the portfolio problem. Section 3 briefly reviews proposals for incorporating DM preferences in multi-objective optimization metaheuristics, and on this background, the method by Fernandez et al. [10,15] is detailed. Our proposal is presented in Section 4, followed by test examples and comparisons with other approaches (Section 5). Finally, some conclusions are discussed in Section 6.

2 Description and formalization of the problem

Here, we follow the proposal by Stummer and Heidemberger in [5] that was also addressed by Doerner et al. [16,17] and Carazo et al. [18,19].

Let X be the set of applicant projects competing for resources. A portfolio (a subset of X) is typically represented by a binary vector $x = \{x_1, x_2, \ldots, x_N\}$, where N is the total of project proposals; the variables x_j indicate whether the project j is included in the portfolio ($x_j = 1$) or not ($x_j = 0$).

Let us denote by $f(j) = \{f_1(j), f_2(j), \ldots, f_p(j)\}$ the benefits provided by the jth project. The benefits provided by portfolio x are expressed by Equation (2):

$$z(x) = \{z_1(x), z_2(x), \ldots, z_p(x)\}, \tag{2}$$

where $z_k(x)$ is defined as

$$z_k(x) = \sum_{j=1}^{N} x_j \cdot f_k(j) + \sum_{i=1}^{S} g_i(x) \cdot a_{i,k}. \tag{3}$$

In Equation (3), the first term is the cumulative sum of the benefits from the selected projects to the kth objective function. The second term is the sum of the synergetic interactions among the projects in the portfolio. S is the number of interactions that impact the objectives. Let us assume that those interactions have been identified by the DM. Function $g_i(x)$ indicates if the ith interaction occurs in the portfolio x. If $A_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,N}\}$ is a binary vector that indicates which projects are affected by the ith interdependency ($A_{i,j} = 1$ represents that the jth project is considered in the ith objective interaction), $g_i(x)$ may be defined as

$$g_i(x) = \begin{cases} 1 & \text{if } m_i \leq \sum_{j=1}^{N} (x_j \cdot A_{i,j}) \leq M_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

In Equation (4), m_i and M_i are respectively the minimum and maximum number of projects required for synergy i to occur, thus gaining additional benefits.

In Equation (3), $a_{i,k}$ is the value added to the kth objective when the ith synergy is activated. The interaction has been particularly named *cannibalization* if $a_{i,k}$ is negative.

Suppose that there are q categories of resources destined for supporting project proposals. Let $\{\mathbb{B}_1, \mathbb{B}_2, \ldots, \mathbb{B}_q\}$ be the set containing the quantity of available resources for each category (e.g. financial, human or technological resources), and let $c_{j,k}$ be the amount of the kth resource requested by project j. Thus, the total of the kth resource needed for implementing portfolio x, is expressed by Equation (5):

$$c_k(x) = \sum_{j=1}^{N} x_j \cdot c_{j,k} + \sum_{i=1}^{R} h_i(x) \cdot b_{i,k}. \tag{5}$$

The first term in Equation (5) is the sum of resources consumed by the projects in x, without considering resource interactions. The second term is the sum concerning interactions that affect costs and resources requested. R is the number of these interdependencies, $h_i(x)$ is a binary function that indicates if the ith resource interaction occurs, and $b_{i,k}$ is the change in the kth cost produced by the ith interaction. $h_i(x)$ is defined in Equation (6) similarly to $g_i(x)$, but considering n_i and N_i as limits for activating synergy. Equation (6) presents the definition of $h_i(x)$:

$$h_i(x) = \begin{cases} 1 & \text{if } n_i \leq \sum_{j=1}^{N} (x_j \cdot C_{i,j}) \leq N_i, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $C_i = \{C_{i,1}, C_{i,2}, \ldots, C_{i,N}\}$ is a binary vector that indicates which projects are affected by the ith cost interdependency.

Of course, Problem (1) is subject to the budgetary constraint:

$$c_k(x) \leq \mathbb{B}_k \quad \forall k \in \{1, 2, \ldots, q\}. \tag{7}$$

Besides Equation (7), other strategic and logical constraints could be regarded. For example:

– *Constraints to ensure equitable conditions for all competent areas of the organization.* All applicant projects are grouped according to pre-established criteria. The organization determines limits in terms of number of supported projects (or quantities of allocated budget) for each group.
– *Constraints to prevent the presence of mutually-excluding projects.* Some projects (primarily because of their nature and organizational rules) cannot simultaneously receive support in the same portfolio decision process. These projects often receive the adjective 'redundant'.

We are not taking into account project scheduling, thus we are tackling the stationary version of the problem presented in [16,19]; for this reason, all the concerns related to schedule are not included in either Equations (2–7) or the above-mentioned constraints. Conditions of partial support have no special processing, but it is possible to include dummy projects that represent different versions of the same project. So, dummy projects are treated like redundant proposals, in the same sense as it is suggested in [5,16,17,18,19].

3 An outline of the state of the art

3.1 A brief outline and some criticisms of previous approaches

Only non-dominated solutions to (1) can fulfil the conditions necessary for being considered the best portfolio. So most solution methods seek to generate the Pareto frontier, and later, by some interactive method, multicriteria procedure or heuristic, try to identify the best compromise. These approaches assume that the DM has the capacity to make valid judgments about the set of efficient points until the best compromise is reached. This way to identify the best solution is commonly referred to as *a posteriori* preferences modelling [20].

In [21], Ghasemzadeh et al. model preferences using a weighted-sum function. They approximate the Pareto frontier by changing the weights and solving the resultant model by 0-1 programming. Stummer and Heidenberger in [5] include synergy and redundancy in selecting R&D projects; their procedure consists of three phases: 1) filtering the proposals and retaining the most promising projects in order to reduce the set of projects to a 'manageable' size, 2) generating the efficient frontier of portfolios for the reduced set by an integer linear programming method, and 3) supporting the decision-making process, helping the DM to identify the best compromise by an interactive process.

However, most recent works show the advantages of multi-objective metaheuristic methods to approximate the Pareto set (e.g. [8,19,22,23,24,25,26,27,28]). Doerner et al. in [17] combine Ant Colony Optimization (ACO) with 0-1 dynamic mathematical programming to initialize the algorithm with enhanced solutions. One of the most complete proposals was suggested by Carazo et al. [18, 19]; they model interactions among projects (in the same way as Stummer and Heidenberger in [5]) and temporal dependencies, enabling the allocation of resources not used in previous periods. By means of a Scatter Search, Carazo et al. [18] outperform SPEA2 [29] in the range of 25–60 projects considering up to six objective functions.

Compared with multi-objective optimization methods based on mathematical programming, metaheuristic approaches exhibit relevant advantages:

– they have the ability to deal with a set of solutions (called a population) at the same time, allowing for the efficient frontier to be approximated in a single algorithm run, and
– they are less sensitive to the mathematical properties of objective functions and problem constraints.

However, many researchers have argued that, when the number of objective functions increases, the selection of appropriate individuals for conducting the population towards the Pareto frontier becomes more difficult (e.g. [30,31,32,33]). According to [32], other important concerns are the so-called *Dominance Resistant Solutions* (e.g. [34]). They are not Pareto solutions, but they have near-optimal values in some objectives though with a poor value in at least one of the remaining objectives. These solutions can be hardly dominated in a population. Their number grows as the dimension of the objective space is increased.

In the presence of many objectives, there are other important concerns associated with the *a posteriori* articulation of preferences:

1. The visualization of the Pareto front in high-dimensional objective spaces is very cumbersome.
2. The number of Pareto optimal points grows exponentially, making it hard to obtain a representative sample of the non-dominated frontier.
3. According to the famous Miller's paper [35], the human mind is limited to handling a small number of information pieces simultaneously, thus being questionable the issue of identifying the best compromise solution when the DM should compare even a small subset of non-dominated solutions in problems with many objectives.

Most approaches from the field of Multi-Criteria Decision Analysis (MCDA) do not perform well on large decision problems. Incomparability, non-transitivity, cyclic preferences and dependence with respect to 'irrelevant alternatives' make it difficult to reach a reliable final prescription.

In order to make the decision making phase easier, the DM would agree incorporate his/her multicriteria

preferences into the search process. This preference information is used to guide the search towards the *Region of Interest* (RoI) [36], the privileged zone of the Pareto frontier that best matches the DM's preferences.

The DM's preference information can be expressed in different ways. According to Bechikh [37], the most commonly-used ways are the following:

1. Those in which importance factors (weights) are assigned by the DM to each objective function (e.g. [38,39,40]).
2. Those in which the DM makes pair-wise comparisons on a subset of the current population, in order to rank the sample's solutions (e.g. [41,42,43,44,45,46]).
3. Those in which pair-wise comparisons between pairs of objective functions are performed in order to rank the set of objective functions (e.g. [47,48,49]).
4. Those based on goals or aspiration levels to be achieved by each objective (reference point) (e.g. [36, 50,51,52,53,54]).
5. Those in which the DM identifies acceptable trade-offs between objective functions (e.g. [55]).
6. Those in which the DM supplies the model's parameters to build a fuzzy outranking relation (e.g. [15,56]).
7. The construction of a desirability function which is based on the assignment of some desirability thresholds (e.g. [57]).

In the field of project portfolio optimization, the model proposed in [10] has shown substantial benefits for tackling these problems. This model is briefly explained below.

3.2 The best portfolio in the sense of Fernandez et al. [10]

The proposal by Fernandez et al. [10,15] is based on the relational system of preferences described in [58] by Roy. A crucial model is the degree of credibility of the statement 'x is at least as good as y'. This is represented as $\sigma(x,y)$ and could be calculated using proven methods from the literature, such as ELECTRE [59] and PROMETHEE [60]. Considering the parameters λ, β, and ε ($0 \leq \varepsilon \leq \beta \leq \lambda$ and $\lambda > 0.5$), the proposal in [10, 15] identifies one of the following relations for each pair of portfolios (x,y):

1. *Strict preference*: Denoted as xPy, represents the situation when the DM significantly prefers x. It is defined as a disjunction of the conditions:
   (a) x dominates y.
   (b) $\sigma(x,y) \geq \lambda \wedge \sigma(y,x) < 0.5$.
   (c) $\sigma(x,y) \geq \lambda \wedge [0.5 \leq \sigma(y,x) < \lambda] \wedge [\sigma(x,y) - \sigma(y,x)] \geq \beta$
2. *Indifference*: From the DM's perspective, the two alternatives have a high degree of equivalence, so he/she cannot state that one is preferred over the other. This relationship is denoted as xIy. In terms of $\sigma(x,y)$ this is defined as the conjunction of:
   (a) $\sigma(x,y) \geq \lambda \wedge \sigma(y,x) \geq \lambda$.
   (b) $|\sigma(x,y) - \sigma(y,x)| \leq \varepsilon$.
3. *Weak preference*: Represented as xQy, this models a state of doubt between xPy and xIy. It can be defined as the conjunction of:
   (a) $\sigma(x,y) \leq \lambda \wedge \sigma(x,y) \geq \sigma(y,x)$.
   (b) $\neg x$P$y \wedge \neg x$Iy.
4. *Incomparability*: From the point of view of the DM, there is high heterogeneity between the alternatives, so he/she cannot set a preference relation between them. This is denoted as xRy, and is expressed in terms of $\sigma(x,y)$ as xR$y \Rightarrow \sigma(x,y) < 0.5 \wedge \sigma(y,x) < 0.5$.
5. *k-Preference*: This represents a state of doubt between xPy and xRy, and is denoted as xKy. $(x,y) \in$ K if the following three conditions are true:
   (a) $0.5 \leq \sigma(x,y) \leq \lambda$.
   (b) $\sigma(y,x) < 0.5$.
   (c) $\sigma(x,y) - \sigma(y,x) > \frac{\beta}{2}$.

Indifference corresponds to the existence of clear and positive reasons that justify equivalence between the two options. Additionally, incomparability represents situations where the DM cannot, or does not want to, express a preference. Strict preference is associated with conditions in which the DM has clear and well-defined reasons justifying the choice of one alternative over the other. However, because the DM usually shows non-ideal behaviour, the weak preference and the k-preference also exist. These relations can be considered as 'weakened' ways of the strict preference.

From a set of feasible portfolios O, the preferential system defines the following sets:

1. $S(O,x) = \{y \in O \mid y\text{P}x\}$ is composed of the solutions that strictly outrank x.
2. $NS(O) = \{x \in O \mid S(O,x) = \emptyset\}$ is known as the *non-strictly-outranked frontier*.
3. $W(O,x) = \{y \in NS(O) \mid y\text{Q}x \wedge y\text{K}x\}$ is composed of the non-strictly-outranked solutions that weakly outrank x.
4. $NW(O) = \{x \in O \mid W(O,x) = \emptyset\}$ is known as the *non-weakly-outranked frontier*.

Besides the weak outranking, the net flow score is another measure used in [10,15] to identify the DM's preferences in the non-strictly-outranked frontier. It can be defined as:

$$F_n(x) = \sum_{y \in NS(O) \setminus \{x\}} [\sigma(x,y) - \sigma(y,x)]. \qquad (8)$$

Since $F_n(x) > F_n(y)$ indicates a preference for x over y, Fernandez et al. [15] define:

1. $F(O,x) = \{y \in NS(O) \mid F_n(y) > F_n(x)\}$ to be the set of non-strictly-outranked solutions that are greater in net flow to x.
2. $NF(O) = \{x \in NS(O) \mid F(O,x) = \emptyset\}$ to be the *net-flow non-outranked frontier*.

Fernandez et al. [10] proved that the best portfolio compatible with the fuzzy outranking relation σ should be a non-strictly outranked solution that is simultaneously a non-dominated solution to the problem:

$$\min_{x \in O} \{\langle |S(O,x)|, |W(O,x)|, |F(O,x)| \rangle\}. \quad (9)$$

As a consequence of the last remark, the best portfolio can be found through a lexicographic search, with pre-emptive priority favouring $|S(O,x)|$.

The above three-objective problem is a map of the original problem in (1). When the DM is confident on the preference model, he/she should accept that the best compromise is a non-dominated solution of Problem (9). It is also interesting that the equivalence between the problem in (1) and its mapped three-objective problem is valid independently of the original objective space dimension. This may be very important in solving portfolio problems with many objective functions [15].

The model parameters need to be adjusted according to the specific characteristics of the problem and of the DM. This can be done by an interaction between the DM and a Decision Analyst (DA), utilizing, if necessary, indirect elicitation methods to support this task [61,62,63]. The DM should assess the parameters included in:

– the calculation of σ (e.g. criterion weights and thresholds), and
– the system of preferences (λ, β and ε).

This is not an easy task since DMs usually have difficulties in specifying outranking parameters and require an intense support by a DA. To facilitate this process, the pair DM-DA can use the Preference Disaggregation Analysis (PDA) paradigm (e.g. [61]), which has received increasing interest from the MCDA community. PDA infers the model's parameters from holistic judgments provided by the DM. Those judgments may be obtained from decisions made for a limited set of fictitious portfolios, or decisions taken for a subset of the portfolios under consideration for which the DM can easily make a judgment. In the framework of outranking methods, PDA has been recently approached in [62,63].

Fernandez et al. in [10] solved problems of allocating public funds via their outranking model. However, that work does not consider interactions among projects, which is an important concern in most practical applications.

In light of this feedback, we propose here a portfolio optimization metaheuristic approach based on the preferential model proposed in [15]. So, our metaheuristic inherits all the advantages of this model, but we have incorporated the capacity to solve portfolios with interdependent projects. Several papers in the literature consider synergy as an inherent characteristic of the portfolio problem (e.g. [5,16,17,18,19]). Our solution approach, called *Non-Outranked Ant Colony Optimization* shows promising results compared to other related algorithms. Experimental results provide evidence that it is very capable of getting close to the Pareto frontier when the best compromise is sought.

4 Our proposal

Our algorithm, NO-ACO (*Non-Outranked Ant Colony Optimization*), is based on the optimization idea proposed in [64] by Dorigo and Gambardella, which has been adapted more than once to find a set of Pareto solutions (e.g. [16,65,66,67]), but incorporates the preference model from [15]. The algorithm performs the optimization process through a set of agents called ants. Each ant in the colony builds a portfolio by selecting a project at a time. The way of choosing each project is called a selection rule. When all ants have finished constructing their portfolios, these are evaluated and each ant drops pheromone according to this assessment. Pheromone is used for learning, allowing the next generation of ants to acquire knowledge about the structure of the best solutions. To prevent premature convergence, the colony includes a strategic oblivion mechanism, known as evaporation, which reduces the pheromone trail over specified periods of time. In order to improve the intensification, NO-ACO includes a variable neighbourhood search for the best solutions. This local search runs once per iteration. This intensifier scheme is complemented by a diversifier mechanism, in which portfolios that have remained non-strictly-outranked for more than γ generations are removed from the solution set. This allows the selective pressure to be relaxed. This behaviour is desirable when the algorithm has only found out local optima. The optimization process ends when a predetermined termination criterion (such as a maximum number of iterations, or a subsequent recurrence of the best solution) is reached. The following sections describe the elements of the NO-ACO algorithm in further detail.

4.1 Pheromone representation

Pheromone is usually represented by the Greek letter τ and is modelled in NO-ACO as a two dimensional array of size $N \times N$, where N is the total number of applicant project proposals. The pheromone between two projects i and j is represented as $\tau_{i,j}$, and indicates how good it is that both projects receive financial support. Pheromone values are in range $(0,1]$, initializing at the upper limit to prevent premature convergence. The pheromone matrix acts as a reinforcement learning structure reflecting the knowledge gained by the ants that formed high-quality portfolios.

The pheromone representation of NO-ACO allows identifying pairs, trios, quartets or larger project subgroups present in the best portfolios. Most likely, some synergies (mainly those that decrease costs and/or increase objectives) occur in the best portfolios. These

favourable project interactions are detected through the pheromone matrix and this knowledge is transmitted to ants of the next generation for building better solutions.

4.2 Selection rule

Each ant builds its portfolio by selecting the projects one by one, taking into account two factors:

–*Local knowledge* (*heuristic*): This considers the benefits provided by the project to the portfolio and how many resources the project consumes. Local knowledge for the jth project is denoted by η_j and is calculated by the expression:

$$\eta_j = \frac{\frac{1}{c(j)}\sum_{k=1}^{p} f_k(j)}{\max_{l\in X}\left\{\frac{1}{c(l)}\sum_{k=1}^{p} f_k(l)\right\}}, \tag{10}$$

where $c(j)$ is a measure proportional to the cost of project j, p is the number of objectives, X is the applicant project list, and $f_k(j)$ is the benefit from project j to the kth objective. Equation (10) promotes the inclusion of projects that have a good balance between intended objectives and requested budget. In Equation (10), $c(j)$ is defined as

$$c(j) = \frac{1}{q}\sum_{k=1}^{q}\left(\frac{c_{j,k}}{\mathbb{B}_k}\right), \tag{11}$$

where, q is the number of categories of resources, $c_{j,k}$ is the kth resource cost requested by project j, and $\mathbb{B}_k$ is the available amount of resource in the kth category.

–*Global knowledge* (*learning*): This takes into account the experience of previous generations of ants, expressed in the pheromone matrix. The global knowledge for project i to be included in a portfolio x is denoted by $\overline{\tau(x,i)}$ and is defined by the expression:

$$\overline{\tau(x,i)} = \frac{\sum_{j=1}^{N}(x_j)\tau_{i,j}}{\sum_{j=1}^{N} x_j}, \tag{12}$$

where N is the total number of applicant projects, x_j is the binary value indicating whether the jth project is included in the portfolio x, and $\tau_{i,j}$ is the pheromone for projects i and j. The numerator in Equation (12) is the total sum of pheromone between i and each project in portfolio x; the denominator is the cardinality of x. The global knowledge favours the selection of projects that were part of the best portfolios in previous generations. At the first iteration this knowledge has no effect on portfolio formation process.

Both knowledge factors are linearly combined into a single evaluation function, which corresponds to Equation (13):

$$\Omega(x,i) = w\cdot\eta_i + (1-w)\cdot\overline{\tau(x,i)}, \tag{13}$$

where w is a parameter weighing global and local knowledge. Each ant in the colony has a different value for w, which is generated at random in the range $[0,W]$ with $W<1$. W determines the possible greatest value of w for each ant. The function Ω forms the basis of the selection rule.

If x is a *partially-constructed portfolio*, one or more projects may be included in x. From among all the project proposals, only those that are not part of x and the inclusion of which favours the fulfilment of budgetary constraints should be considered. This set is known as the *candidate project list* and is denoted by $X^{\ominus}$. Note that $X^{\ominus}$ is a subset of X. The choice of which $j\in X^{\ominus}$ will be added is made by using the selection rule:

$$j = \begin{cases} \arg\max_{i\in X^{\ominus}}\{\Omega(x,i)\} & \text{if } \wp\le\alpha_1, \\ \mathscr{L}_{i\in X^{\ominus}}\{\Omega(x,i)\} & \text{if } \alpha_1<\wp\le\alpha_2, \\ \ell_{i\in X^{\ominus}} & \text{otherwise,} \end{cases} \tag{14}$$

where j is the next project to be included; $\wp$ is a pseudorandom number between zero and one; α_1 is a parameter that sets the intensification probability in the algorithm (choosing the project with the greatest value of Ω); and $\alpha_2-\alpha_1$ is the probability of triggering a middle state between intensification and diversification (randomly selecting a project i with probability proportional to its assessment Ω), this selection scheme is represented by $\mathscr{L}$; in the event that $\wp>\alpha_2$, diversification is promoted by means of the function ℓ (taking a project uniformly at random).

4.3 Pheromone laying and evaporation

At the beginning of the first iteration, the pheromone matrix is initialized to $\tau_{i,j}=1$ for all $(i,j)\in N\times N$. After that, each ant constructs a feasible portfolio. In a colony with n ants, n new solutions are generated after each iteration, and there is also a set of size m with the best portfolios found from the previous iterations. If all alternatives are integrated into a set O whose cardinality is $n+m$, we can identify the non-strictly-outranked front $NS(O)$.

In addition, $NS(O)$ is subdivided into domination fronts. The fronts are obtained by considering the minimization of two objectives, $W(O,x)$ and $F(O,x)$, according to the best-compromise definition given in Problem (9). The set composed by these fronts is denoted by $\mathscr{F}=\{\mathscr{F}_1,\mathscr{F}_2,\ldots,\mathscr{F}_k,\mathscr{F}_{k+1},\ldots\}$, where $\mathscr{F}_1$ contains the non-dominated solutions, $\mathscr{F}_2$ contains the portfolios that are dominated by only one solution, $\mathscr{F}_3$ those dominated by two solutions, and so forth. In general, the portfolios dominated by k solutions are in $\mathscr{F}_{k+1}$. The set

$\mathscr{F}$ will be used in the pheromone intensification in order to increase the selective pressure towards the best compromise.

Each pair of projects (i, j) for each solution $x \in O$ intensifies the pheromone trail according to the expression:

$$\tau_{i,j} = \begin{cases} \tau_{i,j} + \Delta\tau_{i,j} & \text{if } x \in NS(O), \\ \tau_{i,j} & \text{otherwise.} \end{cases} \quad (15)$$

If x is a non-strictly-outranked solution, then there is a k such that $x \in \mathscr{F}_k$. The pheromone increase depends on k, and is defined as:

$$\Delta\tau_{i,j} = \left(\frac{|\mathscr{F}| - k + 1}{|\mathscr{F}|}\right)(1 - \tau_{i,j}) \quad \text{if } x \in \mathscr{F}_k, \quad (16)$$

where i and j belong to portfolio x.

At the end of each iteration, the entire pheromone matrix is evaporated by multiplication by a constant factor lying between zero and one, denoted as ρ.

Algorithm 1: NO-ACO's local search algorithm

```
   Data: NS(O) (non-strictly-outranked frontier), X
         (applicant project list)
   Result: A better approximation of NS(O)
1  Initialize: N ← |X|, v ← ⌈ln N⌉, O' ← ∅
2  P ← select_projects(v,X)
3  C ← generate_combinations(P)
4  foreach c ∈ C do
5      foreach o ∈ NS(O) do
6          o' ← o
7          foreach p ∈ P do
8              if p ∈ c then
9                  Add project p to portfolio o'
10             else
11                 Remove project p from portfolio o'
12         repair(o')
13         if o' ∈ R_F then
14             O' ← O' ∪ {o'}
15 O ← O ∪ O'
16 Recalculate NS(O)
17 return NS(O)
```

4.4 Local search

The algorithm intensification is promoted by a greedy variable-neighbourhood local search that is only carried out on non-strictly-outranked solutions. This search explores regions near to the best known solutions by a simple scheme consisting of randomly selecting v projects, and generating all possible combinations of them for each solution in the non-strictly-outranked frontier. Small values for v provoke behaviour that is too greedy, whereas large values produce intolerable computation times. In our experiments we obtained a good balance between these by using $v = \lceil \ln N \rceil$. The algorithmic outline for the local search is illustrated by Algorithm 1.

As observed in Algorithm 1, the search starts by choosing v projects at random (Line 2), and generating all combinations of them (Line 3). Every combination is set for each portfolio in $NS(O)$ (Lines 4–11).

In Line 12, procedure `repair` has two main goals: 1) improving clearly-suboptimal portfolios, and 2) bringing unfeasible portfolios to the feasible region. Thus, it has two conditions to check:

– *If the generated solution is partially constructed*: then `repair` adds projects to portfolio, according to selection rule but respecting the bits assigned by the current combination (represented by c in Algorithm 1). This is done until no project can be added to the portfolio.
– *If the generated solution is unfeasible*: then `repair` removes projects at random until the portfolio does not surpass the budget. The probability of removing a project is inversely proportional to its expected benefits. No project chosen by the current combination can be removed. In the generated instances, `repair` procedure could make feasible the most of solutions.

Each feasible solution is evaluated to verify whether or not it is a non-strictly-outranked solution (Lines 13–17).

4.5 Algorithmic description of NO-ACO

Algorithm 2 presents an algorithmic outline of NO-ACO. Line 1 indicates the initialization of the control variables, and Lines 2–27 show the search process.

Lines 4–12 of Algorithm 2 illustrate the process of the formation of portfolios. Each ant starts from an empty portfolio, and projects are added by the selection rule, one at a time. Complete and feasible solutions are stored in O. These are then evaluated according to Problem (9), and the non-strictly-outranked solutions are refined by local search. Pheromone increase is the next step (Lines 14–17).

In Lines 18–23, the non-strictly-outranked set and some algorithm control variables are updated. Subsequently, at Line 24, the procedure `remove_and_refill` counts the number of iterations of each solution in the local NS frontier. All solutions with more than γ iterations are removed from the local set, and replaced by new solutions in the global NS frontier. These new solutions should not have belonged to NS_{local}, therefore they have to be generated by the local search on NS^*_{global}. While this search is providing non-strictly-outranked portfolios the replacement will be possible. The removed solutions can still belong to the global non-strictly-outranked front, but no longer influence the optimization process made by the colony.

Algorithm 2: Non-Outranked Ant Colony Optimization algorithm

Data: X (applicant project list), $\mathbb{B}$ (budget)
Result: An approximation of $NS(O)$

1 Initialize: $iter \leftarrow 1$, $rep \leftarrow 0$, $NS_{local} \leftarrow \emptyset$, $NS_{global} \leftarrow \emptyset$, $NS^*_{local} \leftarrow \emptyset$
2 **repeat**
3 $O \leftarrow \emptyset$
4 **foreach** *ant in the colony* **do**
5 $x \leftarrow$ `make_empty_portfolio()`
6 $X^{\ominus} \leftarrow$ `get_candidate_projects(`X,x`)` `// Section 4.2`
7 **repeat**
8 $j \leftarrow$ `selection_Rule(`$X^{\ominus},x$`)` `// Equation (14)`
9 $x_j \leftarrow 1$
10 $X^{\ominus} \leftarrow$ `get_candidate_projects(`X,x`)` `// Section 4.2`
11 **until** $X^{\ominus} = \emptyset$
12 $O \leftarrow O \cup \{x\}$
13 $O \leftarrow O \cup NS_{local}$
14 $NS_{local} \leftarrow \arg\min_{x \in O} \{\langle |S(O,x)|, |W(O,x)|, |F(O,x)| \rangle\}$ `// Problem (9)`
15 $NS_{local} \leftarrow$ `local_search(`NS_{local},X`)` `// Algorithm 1`
16 **foreach** $x \in NS_{local}$ **do**
17 `lay_pheromone(`x,O`)` `// Equations (15-16)`
18 $NS^*_{global} \leftarrow NS_{global} \cup NS_{local}$
19 $NS^*_{global} \leftarrow$ `local_search(`NS^*_{global},X`)` `// Algorithm 1`
20 **if** $NS_{global} = NS^*_{global}$ **then**
21 $rep \leftarrow rep + 1$
22 **else**
23 $rep \leftarrow 0$
24 `remove_and_refill(`$NS_{local},NS^*_{global},\gamma$`)`
25 Evaporate pheromone `// Section 4.3`
26 Update: $iter \leftarrow iter + 1, NS_{global} \leftarrow NS^*_{global}$
27 **until** $rep = rep_{\max} \vee iter = iter_{\max}$
28 **return** NS_{global}

At the end of each iteration, pheromone is evaporated (Line 25), and the remaining algorithm control variables are updated (Line 26). The algorithm finishes when it has iterated with the same set of solutions as the non-strictly-outranked frontier during $rep_{\max}$ iterations, or if it has reached the maximum number of iterations $iter_{\max}$ (Line 27).

5 Case study: Optimization of social assistance portfolios

Consider a DM facing a portfolio problem, with 100 project proposals are aimed at benefitting the most precarious social classes. The project quality is measured as the number of beneficiaries for each of nine criteria that have previously been established. Each objective is associated with one of three classes (extreme poverty, lower class and lower-middle class) and one of three levels of impact (low, medium and high).

The total budget to distribute is 250 million dollars. The proposals can be grouped into three types according to their nature, and into two geographic regions according to the location of their impact. Furthermore, in a desire to provide equitable conditions, the DM imposes the following restrictions:

1. The budget allocated to support each project type should be between 20% and 60% of the total budget.
2. The financial support allocated to each region must be at least 30% of the total budget, and no more than 70%.

The DM has also identified 20 relevant interactions among projects: four of them are cannibalization phenomena, six correspond to situations of mutually-excluding projects, and ten are synergism interactions. There are up to five projects per interaction.

In order to make easier the comparative descriptions, in this section the term *Pareto efficiency* (and all the related terms, such as *optimal* or *efficient portfolio*) will be used to refer to non-dominated solutions of (1), and the term *best compromise* to best solutions to (9) (the best portfolio compatible with the fuzzy outranking relation [10,15]).

Below, we present a range of experiments to verify the validity and advantages of our approach to solving this case study. They give evidence of the benefits of incorporating the DM's preferences during the

Table 1: Effect of preferences incorporation on the Pareto Ant Colony Optimization algorithm

Instance	*Algorithm*	*Time (seconds)*	*Size of the solution set*	*Non-dominated solutions in* $O_1 \cup O_2$	*Solutions belonging to* $NS(O_1 \cup O_2)$	*Obtains the best compromise in* $O_1 \cup O_2$
1	P-ACO	3448.07	2006	928	10	
	P-ACO-P	536.66	15	15	10	✓
2	P-ACO	3470.29	2514	1295	7	
	P-ACO-P	775.94	19	19	13	✓
3	P-ACO	3485.16	2456	280	13	
	P-ACO-P	1112.49	34	34	17	✓
4	P-ACO	3591.27	2587	1392	10	✓
	P-ACO-P	734.58	38	37	19	✓
5	P-ACO	3525.85	2245	1165	10	
	P-ACO-P	1035.85	21	21	15	✓
6	P-ACO	3496.68	2013	161	11	
	P-ACO-P	855.68	18	18	10	✓
7	P-ACO	3549.55	2211	766	13	✓
	P-ACO-P	161.02	19	19	14	✓
8	P-ACO	3464.27	2285	1317	13	
	P-ACO-P	1646.32	28	28	21	✓
9	P-ACO	3707.65	965	762	4	✓
	P-ACO-P	712.24	25	25	11	✓
10	P-ACO	3549.67	2255	1403	15	✓
	P-ACO-P	651.43	18	18	16	✓

Note: O_1 and O_2 are the solution sets generated by P-ACO and P-ACO-P respectively.
The best compromise is the best solution to Problem (9) on $O_1 \cup O_2$.

optimization process, and thus they also prove that our approach has good potential for solving real resource-allocation problems.

5.1 Effect of incorporating the DM's preferences

To the best of our knowledge, the P-ACO algorithm [16] is the most relevant ant colony algorithm applied to solve project portfolio selection. In order to appraise the effect of incorporating the DM's preferences on a multi-objective optimization algorithm, we implemented a version of P-ACO that included the preferential model described in Section 3.2. This adaptation was called P-ACO with preferences (P-ACO-P). Instead of approximating the Pareto frontier defined by the nine maximizing objectives of the problem, it searches for the best compromise expressed by Problem (9). In order to reflect a credible decision situation, we assign the values suggested by Fernandez et al. in [15] to the preferential model parameters. There is no other difference between P-ACO and P-ACO-P. Both algorithms were programmed in Java language, using the JDK 1.6 compiler, and NetBeans 6.9.1 as Integrated Development Environment (IDE). The experiments were run on a Mac Pro with an Intel Quad-Core 2.8 GHz processor and 3 GB of RAM.

The P-ACO parameter setting was that suggested in [16] by Doerner et al. The version that incorporates preferences has the same setting values.

Table 1 shows the experimental results on ten artificial instances following the case-study features.

The best compromise has been identified from solutions sets generated by both optimization methods. In this sense, that best compromise is related to the known solution set; therefore, it will be called the *known best compromise*, which approximates the *true best compromise*. This is a non-dominated solution of Problem (9) on the original objective space. So, the true best compromise must belong to the true efficient set, and it should not be strictly outranked by any other Pareto solution.

As can be seen from Table 1, incorporating preferences provides a closer approximation to a privileged region of the Pareto frontier. The version considering preferences provides solutions that dominated the 54%, on average, of solutions produced by the original version of the algorithm. Probably, with many objectives, P-ACO is sensitive to the existence of dominant resistant solutions. There is also a significant run-time reduction (in the test cases, this reduction was 76% on average). Also, if the model of preferences matches with the DM's preferences, the best compromise among the set of all portfolios generated is always identified by P-ACO-P. Furthermore, when the DM has to

Table 2: Efficiency analysis of NO-ACO

Instance	*Algorithm*	*Time (seconds)*	*Size of the solution set*	*Non-dominated solutions in* $O_1 \cup O_2$	*Solutions belonging to* $NS(O_1 \cup O_2)$	*Obtains the best compromise in* $O_1 \cup O_2$
1	SS-PPS	37946.70	4997	4981	12	
	NO-ACO	5101.78	16	16	15	✓
2	SS-PPS	23223.68	4996	4956	10	
	NO-ACO	2130.98	18	18	18	✓
3	SS-PPS	33265.31	4996	4970	21	
	NO-ACO	3091.89	29	29	28	✓
4	SS-PPS	49865.11	4997	4946	24	
	NO-ACO	4720.02	43	43	40	✓
5	SS-PPS	30218.23	4996	4959	12	
	NO-ACO	4009.47	32	32	32	✓
6	SS-PPS	43253.64	4996	4949	18	✓
	NO-ACO	2743.55	26	26	22	✓
7	SS-PPS	29386.18	4973	4973	14	
	NO-ACO	4512.12	21	21	21	✓
8	SS-PPS	38585.35	4996	4940	27	
	NO-ACO	3901.76	35	35	35	✓
9	SS-PPS	35514.66	4996	4936	9	
	NO-ACO	1238.33	16	16	12	✓
10	SS-PPS	46241.69	4996	4956	16	
	NO-ACO	1467.29	20	20	20	✓

Note: O_1 and O_2 are the solution sets generated by SS-PPS and NO-ACO respectively. The best compromise is the best solution to Problem (9) on $O_1 \cup O_2$.

choose one alternative as the final decision, the thousands of portfolios from P-ACO make it difficult to reach a decision. By incorporating preferences, this drawback is very strongly reduced.

5.2 Evaluation of NO-ACO solutions

For the problem presented in this section, the only way to ensure that a solution is the true best compromise is if we know the whole true Pareto frontier, or at least, the full non-strictly-outranked frontier. For instances of large size like those we have addressed, it is not possible to know with certainty the Pareto frontier. However, there are methods reported in the literature that can approximate this frontier with an acceptable error.

In order to verify whether the NO-ACO solutions acceptably approximate the true Pareto frontier, we have estimated the Pareto set by means of SS-PPS, as proposed by Carazo et al. [18,19]. This is one of the most recent algorithms for portfolio optimization, and experimental tests prove its high performance, outperforming SPEA2. SS-PPS solved the case-study instances by finding a representative sample of up to five thousand efficient points according to the parameter setting suggested in [18,19].

NO-ACO was programmed in Java language, using the JDK 1.6 compiler, and NetBeans 6.9.1 as IDE. The experiments were run on a Mac Pro with an Intel Quad-Core 2.8 GHz processor and 3 GB of RAM.

Again we used the values suggested in [15] for the preferential model parameters. Besides, the NO-ACO parameter setting used to obtain the results in this section is: $\alpha_1 = 0.65$, $\alpha_2 = 0.85$, $\rho = 0.9$, $\gamma = 25$, $W = 0.60$, $rep_{\max} = 50$ and $iter_{\max} = 100000$. Moreover, the colony has one hundred ants. This setting was obtained from exploring parameter values with the objective of achieving a good algorithmic performance. Taking into account the results in a wide range of instances, we consider that these parameter values are robust enough to maintain an efficient behavior of NO-ACO.

We want to give evidence that our approach acceptably approximates the best compromise. With this aim, we solved the same ten instances from Section 5.1. For these, we have approximated: 1) the best compromise by using NO-ACO, and 2) the Pareto frontier by means of SS-PPS.

The results are summarized in Table 2. On analysing the data, we may conclude that our algorithm has efficient behaviour. NO-ACO got close to the Pareto frontier better than SS-PPS in the most preferred region (the so-called RoI), that is, the non-strictly-outranked frontier. No NO-ACO solution is dominated by an SS-PPS one, and our approach could dominate 16–60 solutions suggested by the other method. Additionally, our proposal was able to identify the best compromise from the entire

Table 3: A sample of the non-strictly-outranked frontier generated by NO-ACO compared to the ranking-based solution

	Portfolio	Values of objective functions																Number of solutions that outranks it		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	strictly	weakly	in net flow score
by NO-ACO	1	106	806	504	612	107	811	502	605	983	871	473	610	108	847	499	597	0	0	0
	2	96	766	467	556	98	786	459	562	988	772	457	565	98	756	454	545	0	0	1
	3	98	730	461	562	99	740	475	564	988	796	464	563	95	767	453	541	0	1	2
	4	100	742	479	545	94	744	459	565	992	785	451	547	96	745	447	535	0	2	1
	5	96	742	462	553	95	751	456	562	999	809	454	562	94	776	452	546	0	2	1
	6	98	743	462	550	95	730	473	559	991	765	460	553	95	740	450	541	0	2	5
	7	98	746	466	556	98	769	454	569	990	790	447	565	94	770	454	547	0	3	3
	8	92	739	469	557	91	753	445	556	990	784	468	565	90	738	440	549	0	4	6
	9	98	733	461	556	95	750	448	567	987	791	454	565	97	777	454	556	0	8	7
ranking-based		96	736	471	558	95	762	453	561	944	768	469	565	97	756	436	540	9	0	9

approximated frontier, using only, on average, 10% of the time required to estimate the whole Pareto set.

There is evidence of the advantages of incorporating the DM's preferences: it decreases the computational effort and increases the algorithm efficiency on the solution region that best matches the DM's formulated preferences.

In Table 2, the best compromises are related to the outranking model's parameters that were set *a priori*. In multi-objective optimization, the DM 'learns' trade-offs while he/she finds and judges new Pareto solutions; thus his/her aprioristic preferences could be modified. Once the best compromise and others non-strictly outranked solutions have been obtained and evaluated by the DM, the model's parameter setting may be updated, perhaps using PDA as proposed in [63]. If the parameter values were modified, with an additional NO-ACO run the final best compromise should be reached.

5.3 Solving problems with high dimensionality

The test in the previous section was limited to 100 projects and nine objectives. These dimensions exceed those addressed by most studies in the specialized literature (e.g. [5,16,17,18,19,68]). These dimensions are appropriate for most portfolio problems in the business sector; however, in public organizations, the problem size may be larger. In order to explore the capacity of our algorithm to solve instances with a large size, we generated a set of instances with 500 projects and 16 criteria to optimize.

The interpretation is similar to that described at the beginning of this section: there is a budget of 250 million dollars to distribute, and the DM wants to keep a balance so has grouped the projects into two areas and three regions and imposed budgetary constraints for each (30%–70% for each area and 20%–60% for each region).

In addition, the DM has identified 100 relevant interactions between projects: 20 are cannibalization phenomena, 30 correspond to redundant projects and 50 are synergies that generate added value.

Unlike the 100-projects case, it is not possible in these instances to generate an acceptable approximation of the Pareto frontier that can be used as reference for comparison purposes. Even the best multi-objective algorithms are degraded when they attempt to generate it. This is combined with computation times that would be intolerable or with an abrupt interruption of the algorithms if they fail to converge towards the frontier.

In order to test the quality of the solutions suggested by our proposal, a comparison with a popular acceptable way of allocating resources can be performed. Among several heuristics frequently used, we chose one based on assigning budgetary resources according to project-ranking information. Here, a project ranking is built by using a cost-benefit ratio; the benefit is modelled by a weighted sum, whose weights are adjusted to reflect the DM's preferences. The project ranking is built following the order given by the cost-benefit ratio. Once the set of projects has been ranked, the resources may be allocated by following the priorities implicit in the rank order until no resources are left. This at least ensures the inclusion of projects that provide more benefit per dollar. Synergism can be tackled if the project interactions are modelled as dummy projects that can also be ranked.

Table 3 concentrates on only nine of the 164 solutions found by NO-ACO as an approximation to the non-strictly-outranked frontier. Our algorithm converges after 41,625 seconds. The best compromise that was found (Solution 1) outperforms the ranking-based portfolio, even in the Pareto sense.

Another ten instances were generated following the same features. When they were solved by NO-ACO, we observed the same behaviour: the ranking-based portfolio was dominated by the best compromise found by NO-ACO. This test gives some evidence of the applicability of our approach to large-scale real instances.

6 Conclusions and future work

We have presented an original proposal to optimize interdependent projects portfolios. This proposal is an adaptation of the well-known ant colony optimization metaheuristic, but incorporates preferences based on the outranking model of Fernandez et al. [10]. Our algorithm (NO-ACO) searches for optimal portfolios in synergetic conditions and can handle interactions impacting both objectives and costs. Redundancy is also considered during portfolio formation. By incorporating preferences, the selective pressure toward a privileged zone of the Pareto frontier is increased. Thus, a zone that matches the DM's preferences better can be identified. In comparison with other metaheuristic approach that does not incorporate preferences, NO-ACO achieves greater closeness to the region of interest with less computational effort. Our result seems to confirm the hypothesis from [10,15]: the incorporation of DM preferences by solving Problem (9) helps to obtain solutions that dominate others from leading metaheuristics.

Since it is enriched by preferences, our proposal acquires the ability to find good solutions (the known best portfolio) to portfolio problems with higher dimensions (in project and objective spaces) than those reported in scientific literature. Compared to the popular ranking-based method, NO-ACO finds solutions that outperform to the ranking-based portfolio, both in Pareto dominance and in strict outranking.

As immediate work we are going to explore the limits of this approach, by finding the greatest size of the instances that can be solved with an acceptable performance. Additionally, we are going to incorporate an interactive process for updating the preference model according to the new information gained by the DM from the optimized solutions.

Acknowledgements

We acknowledge the support from PROMEP through the network project 'Optimization and Decision Support'. We also thank to DGEST and the Autonomous University of Sinaloa (PROFAPI-055/2012 project).

References

[1] D N Kleinmuntz. *Portfolio Decision Analysis: Improved methods for resource allocation*, chapter Foreword, pages v–vii. Springer, New York-Dordrecht-Heidelberg-London, 2011.

[2] A Salo, J Keisler, and A Morton. *Portfolio Decision Analysis: Improved methods for resource allocation*, chapter An invitation to Portfolio Decision Analysis, pages 3–27. Springer, New York-Dordrecht-Heidelberg-London, 2011.

[3] M A Coffin and B W Taylor. Multiple criteria R&D project selection and scheduling using fuzzy sets. *Comput Oper Res*, 23(3):207–220, 1996.

[4] J Klapka, P Pinos, and V Sevcik. Multicriterial projects selection. *Handbook of Optimization, Intelligent Systems Reference Library*, 38:245–261, 2013.

[5] C Stummer and K Heidenberger. Interactive R&D portfolio analysis with project interdependencies and timeprofiles of multiple objectives. *IEEE T Eng Manage*, 50(2):175–183, 2003.

[6] J L Ringuest, S B Graves, and Case R H. Mean-gini analysis in R&D portfolio selection. *Eur J Oper Res*, 154(1):157–169, 2004.

[7] C Carlsson, R Fuller, M Heikkila, and Majlender P. A dynamic and fuzzy modeling approach for multi-objective R&D project portfolio selection. *Int J Approx Reason*, 44(2):93–105, 2007.

[8] X Zhao, Y Yang, G Wu, J Yang, and X Xue. A dynamic and fuzzy modeling approach for multi-objective R&D project portfolio selection. *J Converg Inf Technol*, 7(1):36–44, 2012.

[9] W Hallerbach, H Ning, A Soppe, and J Spronk. A framework for managing a portfolio of socially responsible investments. *Eur J Oper Res*, 153(2):517–529, 2004.

[10] E Fernandez, E Lopez, G Mazcorro, R Olmedo, and C A Coello Coello. Application of the Non-Outranked Sorting Genetic Algorithm to public project portfolio selection. *Inf Sci*, 228:131–149, 2013.

[11] Georgia Department of Transportation. project list and final investment report. available in http:// www.dot.ga.gov/ localgovernment/ fundingprograms/ transreferendum/ pages/projectlist.aspx (october 4th, 2012).

[12] Georgia Department of Transportation. Central Savannah River Area, unconstrained project list by county. available in http:// www.it3.ga.gov/ documents/ unconstrainedlist/ centralsavannah-unconstrainedlist.pdf (october 4th, 2012).

[13] Georgia Department of Transportation. Heart of Georgia, Altamaha unconstrained project list by county. available in http:// www.it3.ga.gov/ documents/ unconstrainedlist/ heartofgeorgia-unconstrainedlist-fullset.pdf (october 4th, 2012).

[14] Georgia Department of Transportation. River Valley Area, unconstrained project list by county. available in http:// www.it3.ga.gov/ documents/ unconstrainedlist/ rivervalley-unconstrainedlist.pdf (october 4th, 2012).

[15] E Fernandez, E Lopez, F Lopez, and C A Coello Coello. Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method. *Inf Sci*, 181(1):44–56, 2011.

[16] K F Doerner, W J Gutjahr, R F Hartl, C Strauss, and C Stummer. Pareto Ant Colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Ann Oper Res*, 131(1–4):79–99, 2004.

[17] K F Doerner, W J Gutjahr, R Hartl, C Strauss, and C Stummer. Pareto Ant Colony Optimization with ILP preprocessing in multiobjective project portfolio selection. *Eur J Oper Res*, 171(3):830–841, 2006.

[18] A F Carazo, T Gomez, J Molina, A G Hernández-Díaz, F M Guerrero, and R Caballero. Solving a comprehensive model for multiobjective project portfolio selection. *Comput Oper Res*, 37(4):630–639, 2010.

[19] A F Carazo, I Contreras, T Gomez, and F Perez. A project portfolio selection problem in a group decision-making context. *J Ind Manage Optim*, 8(1):243–261, 2012.

[20] C L Hwang and A S M Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Lecture notes in economics and mathematical systems. Springer-Verlag, 1979.

[21] F Ghasemzadeh, N Archer, and P Iyogun. A zero-one model for project portfolio selection and scheduling. *J Oper Res Soc*, 50(7):745–755, 1999.

[22] B Amiri. A multi-objective hybrid optimization algorithm for project selection problem. *J Basic Appl Sci Res*, 2(7):6995–7002, 2012.

[23] T Kremmel, J Kubalik, and S Biffl. Software project portfolio optimization with advanced multi-objective evolutionary algorithm. *Appl Soft Comput*, 11(1):1416–1426, 2011.

[24] A Chen and Ch Chyu. Applying memetic algorithm in multi-objective resource allocation among competing projects. *J Softw*, 5(8):802–809, 2010.

[25] J Gaytán and J García. Multicriteria decision on interdependent infrastructure transportation projects using an evolutionary-based framework. *Appl Soft Comput*, 9(2):512–526, 2009.

[26] S Ghorbani and M Rabbani. A new multi-objective algorithm for a project selection problem. *Adv Eng Softw*, 40(1):9–14, 2009.

[27] C M Lin and M Gen. Multicriteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. *Expert Syst Appl*, 34(4):2480–2490, 2008.

[28] W J Gutjahr, S Katzensteiner, P Reiter, C Stummer, and M Denk. Multi-objective decision analysis for competence-oriented project portfolio selection. *Eur J Oper Res*, 205(3):670–679, 2010.

[29] E Zitzler, M Laumanns, and L Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.

[30] H Ishibuchi, N Tsukamoto, and Y Nojima. Evolutionary many-objective optimization: A short review. In *Proceedings of Evolutionary Computation, 2008 (CEC'2008)*, pages 2419–2426. IEEE, 2008.

[31] D W Corne and J D Knowles. Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO'07, pages 773–780, New York, NY, USA, 2007.

[32] J López, C A COello Coello, K Oyama, and Fujii K. Alternative preference relation to deal with many-objective optimization problems. In R C Purshouse, P J Fleming, C M Fonseca, S Greco, and J Shaw, editors, *Proceedings of Evolutionary Multi-Criterion Optimization - EMO 2013*, Lecture Notes in Computer Science, pages 291–306. Springer-Verlag, 2013.

[33] R C Purshouse and P J Fleming. Evolutionary many-objective optimisation: An exploratory analysis. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 3, pages 2066–2073. IEEE, 2003.

[34] S Huband, L Barone, P Hingston, While R L, D Tuppurainen, and R Bearman. Designing comminution circuits with a multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation*, pages 1815–1822, 2005.

[35] G A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

[36] K Deb, J Sundar, U Bhaskara, and S Chaudhuri. Reference point based multiobjective optimization using evolutionary algorithms. *Int J Comput Intell Res*, 2(3):273–286, 2006.

[37] S Bechikh. *Incorporating Decision Maker's Preference Information in Evolutionary Multi-objective Optimization*. PhD thesis, High Institute of Management of Tunis, University of Tunis, Tunis, Tunisia, 2013.

[38] K Deb. Multi-objective evolutionary algorithms: Introducing bias among pareto optimal solutions. Technical Report KanGAL Report 99002, Indian Institute of Technology, Kanpur, India, 1999.

[39] J Branke and K Deb. *Knowledge Incorporation in Evolutionary Computation*, chapter Integrating user preferences into evolutionary multi-objective optimization, pages 461–478. Springer-Verlag, Berlin-Heidelberg, 2004.

[40] E Zitzler, D Brockhoff, and L Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Proceedings of the 4th international conference on Evolutionary Multi-criterion Optimization (EMO'07)*, pages 862–876, Matsushima, Japan, 2007. Springer-Verlag.

[41] G W Greenwood, X S Hu, and J C D'ambrosio. *Foundations of Genetic Algorithms*, chapter Fitness functions for multiple objective optimization problems: combining preferences with Pareto rankings, pages 437–455. Morgan Kaufmann, 1996.

[42] K Deb, A Sinha, P Korhonen, and J Wallenius. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE T Evolut Comput*, 14(5):723–739, 2010.

[43] M Köksalan and I Karahan. An interactive territory defining evolutionary algorithm: itdea. *Evolut Comput*, 14(5):702–722, 2010.

[44] R Battiti and A Passerini. Brain-computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker. *IEEE T Evolut Comput*, 14(5):671–687, 2010.

[45] J W Fowler, ES Gel, M Köksalan, P Korhonen, J L Marquis, and J Wallenius. Interactive evolutionary multi-objective optimization for quasi-concave preference functions. *IEEE T Evolut Comput*, 206(2):417–425, 2010.

[46] J Branke, S Greco, R Slowinski, and P Zielniewski. Interactive evolutionary multiobjective optimization driven by robust ordinal regression. *Bull Polish Acad Sci*, 58(3):347–358, 2010.

[47] Y C Jin and B Sendhoff. Incorporation of fuzzy preferences into evolutionary multiobjective optimization. In *Proceedings of the 4th Asia-Pacific conference on Simulated Evolution and Learning*, Nanyang, Singapore, 2002.

[48] D Cvetkovic and I C Parmee. Preferences and their application in evolutionary multiobjectiveoptimisation. *IEEE T Evolut Comput*, 6(1):42–57, 2002.

[49] L Rachmawati and D Srinivasan. Incorporating the notion of relative importance of objectives in evolutionary multiobjective optimization. *IEEE T Evolut Comput 2010*, 14(4):530–546, 2010.

[50] K Deb and A Kumar. Interactive evolutionary multi-objective optimization and decision making using reference direction method. In *Proceedings of the 9th Genetic and*

Evolutionary Computation Conference (GECCO'07), pages 781–788, London, UK, 2007. ACM.

[51] R Allmendinger, X Li, and J Branke. Reference point-based particle swarm optimization using a steady-state approach. In *Proceedings of the 7th international conference on Simulated Evolution and Learning (SEAL'08)*, pages 200–209, Melbourne, Australia, 2008. Springer-Verlag.

[52] J Molina, L V Santana-Quintero, A G Hernández-Díaz, C A Coello Coello, and R Caballero. g-dominance: Reference point based dominance for multiobjectivemetaheuristics. *Eur J Oper Res*, 197(2):685–692, 2009.

[53] S Bechikh, S L Ben, and K Ghédira. The r-dominance: a new dominance relation for preference-based evolutionary multi-objective optimization. Technical Report BS-2010-001, SOIE Research Unit, High Institute of Management of Tunis, University of Tunis, Tunis, Tunisia, 2010.

[54] A López, A Arias, and C A Coello Coello. Preference incorporation to solve many-objective airfoil design problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'11)*, pages 1605–1612, New Orleans, 2011. IEEE.

[55] J Branke, T Kaussler, and H Schmeck. Guidance in evolutionary multi-objective optimization. *Adv Eng Softw*, 32(6):499–507, 2001.

[56] E Fernandez, E Lopez, S Bernal, C A Coello Coello, and J Navarro. Evolutionary multiobjective optimization using an outranking-based dominance generalization. *Comput Oper Res*, 37(2):390–395, 2010.

[57] T Wagner and H Trautmann. Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE T Evolut Comput*, 14:688–701, 2010.

[58] B Roy. *Multicriteria Methodology for Decision Aiding*. Nonconvex Optimization and Its Applications. Springer, 1996.

[59] B Roy. *Reading in multiple criteria decision aid*, chapter The Outranking Approach and the Foundations of ELECTRE methods, pages 155–183. Springer-Verlag, 1990.

[60] J Brans and B Mareschal. *Multiple Criteria Decision Analysis: State of the Art Surveys*, chapter Promethee methods, pages 163–190. International Series on Operations Research & Management Science. Springer–Verlag, Berlin, 2005.

[61] E Jacquet-Lagreze and Y Siskos. Preference disaggregation: 20 years of mcda experience. *Eur J Oper Res*, 130(2):233–245, 2001.

[62] M Doumpos, Y Marinakis, M Marinaki, and C Zopounidis. An evolutionary approach to construction of outranking models for multicriteria classification: The case of the ELECTRE TRI method. *Eur J Oper Res*, 199(2):496–505, 2009.

[63] E Fernandez, J Navarro, and G Mazcorro. Evolutionary multi-objective optimization for inferring outranking model's parameters under scarce reference information and effects of reinforced preference. *Found Comput Decis Sci*, 37(3):163–197, 2012.

[64] M Dorigo and L M Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE T EvolutComput*, 1(1):53–66, 1997.

[65] A Berrichi, F Yalaoui, L Amodeo, and M Mezghiche. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. *Comput Oper Res*, 37(9):1584–1596, 2010.

[66] S Chaharsooghi and A H M Kermani. An effective ant colony optimization algorithm for multi-objective resource allocation problem. *Appl Math Comput*, 200(1):167–177, 2008.

[67] B Yagmahan and M M Yenisey. Ant colony optimization for multi-objective flow shop scheduling problem. *Comput Ind Eng*, 54(3):411–420, 2008.

[68] J Liesiö, P Mild, and A Salo. Robust portfolio modeling with incomplete cost information and project interdependency. *Eur J Oper Res*, 190(3):679–695, 2008.

Laura Cruz received the M.Sc. degree in Computer Science from Instituto Tecnológico y de Estudios Superiores de Monterrey (Mexico) in 1999 and she received the Ph.D (Computer Science) degree from Centro Nacional de Investigación y Desarrollo Tecnológico (Mexico) in 2004. She is a full time professor at Madero Institute of Technology, Mexico. Her research interests include optimization techniques, complex networks, autonomous agents and algorithm performance explanation. Laura Cruz is a member of the Mexican National System of Researchers and the Mexican Society of Operation Research. Professor Cruz is referee of some international journals in the frame of metaheuristic optimization.

Eduardo Fernandez received the Ph.D degree in Computer Aided Design of Electronic Circuits, from Poznan University of Technology, 1987. He is currently Senior Professor of the Faculty of Civil Engineering, Autonomous University of Sinaloa (UAS), Mexico. His main areas of interest are multi-criteria and intelligent decision support, multi-objective optimization, group decision models and project portfolio selection. In those fields he has published more than eighty papers and book chapters. Professor Fernandez is a member of the Mexican National System of Researchers, the Mexican Society of Operation Research, the International EUREKA Network for Knowledge Discovery, Knowledge Management and Decision Support, the International Society on Multi Criteria Decision Making and the Euro Working Group on Multi-Criteria Decision. He has been nominated three times for "OR in Development" Prize.

Claudia Gomez was born in Mexico in 1971. She is a full time professor at Madero Institute of Technology. She received the Ph.D degree in Advanced Technology from National Polytechnic Institute (Mexico), in 2009. She received the M.Sc. degree in Computer Science from Leon Institute of Technology (Mexico), in 2000. Her research interests are optimization techniques, complex network and autonomous agents.

Gilberto Rivera was born in Mexico in 1984. He is a Ph.D student at Inter-institutional Network of Tijuana-Madero-and-Leon Intitutes of Technology. He received the M.Sc. degree in Computer Science from Madero Institute of Technology (Mexico), in 2010. His research interests are in the areas of project portfolio selection, optimization techniques, swarm intelligence and autonomous agents.

Fatima Perez is a Research Fellow at University of Malaga (Spain). She received the Ph.D degree in Mathematical Economics at University of Malaga. She has published research articles in reputed international journals in the frame of applied mathematics and applied economics. Her research interests are in the areas of project portfolio selection, multiobjective programming, metaheuristic algorithms, development of computer software and composite indicators. She is referee of some international journals in the frame of multiobjective programming.