

A Cultural Algorithm for the Urban Public Transportation*

Laura Cruz Reyes¹, Carlos Alberto Ochoa Ortíz Zezzatti², Claudia Gómez Santillán^{1,3},
Paula Hernández Hernández¹, and Mercedes Villa Fuerte¹

¹Instituto Tecnológico de Ciudad Madero, División de Estudios de Posgrado e Investigación Juventino Rosas y Jesús Urueta s/n, Col. Los mangos, C.P. 89440, Cd.Madero,Tamp, México

²Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez, Henry Dunant 4016, Zona Pronaf, C.P. 32310, Cd. Juárez, Chihuahua, México

³Instituto Politécnico Nacional, Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada. Carretera Tampico-Puerto Industrial Altamira, Km.14.5. Altamira,Tamp., México

lcruzreyes@prodigy.net.mx,

{megamax8, cggs71, paulahdz314, pvmercedes}@hotmail.com

Abstract. In the last years the population of Leon City, located in the state of Guanajuato in Mexico, has been considerably increasing, causing the inhabitants to waste most of their time with public transportation. As a consequence of the demographic growth and traffic bottleneck, users deal with the daily problem of optimizing their travel so that to get to their destination on time. To give a solution to this problem of obtaining an optimized route between two points in a public transportation, a method based on the cultural algorithms technique is proposed. Cultural algorithms are used in the generated knowledge in a set of time periods for a same population, using a belief space. These types of algorithms are a recent creation. The proposed method seeks a path that minimizes the time of traveling and the number of transfers. The results of the experiment show that the technique of the cultural algorithms is applicable to these kinds of multi-objective problems.

Keywords: Cultural Algorithms, Evolutionary Algorithms, Optimization, Transport, Population, Beliefs, Agent.

1 Introduction

The population growth has increased the traffic jam and, as a result, the demand of public services, such as transportation, is rising rapidly. In these circumstances, an effective public transportation is a necessity. To move from one place to another users are faced with the problem of optimizing resources in a transport system with many kinds of goals; for example, to minimize the number of stations, path length, travel mode, and travel time. Many times, solving this routing problem involves conflicts in goals. A solution with the fewer number of visited stations and the highest travel time is an example of conflict in goals.

* This research was supported in part by CONACYT, DGEST and IPN.

For the multi-objective nature, the public transportation problem can be solved through evolutionary computation. The use of evolutionary algorithms has increased to solve several problems of multi-objective optimization which occur in the society, among the public transportation optimization.

A wide bibliographical revision revealed that Cultural Algorithms (CAs) have not been applied yet to the public transportation problem. Reynolds developed the cultural algorithms as a complement to the metaphor which inspired the evolutionary algorithms (natural selection and genetic concepts) [3]. CAs are an evolutionary computation technique, that uses the knowledge that has been generated in several times (same individuals, but in different time-space), for the same population, using a belief space, this characterizes to CAs, and they make an area of great interest for research. Two problems that have been implemented CAs are: Diorama's Representation using a Mosaic Image [4] and Public Security System [6].

For the previously exposed, in this article is presented a solution method for the public transportation problem based in CAs. The proposed method looks for a path that minimizes the travel time and the number of transfers, which are objectives in conflict. The Public Transportation Problem is presented as a case of study. On one hand system users save money using the path with less transfers and the other hand, they save time using the path of shortest travel time. When looking for the optimal path, first the algorithm considers the path with the least travel time, and then minimizes the number of transfers.

2 Problem Description

Instance: Given a transport network formed by a directed graph $G = \{V, E\}$, where V is a set of vertexes that represent the bus stations, and a set of edges E that represent the routes that connect the stations. The V contains the information of the previous and next stop. The E is identified with the route number NR that joins the stations, the sense that follows the route and the time T required between two stations.

An initial station v_0 , an end station v_n , a time t_0 and a date of consultation is provided. The search begins from v_0 , covering the edges $v_0 + i$ that allows it to arrive v_n , accumulating $t_a = t_a + i$ that took to arrive and the number of route changes or transfers done $nr = nr + 1$. The process finishes when $v_i = v_n$.

Objective: To Find a set of routes NR and continuous terminals for going from v_0 to v_n , in such a way, that minimizes t_a and NR , to arrive v_n .

3 Description of Cultural Algorithms

Cultural algorithms employ a basic set of knowledge sources, each related to knowledge observed in various social species. These knowledge sources are then combined to direct the decisions of the individual agents in solving optimization problems [1]. Cultural algorithms consist of two main components: a population space, belief space and a communication protocol [2]. It is the following:

Population space, this space maintains a set of individuals, which represent potential solutions to the problem. Each individual possesses its characteristics, and these characteristics define its fitness in the environment. Through different time periods called generations, individuals can be replaced by their descendants, obtained by means of the application of the operator that somehow affects the population [3].

Belief space is the space where the knowledge acquired by the individuals through the generations will be stored, and this must be accessible to any individual in the population, and can be used to influence its behavior, for example to modify its characteristics and then modify its fitness [3].

Communication protocol describes the way in which the knowledge is exchanged between the population space and belief space [2]. It states the rules about the individuals that can contribute to the belief space with their experiences through the acceptance function. The influence function through the belief space can influence to the new individuals [5].

Algorithm 1. Pseudo code of a cultural algorithm [3].

```

1   initialize_population ( $P = \{x_1, \dots, x_n\}$ )
2   evaluate_population ( $P$ )
3   initialize_beliefespace( $B$ )
4   repeat
5       update( $B$ , accept( $P$ ))
6        $P' = \text{influence}(\text{operators}(\mathcal{P}))$ 
7       evaluate_population ( $P'$ )
8        $P = \text{selection}(P')$ 
9   until the termination condition is achieved

```

The Algorithm 1 shows the main steps of a Cultural Algorithm. Most of the steps of a cultural algorithm are similar to those of a standard evolutionary algorithm, for example lines 1 and 2. The novel thing of these algorithms is the use of a belief space B . The line 3 is the initialization of the belief space. The details of this procedure depend on the structure of the belief space, and may be different in each case.

The update (also called *adjust*) function of the belief space is in the main loop (line 4 to line 9). The algorithm incorporates new experiences to the belief space, from a selected group of individuals P . In the line 5 the function *accept*, selects the individuals of this group which are chosen from the population.

In the line 6, the variation operators of the algorithm are modified by the function *influence*. Finally in the line 7, the population influenced by the belief space is evaluated to choose the best option.

The *selection* function generates a solution to the search space. It is influenced from five sources of knowledge basic [7]. The best solution will be part of the belief space; this step is shown in line 8.

The Cultural knowledges of any cultural evolution model are: situational, normative, topographic, historical or temporal, and domain knowledge [7].

A representation of the interactions in the elements of Cultural Algorithms is shown as a diagram in Figure 1 [3].

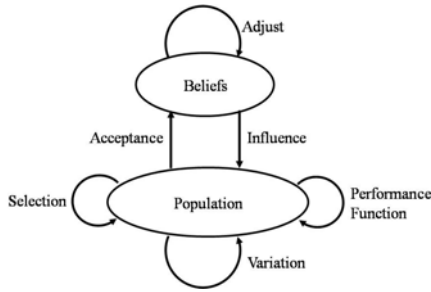


Fig. 1. Representation of Cultural Algorithms

4 Experiments and Results

The Public Transportation Problem (Characterized by Routing and Scheduling) is presented as a case of study. On one hand system users save money using the path with less transfers and the other hand, they save time using the path of shortest travel time. When looking for the optimal path, first the algorithm considers the path with the least travel time, and then minimizes the number of transfers. Transport net is represented by a graph G , and for modeling had been employed 3 tables described bellow:

tbase_rutas: contains general information about the routes. This table is formed by an identifier field that is used to link other tables; a field that contains the route name that is used to indicate the user the number of the route (bus) that he must take; a field that stores the route type as an additional information that identifies the bus for the user and finally a field that contains the route direction that is used in the intern process of solution construction, this information also presented to the user.

tparadas: contains information about the realized stations for each route that is important for the tour's construction, from which is taken the specification about the places of the needed stations for the transfers that are suggested to the user. This table is formed by an identifier field of the route that is used to link other tables to obtain additional information; a field that contains the name of each station that the bus does; a field that contains the previous station to the current station; a field that contains the next station to the current station and finally a field that stores the necessary time for arriving from the current station to the next station with the current route.

thorarios: contains schedule information of the beginning and end daily departures for each route. This table is formed by an identifier field for the route, with the object to obtain additional information; a field to store the place of the route departure; a field that contains the first departure hour of the route; a field to store the last departure hour of the route; a field to store an interval time for the departure of each route; a field that contains the working days for that route and finally a field to specify if the route departs from the origin or the destination of the route.

For the data base development, a route map of the city and the service schedule for the routes are provided; using for the table's filling the information described before. A part of the map that was included in the application is in the Figure 2.

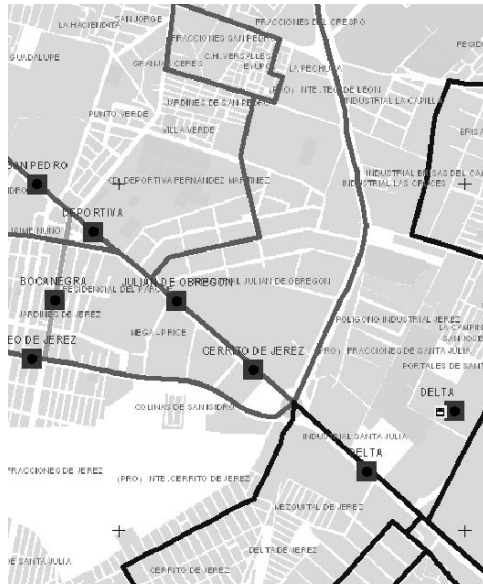


Fig. 2. Map of the used quadrant for the test and prototype elaboration. The brought out lines with dark color are the streets where the public transportation bus passes by.

After we have filled the tables, the next step is the construction of waypoints (routes). "Algorithm 2" shows the main steps to construct a route.

Algorithm 2. The construction of a route.

```

GenerateRoute (Origin, Destination, Time, Date)
1  initialize pActual=Origin, pPrior=Origin,
   Routes="", Stations=""
2  do
3      consult existence of routes that pass through
   p_Actual and Destination and available for time
   and date requested.
4  IF exist
5      route=selectQueryPath()
6  ELSE
7      consult if routes that pass through pActual exist
   and available for time and date requested.
8      route=selectQueryPath()
9  pActual=next(pActual)
10 pPrior_next=next(pPrior)
11 IF (pActual=pPrior_next)
12     route=routePrior
13 IF change of route {
14     Routes=routes+route
15     Stations=Stations+pActual }
16 until pActual=Destination

```

In the algorithm, a consult is made (lines 3 and 5) with the table *tparas* to check the existence of any route that passes through the actual stop and that it also passes through destination stop. If several routes exist, you choose one of them; we execute the opposite case (lines 7 and 8), where we verify again the table *tparas* to see routes that pass through that stop and we select any of those routes. By means of the table *tparas* (lines 9 to 12) a search for the next stop according to the chosen route is done. If in the selected route, its next stop is the same as the next stop from the previous route, then the previous route is selected, this is done to reduce the number of transfers. The solution keeps storing every time a transfer occurs (lines 13 to15); this process repeats itself until we arrive to the destination that checks in line 16.

For the elaboration of the tests, we took in count the next requests as instances. In Table 1, we only show an example of a request. The requests were selected for these tests due to the traveled distances for getting to the original destiny are long, besides that in this places many options of routes pass through, this is why that possibilities of arriving to the destination are plenty. We carry on 5 executions of the application per instance. We made requests for 3 different hours so to observe the results accomplished in each execution.

Table 1. Instance Request example

Origin:	EYUPOL.Popular
Destination:	PASEO DE JEREZ
Hour:	12:00
Date:	01/08/2009
Agents:	3
No. improvement periods:	9

After the execution of the application with the previously described data, we obtain the results shown in Table 2. This results show us estimated time in minutes and the number of used buses, after executing the instance example 5 times. In Table 3 we can observe the evolution of periods and how their solution improves by measuring estimated time in minutes and the number of used buses, for the instance example and only one execution.

Table 2. Results of 5 executions of the application for the instance example

Origin	Destination	Time estimated in minutes	Number of used buses
EYUPOL.Popular	PASEO DE JEREZ	15	2
EYUPOL.Popular	PASEO DE JEREZ	15	2
EYUPOL.Popular	PASEO DE JEREZ	15	2
EYUPOL.Popular	PASEO DE JEREZ	15	2
EYUPOL.Popular	PASEO DE JEREZ	15	2

Table 4 shows a concentrate of all requests executing 5 times each. Using the same Origin and Destination but changing the hour of request. Obtaining the average of time in minutes and used buses. After the result analysis we observe at what hour the request is a meaningful factor for the experiment's efficiency.

Table 3. Results by period of the first execution for the instance example

Period	Origin	Destination	Time estimated in minutes	Number of used buses
1	EYUPOL.Popular	PASEO DE JEREZ	39	8
2	EYUPOL.Popular	PASEO DE JEREZ	33	6
3	EYUPOL.Popular	PASEO DE JEREZ	15	2
4	EYUPOL.Popular	PASEO DE JEREZ	15	2
5	EYUPOL.Popular	PASEO DE JEREZ	15	2
6	EYUPOL.Popular	PASEO DE JEREZ	15	2
7	EYUPOL.Popular	PASEO DE JEREZ	15	2
8	EYUPOL.Popular	PASEO DE JEREZ	15	2
9	EYUPOL.Popular	PASEO DE JEREZ	15	2
10	EYUPOL.Popular	PASEO DE JEREZ	15	2
11	EYUPOL.Popular	PASEO DE JEREZ	15	2

Table 4. Test results with different instances and hours

Origin	Destination	Date	Hour	Time Avg. in minutes	Avg. of used buses
San Pedro	Industrial Delta.Beta	01/08/2009	06:00	14.4	2.6
San Pedro	Industrial Delta.Beta	01/08/2009	12:00	14.6	2.4
San Pedro	Industrial Delta.Beta	01/08/2009	22:00	14.2	2.6
CERRITO DE JEREZ.Cerrito de Jerez	ITL.Prol.Españita	01/08/2009	06:00	26.6	4
CERRITO DE JEREZ.Cerrito de Jerez	ITL.Prol.Españita	01/08/2009	12:00	25.8	4
CERRITO DE JEREZ.Cerrito de Jerez	ITL.Prol.Españita	01/08/2009	22:00	27	3.6
EYUPOL.Popular	PASEO DE JEREZ	01/08/2009	06:00	15	2
EYUPOL.Popular	PASEO DE JEREZ	01/08/2009	12:00	15	2
EYUPOL.Popular	PASEO DE JEREZ	01/08/2009	22:00	16	2.5
DEPORTIVA	DELTA.Estacion	01/08/2009	06:00	9.6	1.4
DEPORTIVA	DELTA.Estacion	01/08/2009	12:00	9.4	1.6
DEPORTIVA	DELTA.Estacion	01/08/2009	22:00	9.2	1.8

5 Conclusions and Future Work

For solving the path generating problem of public transportation is proposed a Cultural Algorithm. In general, the Cultural Algorithms consist of three elements: population space, belief space and the communication protocol. The proposal is centered in

the adaptation of the belief space. The algorithm takes the best proposed path in each period.

The experimentation results show that this path has a big influence in the generation of new paths for subsequent periods. Because of that, we can assure that the belief space is a decisive factor for finding the best solution. The proposed algorithm obtains good results in comparison with handbook estimates done by experts in the area of generating transportation paths.

In addition, this analysis confirms that the parameter of application time is a significant factor to generate the problem solution, because when this parameter changes the solution changes considerably too.

As future work, it is intended to expand the experimentation with other kind of evolutionary algorithms and other instances, because it was considered only one quadrant of the Leon City, Guanajuato. For this study is necessary to extend the database to include all Public Transportation Infrastructure of the City. Another line of work is the extension of the Cultural Algorithm to incorporate objectives and restrictions that are also present in the system in study.

References

1. Reynolds, R.G., et al.: Cultural Algorithms: Knowledge-driven engineering optimization via weaving a social fabric as an enhanced influence function. In: IEEE Congress on Evolutionary Computation 2008 (2008)
2. Ochoa, A.: Algoritmos Culturales. Gaceta Ideas Concyteg 3(31) (2008)
3. Becerra, R.L.: Doctoral thesis: Use of Domain Information to Improve the Performance of an. In: Center for Research and Advanced Studies of The National Polytechnic Institute of Mexico, Computer Science Department, Mexico City, Mexico (June 2007)
4. Ochoa, A., et al.: Evolving Optimization to Improve Diorama's Representation using a Mosaic Image. Journal of Computers 4(8) (August 2009)
5. Becerra, R.L., et al.: Use of Domain Information to Improve the Performance of an Evolutionary Computation. In: Genetic And Evolutionary Computation Conference. Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation (2005)
6. Ochoa, A., et al.: Sistema de Seguridad Pública basado en inteligencia artificial. Publicación Trimestral de ADIAT, Año VIII, Núm. 33, Enero –Marzo de (2009)
7. Reynolds, R.G., Kobti, Z., Kohler, T.: Agent-Based Modeling of Cultural Change in Swarm Using Cultural Algorithms. In: Proceedings of SWARMFEST 2004, May 9-11. University of Michigan, Ann Arbor (2004)

A Visualization Tool for Heuristic Algorithms Analysis

Laura Cruz-Reyes¹, Claudia Gómez-Santillán¹, Norberto Castillo-García¹,
Marcela Quiroz¹, Alberto Ochoa², and Paula Hernández-Hernández¹

¹ Instituto Tecnológico de Ciudad Madero, División de Estudios de Posgrado e Investigación. Juventino Rosas y Jesús Urueta s/n, Col. Los mangos, C.P. 89440, Cd.Madero, Tamaulipas, México

lcruzreyes@prodigy.net.mx, {cggs71,norberto_castillo15,
paulahdz314}@hotmail.com, qc.marcela@gmail.com,

² Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez. Henry Dunant 4016, Zona Pronaf, C.P. 32310, Cd. Juárez, Chihuahua, México
doctor_albertoochoa@hotmail.com

Abstract. The performance of the algorithms is determined by two elements: efficiency and effectiveness. In order to improve these elements, statistical information and visualization are key features to analyze and understand the significant factors that affect the algorithm performance. However, the development of automated tools for this purpose is difficult. In this paper a visual diagnosis tool named VisTHAA, which provides researchers statistical and visual information about instances and algorithms, is proposed. Besides, VisTHAA allows researchers to introduce characterization measurements, make algorithm comparisons with a non-parametric test and visualize the search space ruggedness and the behavior of the algorithm. Due to the above, in this study we used VisTHAA as a tool for improving the efficiency of a reference algorithm in the literature. In the study case, the experimental results showed that through visual diagnosis it was possible to increment the algorithm's efficiency to 93% without a considerable loss of effectiveness.

Keywords: Algorithm's Performance, Visualization, VisTHAA.

1 Introduction

The objective of experimental algorithmics is to promote that the experiments are relevant, accurate and reproducible in order to produce knowledge to improve the design of algorithms and their performance accordingly [1]. Nowadays, it is not enough to know that an algorithm is the best in solving a particular set of instances, it is also important to understand and explain formally its behavior. The performance of heuristic algorithms depends on many factors including design quality; thus, a poor design can lead to a poor performance. There are no rules or guidelines widely accepted to design properly heuristics. Particularly, researchers

consume too much time to fine-tune an algorithm; usually they adjust their control parameters manually by trial and error requiring a considerable time.

In this paper VisTHAA (Visualization Tool for Heuristic Algorithm Design) is introduced, which is a modular visual diagnostic tool that will allow other researchers extend its features and functionalities through modules integrated into an architecture to facilitate the analysis of heuristics. The current version of VisTHAA has the following modules available: Input and Pre-processing data, characterization of the instances, visualization of the instances and the algorithm’s behavior, visualization of the three-dimensional search space, and the algorithm comparative analysis.

In the literature there are other tools similar to VisTHAA [2, 3] however none of them are able to allow the researchers to introduce their own characterization measurements or to visualize the search space in three dimensions. In the study case we use some features of VisTHAA, which make our tool different from the other ones, to solve the optimization problem under study: the Bin-Packing Problem (BPP).

The remaining paper is organized as follows. In Section 2 we show a detailed description of the proposed tool, emphasizing their main modules and the interaction among them. In Section 3 a study case is presented, the used methodology aims to identify areas of improvement, which allows the algorithm adjustment. Section 4 shows the analysis of the experimental results, using the well-known Wilcoxon Test to validate statistically the results. Finally, we discuss the most important remarks and some future work in Section 5.

2 VisTHAA

VisTHAA is a visual tool applied to the analysis of heuristic algorithms. Figure 1 shows only the modules considered for this initial release. The most innovative part are the characterization modules, which can be applied to the problem,

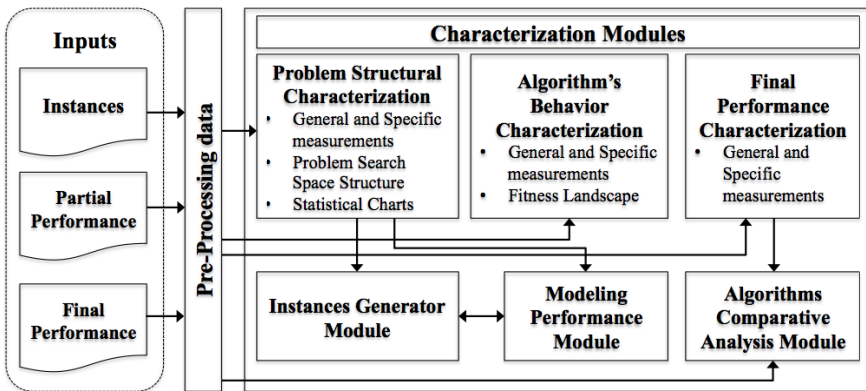


Fig. 1 Architecture of VisTHAA

algorithm, or final results, and contribute at every phase of the heuristic optimization process by identifying which factors have a significant influence on performance. This tool is available from <https://sth-se.diino.com/lauracruzreyes/VisTHAA/ExecutableCode>.

2.1 *Input and Pre-processing Data Module*

Researchers use their own formats for reading and processing data stored in files. When they want to analyze their data, they need to adjust their files to the input formats required for the available tool. It is complicated for them to change the format of their files. In the design of VisTHAA, the inputs may represent instances of a problem, algorithm data or performance data. In order to simplify the researcher's work, we design a preprocessing module. In the case of the input of the instances, different formats and different problems can be read.

2.2 *Characterization Module*

This module includes a set of options to characterize and quantify the factors that define the problem structure as well as partial and final algorithm performance. As we can see in Figure 1, in each characterization option exists a set of general-purpose measurements, in order to identify which factors impact on the heuristic optimization process, i.e., *mean*, *variance* and *mode*. Another contribution that is closely linked to the characterization module is the use of a calculator, which shows the instance parameter and general-purpose measurements. From this stored information the researcher is allowed to introduce new characterization measurements in infix notation. This module uses the shunting yard algorithm [4]. In the final performance module are available measurements proposed by Quiroz and Cruz [5, 10] to quantify the quality of the solutions found by the algorithm.

- a) **Search Space Structure:** Represent the objective function domain elements that are created randomly [6]. Search space structure is obtained by a random walk algorithm, which gives exactly the same probability of choosing to each possible solution of the instance, this is because random walk is a blind search technique, so it does not prefer search space areas with the greatest potential of finding the optimum, in [7] a detailed explanation of random walk can be found. Representing the search space through three-dimensional visualization techniques, helps the researcher to analyze how the instance structure is, that is, it is possible to see if the instance is rough or not, and make decisions about intensification and diversification strategies to be applied to each instance.
- b) **Statistical Charts:** Through them it is possible to visualize characteristics that describe the problem factors and in some cases identify patterns in them. VisTHAA allows the researchers use frequency charts, which characterize the distribution of values in a dataset. The scatter plots display the correlation between a pair of variables.
- c) **Algorithm's Behavior Fitness Landscape:** This kind of graphic shows the algorithm's behavior during its execution, displaying the fitness values obtained in each iteration giving the researcher visual arguments to identify if

the algorithm is improving or not, and then adjust the diversification and intensification strategies. A formal definition of fitness landscape can be found in [8].

- d) **Algorithms Comparative Analysis Module:** In this module the performance of two algorithms is evaluated. This evaluation is made through the Wilcoxon test, which is a non-parametric procedure that determines whether or not the differences between two datasets are statistically significant [9].

3 Study Case

The study case has a three-main-phases methodology: 1) data input, 2) instance characterization, and 3) instance visualization. Once these phases are concluded, if areas of improvement are observed, a redesign phase is performed. Finally the results are validated using the Wilcoxon test, which allows verifying if the results are better statistically.

The optimization problem under study is the Bin Packing Problem (BPP), which is one of the most studied optimization problems. BPP is classified as a NP-hard combinatorial optimization problem [10]. BPP consists in finding an objects distribution that minimizes the number of bins used. In practice, this kind of problems has been solved by heuristic approaches; the Weight Annealing Heuristic (WABP) proposed by Loh et al. [11], is one of them. Basically, WABP has four steps. The first step initializes the parameters; then, the second step constructs and initial solution using the FFD procedure; the third step is the main, it consists in trying to improve iterative the current solution through a set of swaps between some items from different bins; finally, the last step outputs the results.

3.1 Phase 1: Introducing the Instances to VisTHAA

In order to introduce the instances to VisTHAA, two additional files are required. The first one is named *metainstance*, which describes the format of the instances. The second one is named *logbook*, which is used to specify the number of instances to be introduced, the name of the file that describes the instances, and the name of each one of them. Figure 2 shows a BPP instance, the *metainstance* and the *logbook* files.

```

a) BPP Instance
//Number of items
60
//Bin Capacity
100.00
//Known Best Solution
20
//Weight of Item
36.60
26.80
36.60
43.00
26.30

b) Metainstance file
comments(1);
variables(1);
n int;
comments(1);
variables(1);
c int;
comments(1);
variables(1);
best int;
comments(1);
variables(1);
weights vector int n;

c) Logbook file
instances(10);
instructions_file("metainstance.txt");
instances_names()
{
Hard0.txt, Hard1.txt, Hard2.txt, Hard3.txt,
Hard4.txt, t60_00.txt, t60_01.txt,
t60_02.txt, t60_03.txt, t60_04.txt }

```

Fig. 2 Example of description of instances files for their correct introduction to VisTHAA

3.2 Phase 2: Characterization of BPP

After the instances are loaded, the researcher can perform the characterization of BPP instances. The statistical characterization is made through measurements. So far, it is possible to select between basic statistics or specialized measurements for BPP reported in literature. VisTHAA has implemented some of the specialized measurements for BPP [5, 10], which can be utilized to gain knowledge of the instances. Nevertheless, VisTHAA gives the researchers the opportunity to introduce their own measurements from variables already defined in the metainstance file.

Figure 3 shows an example of how to introduce a new measurement, in this case, the *normalized_range*. Once a set of measurements is introduced, the researcher can decide which of them to consider for generating the attributes matrix over the set of selected instances; the purpose is to obtain information of the calculated values over a set of instances.

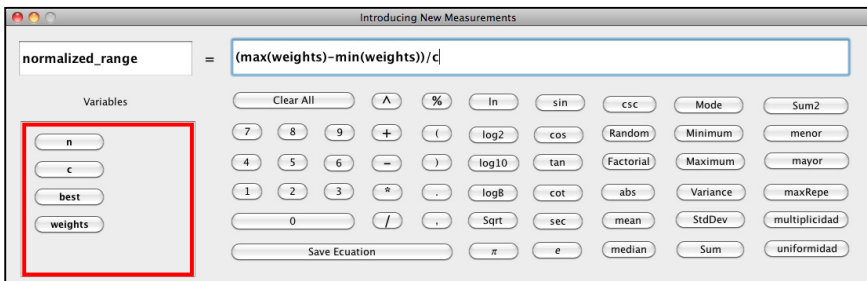


Fig. 3 Module to introduce new measurements to VisTHAA

Table 1 shows the attributes matrix over the loaded instances. It is observed that, for the measured attributes, there are significant differences among the analyzed instances; this is because the instances belong to a two different datasets. The most evident attribute is *uniformity* (attribute 5 in Table 1), which measure the level of uniformity of the weights distribution. In average, the instances belonging to the Hard dataset are highly uniform, with a value of 0.926 (92.6%), whereas the instances belonging to the T dataset are not so uniform, only a 0.473 (47.3%).

Table 1 Attributes matrix over the instances to be optimized

Instance	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8
Hard0	0.201	0.349	1.005	2	0.929	0.272	0.042	0.148
Hard1	0.200	0.349	1.000	1	0.950	0.276	0.043	0.149
Hard2	0.200	0.349	1.005	2	0.890	0.277	0.044	0.149
Hard3	0.200	0.347	1.015	2	0.940	0.272	0.042	0.147
Hard4	0.201	0.350	1.010	3	0.920	0.277	0.041	0.148
t60_00	0.251	0.495	1.200	3	0.466	0.333	0.072	0.244
t60_01	0.251	0.475	1.071	2	0.550	0.333	0.070	0.224
t60_02	0.250	0.498	1.250	3	0.466	0.333	0.081	0.248
t60_03	0.250	0.495	1.224	2	0.450	0.333	0.079	0.245
t60_04	0.250	0.498	1.250	3	0.433	0.333	0.078	0.248

From Table 1, the symbolism *Att* of the measured attributes represents: 1) *minor*, 2) *mayor*, 3) *multiplicity*, 4) *max repetitions*, 5) *uniformity*, 6) *normalized mean*, 7) *normalized standard deviation*, 8) *normalized range*. In the experiments of the next two sections only is considered the characteristic of uniformity.

3.3 Phase 3: Visualization of Instances and Algorithm Behavior

All of the experiments in this section were performed under the next hardware specifications: Intel Atom processor at 1.67 GHz and 1 GB of RAM. In order to analyze WABP heuristic and two variants, instances from recognized benchmarks were used, these instances can be found in a well-known Internet sites [12, 13].

The algorithm behavior of WABP is analyzed primarily with visual support. The initial configuration of this algorithm is: scaling = 0.05; iteration number = 50; initial temperature = 1; and the temperature reduction = 0.05.

The visualization of the search space is another way to characterize the problem structure. The literature states that the ruggedness of an instance is related with its difficulty [5].

To build a graph of the search space, first we extract a representative sample of the different solutions of the instance with their corresponding fitness, through a random walk. The resulting one-dimensional data set (fitness values) is converted into a two-dimensional set through performing "K" partitions of length "I". Finally, for each solution, we get the value of the X-axis (the position relative to K), the value of the Y-axis (the position relative to I) and the value of the Z-axis (the fitness of the solution). With these three variables can be graphed the search space in three dimensions. This simplified graph shows the neighborhood structure partially.

Figure 4 shows two graphics of the fitness landscapes that correspond to two types of instances used in the experiment. As we can see, Figure 4a is not rugged, which implies that the solutions of that instance are very similar to each other. On the other hand, Figure 4b represents a rugged instance. Moreover, the instances belonging to the Hard dataset have similar fitness landscapes graphics; and it is the same with the T dataset instances. Then the Hard dataset is uniform and produces a not rugged surface, while The T dataset is not uniform and produces a rugged surface.

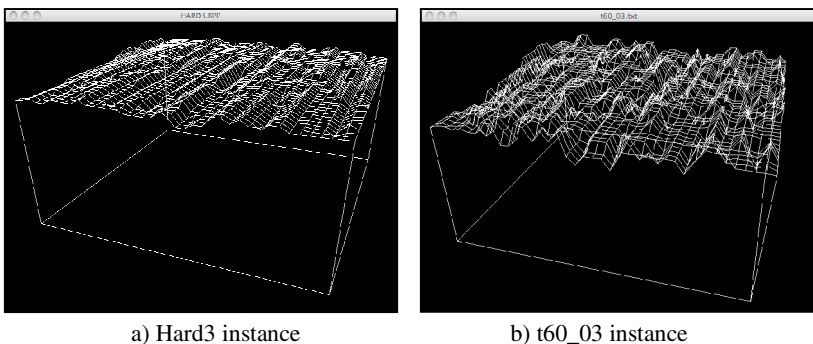
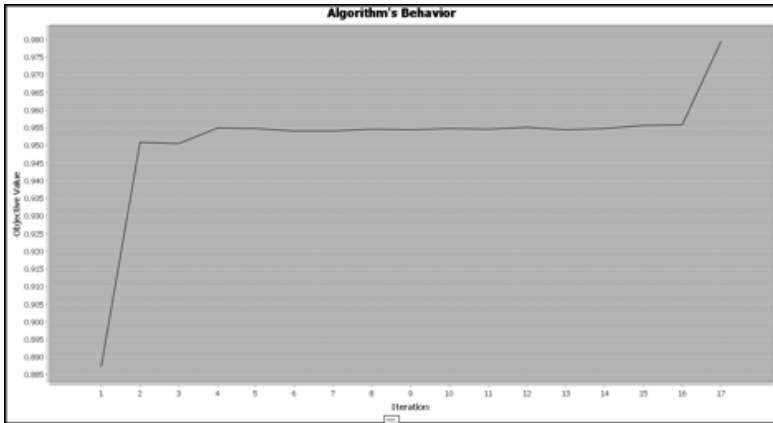
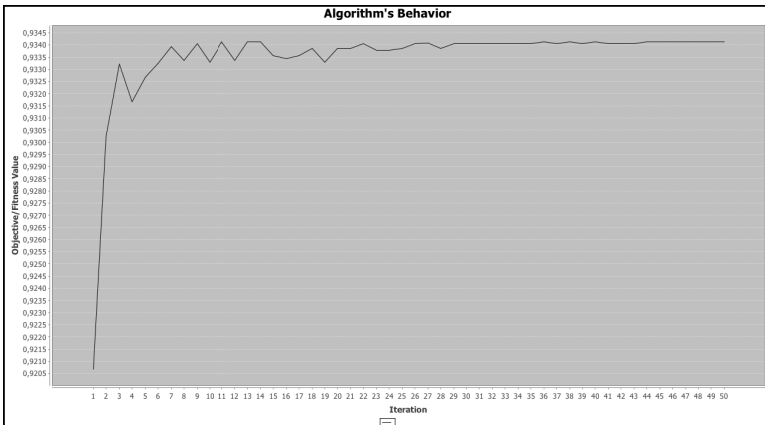


Fig. 4 Fitness landscapes visualization of two instances in VisTHAA



a) Hard3 instance



b) t60_00 instance

Fig. 5 Behavior of WABP Algorithm with two instances

3.4 Algorithm Redesign Phase

According to the visual analysis of the number of iterations without improvement, for the algorithm under study, two possible values for *nloop* parameter were considered; where *nloop* is the number of iterations of the algorithm. The first one was obtained by recognizing the following pattern: except for hard3, the remaining instances converge in early iterations. Moreover, Hard3 instance is the last to converge (17 iterations), which is why the maximum limit number of 17 iterations was set for the first reconfiguration. The second number of maximal iterations was obtained by computing the average of the number of iteration where each instance achieves its last improvement, which was 3.1 iterations; consequently, a maximum limit number of 3 iterations was used for the second reconfiguration.

4 Experimental Results

Table 2 shows that with the initial algorithm configuration, WABP finds four of ten optimum values, which add up to 381 used bins. With the first algorithm reconfiguration it can be seen that WABP achieves a considerable saving of 297 iterations, reflecting an improvement of 63% in computer time, without a loss in the quality of the solutions, that is, WABP find the same objective values for every instance. Analyzing the algorithm's performance with the second reconfiguration, it can be noted that a bigger saving of 437 iterations, which represents the 93% in computer time, is achieved. However, there is a slight loss of quality of 0.26%.

In order to analyze the obtained results and to validate statistically the forcefulness in the improvement of the algorithm performance, the Wilcoxon test was performed. VisTHAA has available this non parametric test.

Table 2 Experimental results decreasing the number of maximal iterations

<i>Instances</i>	<i>nloop=50</i>			<i>nloop=17</i>		<i>nloop=3</i>	
	<i>Optimum</i>	<i>Bins</i>	<i>Iterations</i>	<i>Bins</i>	<i>Iterations</i>	<i>Bins</i>	<i>Iterations</i>
Hard0	56	56	50	56	17	56	3
Hard1	57	57	50	57	17	57	3
Hard2	56	57	50	57	17	57	3
Hard3	55	55	17	55	17	56	3
Hard4	57	57	50	57	17	57	3
t60_00	20	21	50	21	17	21	3
t60_01	20	21	50	21	17	21	3
t60_02	20	21	50	21	17	21	3
t60_03	20	21	50	21	17	21	3
t60_04	20	21	50	21	17	21	3
TOTAL	381	387	467	387	170	388	30

Due to three experiments were conducted ($nloop=50$, $nloop=17$, $nloop=3$) and the Wilcoxon test only deals with two samples, it was necessary do the test twice, the first one with the results of the initial algorithm configuration against the first reconfiguration, and the second one with the results of the first reconfiguration against the second reconfiguration. Both, the first and the second Wilcoxon tests performed revealed that there are significant differences with a 99.5% of confidence.

In addition, Table 2 shows that the reference algorithm WABP, over the instances of the T dataset, could not find the optimum value for any. On the other hand, the instances belonging to the Hard dataset were a little easier to solve by WABP. This could be explained by the instance structure, that is, the fitness landscapes graphics of the instances belonging to the T dataset, in general, are more rugged than the other ones corresponding to the Hard dataset. According to the literature, the ruggedness of an instance is related with its difficulty, in our study, we could observe that characteristic, since the T dataset instances were the more rugged and the most difficult to solve by WABP.

5 Conclusions and Future Work

VisTHAA is a tool for analyzing the behavior of heuristic algorithms during the optimization process. Although VisTHAA is in its initial stage, includes relevant facilities for handling generic instances, statistical and visual characterization of the optimization process and statistical comparison of heuristic strategies.

By analyzing WABP algorithm, the feasibility of VisTHAA was validated, finding that after to characterize in different ways the instances and the algorithm behavior, the visual diagnosis was important to identify what kind of strategy we must follow in order to improve the algorithm performance. The graphics provide extra information than can be complemented with the attributes matrix. Specifically, analyzing the behavior graphic we realized that WABP could improve its performance by reconfiguring one of its control parameter. Two reconfigurations were performed, which were better than its predecessor in efficiency. The first adjustment allowed the significant improvement of the initial configuration of the 63%, without a loss in the quality of the solutions. The second adjustment allowed the significant improvement of the 93%, with an insignificant loss of 0.26%.

As future work, experimentation with different parameter settings is considered, as well as tests other heuristics and try to identify other factors that are crucial for the algorithm performance. Besides, it is important to identify which characteristics of the instances make them difficult to solve for WABP, once identified them, it could be possible develop improvement strategies. Preliminary, instances with high uniformity causes a not rugged search surface for WABP.

We have also considered the expansion of VisTHAA with new functionality, including on-line operation. Currently the process is off-line by requiring researches provide all the data for the optimization process. With the on-line operation, the algorithms can be run on the tool.

References

1. McGeoch, C.C.: Experimental Analysis of Algorithms. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization, vol. 2, pp. 489–513 (2000)
2. Chuin, H., Chong, W., Halim, S.: Tuning Tabu Search Strategies via Visual Diagnosis. In: MIC 2005 The 6th Metaheuristics International Conference. School of Information Systems, Singapore Management University (2005)
3. Jin, M.V.: Visualizing Swarm algorithms. Master's Thesis. Faculteit Toegepaste Wetenschappen (2004)
4. Dijkstra, E.W.: ALGOL-60 Translation. Stichting Mathematisch Centrum. 2e Boerhaavestraat 49, Amsterdam, Rakenafdeling (1961)
5. Quiroz, M.: Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP. Master's thesis. Instituto Tecnológico de Cd. Madero, México (2009)
6. Bartz-Beielstein, T., Chiarandini, M., Paquete, L. (eds.): Experimental Methods for the Analysis of Optimization Algorithms, 1st edn., 457p. Springer, Heidelberg (2010)
7. Gómez, C.G., Cruz-Reyes, L., Meza, E., Schaeffer, E., Castilla, G.: A Self-Adaptative Ant Colony System for Semantic Query Routing Problem in P2P Networks. *Computación y Sistemas* 13(4), 433–448 (2010)

8. Gendreau, M., Potvin, J.Y.: Handbook of Metaheuristics, 2nd edn. International Series in Operations Research & Management Science, vol. 146. Springer (2010) ISBN: 978-1-4419-1663-1, e-ISBN: 978-1-4419-1665-5
9. Mendenhall, W., Sincich, T.: Statistics for Engineering and the Science, 4th edn. Prentice-Hall (1997)
10. Cruz-Reyes, L.: Clasificación de Algoritmos Heurísticos para la solución de Problemas de Bin Packin. PhD thesis, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México (2004)
11. Loh, K.H., Golden, B., Wasil, E.: Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers and Operations Research* 35(2008), 2283–2291 (2006)
12. Klein, R., Scholl, A.: Bin Packing, Allowed in, <http://www.wiwi.uni-jena.de/Entscheidung/binpp/>
13. Beasley, J.E.: OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society* 41, 1069–1072, Allowed in, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html>

ADAPTIVE ANT-COLONY ALGORITHM FOR SEMANTIC QUERY ROUTING

Claudia Gómez Santillán, Laura Cruz Reyes, Elisa Schaeffer, Eustorgio Meza, Gilberto Rivera Zarate

Abstract:

The most prevalent P2P application today is file sharing, both among scientific users and the general public. A fundamental process in file sharing systems is the search mechanism. The unstructured nature of real-world large-scale complex systems poses a challenge to the search methods, because global routing and directory services are impractical to implement. This paper presents a new ant-colony algorithm, Adaptive Neighboring-Ant Search (AdaNAS), for the semantic query routing problem (SQRP) in a P2P network. The proposed algorithm incorporates an adaptive control parameter tuning technique for runtime estimation of the time-to-live (TTL) of the ants. AdaNAS uses three strategies that take advantage of the local environment: learning, characterization, and exploration. Two classical learning rules are used to gain experience on past performance using three new learning functions based on the distance traveled and the resources found by the ants. The experimental results show that the AdaNAS algorithm outperforms the NAS algorithm where the TTL value is not tuned at runtime.

Keywords: parameter tuning, search algorithm, peer-to-peer, adaptive algorithm, local environment, ant-colony algorithms.

1. Introduction

Although popular for other uses, the World Wide Web is impractical for user-to-user file sharing as it requires centralized infrastructure such as an HTTP server. In the past decade, a new class of networks called *peer-to-peer* (P2P) systems began to spread as a solution to the increasing demand of file sharing among Internet users. In P2P networks, the users interconnect to offer their files to one another [1]. The participants, called *peers*, may connect and disconnect freely, and do so constantly, which triggers frequent changes in the network structure [2].

One of the main advantages is that peers are equal in terms of functionality and tasks which are developed. This produces high fault tolerance and auto-organization: peers form unstructured networks with an acceptable connectivity and performance. The *Semantic Query Routing Problem* (SQRP) consists in deciding, based on a set of keywords, to which neighbor to forward the query to search files related with the keywords [2], [3].

The lack of global structure caused that flooding-based search mechanisms have been mainly employed. Flooding-based mechanisms are simple, but unfortunately generate vast amounts of traffic in the network and may produce congestion on Internet. Existing approaches for

SQRP in P2P networks range from simple broadcasting techniques to sophisticated methods [1], [4], [5]. The latter includes proposals based on *ant-colony systems* [6] that are specifically suited for handling routing tables in telecommunications. There exist few algorithms used for SQRP, including SemAnt [3] and Neighboring-Ant Search (NAS) [7], the latter based on the former. In this work we propose an algorithm as an extension to NAS, called the Adaptive Neighboring-Ant Search (AdaNAS). AdaNAS is hybridized with three local strategies: learning, structural characterization and exploration. These strategies are aimed to produce a greater amount of results in a lesser amount of time. The time-to-live (TTL) parameter is tuned at runtime based on the information acquired by these three local strategies.

2. Background

In order to place the research in context, this section is divided in four parts. The first part models a P2P network with graph theory, and in the second part we continue with structural characterization. The third part describes the basic ant-colony algorithms for SQRP algorithms and the last part explains parameter tuning and adaptation.

2.1. Graph Theory

A P2P network is a distributed system that can be modeled mathematically as a *graph*, $G = (V, E)$, where V is a set of *nodes* and $E \subseteq V \times V$ is a set of (symmetrical) *connections*. For more information on graph theory, we recommend the textbook by Diestel [8]. Each peer in the network is represented by a node (also called a *vertex*) of the graph. The direct communications among the peers are represented by the connections (also called the *edges*) of the graph. We denote by n the number of nodes in the system and identify the nodes by integers, $V = 1, 2, 3, \dots, n$. Two nodes that are connected are called *neighbors*; the set of all neighbors of a node i is denoted by $\Gamma(i)$. The number of neighbors of a node i is called degree and is denoted by k_i . Two nodes i and j are said to be connected if there exists at least one sequence of connections that begins at i , traverses from node to node through the connections of the graph, and ends at j . Such sequences of connections are called routes or paths and the number of connections traversed is the length of the route.

2.2. Structural Characterization using Degree Distribution

For the purpose of analyzing the structure and behavior of complex systems modeled as graphs, numerous characterization functions have been proposed [9]. There are two main types of these functions: those based on global infor-

mation that require information on the entire graph simultaneously and those based on local information that only access the information of a certain node i and its neighborhood $\Gamma(i)$ at a time.

The degree k_i of a node i is a local measure of network. $P(k)$ denotes the number of nodes that have degree k , normalized by n . The measure of $P(k)$ can be interpreted as the probability that a randomly chosen node i has degree k . The values of $P(k)$ for $k \in [0, n-1]$ (supposing that there can only be at most one connection between each pair of distinct nodes) form the *degree distribution* of the graph. Whereas the degrees themselves are local properties, obtaining the degree distribution is a global computation.

The degree distribution is widely used to classify networks according to the *generation models* that produce such distributions. Among the first and most famous generation models are the *uniform random graphs* of Erdős and Rényi [10], and Gilbert [11] that yield a binomial distribution that at the limit, approaches the Poisson distribution and most of the nodes in the graph have similar degrees [12].

In the past decade, another type of generation models became popular as various studies revealed that the degree distribution of some important real-world networks (including the WWW, the Internet, biological and social systems) was not Poisson distribution at all, but rather a power-law distribution [13], [14], [15], $P(k) \sim k^{-\gamma}$ with values of γ typically ranging between two and three. The models that produce such distributions are called *scale-free* network models. The notable structural property in networks with power law distribution is the presence of a small set of extremely well-connected nodes that are called *hubs*, whereas a great majority of the nodes has a very low degree [16], [17]. This property translates into high fault tolerance under random flaws, but high vulnerability under deliberate attack [14].

2.3. Parameter Tuning and Adaptation

Metaheuristics offer solutions that are often close to the optimum, but with a reasonable amount of resources used when compared to an exact algorithm. Unfortunately, the metaheuristics are usually rich in parameters. The choice of the values for the parameters is nontrivial and in many cases the parameters should vary during the runtime of the algorithm [18], [19].

The process of selecting the parameter values is known as tuning. The goal of offline tuning is to provide a static initial parameter configuration to be used throughout the execution of the algorithm, whereas online tuning, also known as *parameter control* or *adaptation*, is the process of adjusting the parameter values at runtime. We design a discrete model for adaptation based on the proposed by Holland in 1992 [20]. We assume that the system takes actions at discrete steps $t = 1, 2, 3, \dots$, as this assumption applies to practically all computational System. The proposed model is described in section four.

3. SQRP Search Strategies

In this section we present the problem focused in this work. First, we describe the semantic query routing problem (SQRP) as a search process. Then, strategies for solve SQRP are shown including our proposed algorithm

which uses an adaptive strategy for adjusting an important parameter for the search process: TTL.

3.1. SQRP Description

SQRP is the problem of locating information in a network based on a query formed by keywords. The goal in SQRP is to determine shorter routes from a node that issues a query to those nodes of the network that can appropriately answer the query by providing the requested information. Each query traverses the network, moving from the initiating node to a neighboring node and then to a neighbor of a neighbor and so forth, until it locates the requested resource or gives up in its absence. Due to the complexity of the problem [2], [3], [5], [21], [22], [23], solutions proposed to SQRP typically limit to special cases.

The general strategies of SQRP algorithms are the following. Each node maintains a local database of documents r_i called the *repository*. The search mechanism is based on nodes sending messages to the neighboring nodes to query the contents of their repositories. The *queries* q_i are messages that contain keywords that describe searched resource for possible matches. If this examination produces results to the query, the node responds by creating another message informing the node that launched the query of the resources available in the responding node. If there are no results or there are too few results, the node that received the query forwards it to one or more of its neighbors. This process is repeated until some predefined stopping criteria is reached. An important observation is that in a P2P network the connection pattern varies among the net (*heterogeneous topology*), moreover the connections may change in time, and this may alter the routes available for messages to take.

3.2. SQRP Algorithms

The most popular technique for searching in P2P systems is flooding, where each message is assigned a positive integer parameter known as the *time-to-live* (TTL) of the message. As the message propagates from one node to another, the value of TTL is decreased by one by each forwarding node. When TTL reaches zero, the message will be discarded and no longer propagated in the system. The main disadvantage of flooding is the rapid congestion of the communication channels [24]. Another widely used search strategy is the *random walk* [21]. A random walk in a graph is a route where the node following the initiating node is chosen uniformly at random among its neighbors.

3.2.1. AntSearch

Wu *et al.* [23] propose an algorithm called *AntSearch*. The main idea in the AntSearch algorithm is using pheromone values to identify the free-riders, prevent sending messages to those peers in order to reduce the amount of redundant messages. The estimation of a proper TTL value for a query flooding is based on the popularity of the resources. Wu *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, R : a good search algorithm should retrieve the number of results over but close to R . The second one is the *cost per result* that defines the total amount of query messages divided by the number of searched results; this metric measure

how many average query messages are generated to gain a result. Finally, search latency is defined as the total time taken by the algorithm.

3.2.2. *SemAnt*

Algorithms that incorporate information on past search performance include the SemAnt algorithm [3], [25] that uses an ant-colony system to solve SQRP in a P2P environment. SemAnt seeks to optimize the response to a certain query according to the popularity of the keywords used in the query. The algorithm takes into account network parameters such as bandwidth and latency. In SemAnt, the queries are the ants that operate in parallel and place pheromone on successful search routes. This pheromone evaporates over time to gradually eliminate old or obsolete information. Also Michlmayr [3] considers parameter tuning for the SemAnt algorithm, including manual adjustment of the TTL parameter from a set of possible values 15, 20, 25, 30, 35 and concludes that 25 is the best value for the parameter. The adjustment of the TTL is made without simultaneous tuning of the other parameters.

3.2.3. *Neighboring-Ant Search*

NAS [7] is also an ant-colony system, but incorporates a local structural measure to guide the ants towards nodes that have better connectivity. The algorithm has three main phases: an evaluation phase that examines the local repository and incorporates the classical lookahead technique [4], a transition phase in which the query propagates in the network until its TTL is reached, and a retrieval phase in which the pheromone tables are updated.

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 have a priori knowledge of where the resources are located. On the other hand, differently 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 [4], the local structural metric DDC[26] which measures the differences between the degree of a node and the degree of its neighbors, and three local functions of the past algorithm performance. This algorithm outperforms methods proposed in the literature, such as Random-Walk and SemAnt [7].

3.2.4. *Adaptive Neighboring-Ant Search*

The proposed algorithm in this work, *Adaptive Neighboring Ant Search* (AdaNAS) is largely based on the NAS algorithm, but includes the adaptation of the TTL parameter at runtime, in addition to other changes. The mechanism that may extend the TTL for an ant is called the *survival rule*. It incorporates information on past queries relying on the learning strategies included in AdaNAS, basic characteristics of SQRP and a set of parameters that are adjusted according to the results found when using the *survival*

rule itself. The rule evaluates the length of the shortest known route that begins with the connection (i, j) from the current node i to a node that contains good results for the query t . The form in which the algorithm operates is explained in detail later in Sections 4 and 5.

4. AdaNAS Model

In this section, we present a multi-agent model in order to describe the adaptive behavior of AdaNAS.

4.1. The General Model

The environment is the P2P network, in which two stimuli or inputs are observed:

- I_1 : the occurrences of the documents being searched,
- I_2 : the degree k_i of the node i .

The environment has the following order to send stimuli: observing I_1 has a higher priority than observing I_2 . AdaNAS is an ant-colony system, where each ant is modeled as an agent. AdaNAS has four agent types:

- The *query ant* is accountable for attending the users' queries and creating the *Forward Ant*; moreover it updates the pheromone table by means of evaporation. There is a *query ant* for each node in the net and it stays there while the algorithm is running.
- The *Forward Ant* uses the learning strategies for steering the query and when it finds resources creates the backward ant. It finishes the routing process when its TTL is zero or the amount of found resources is enough that is denoted by R then, it creates an *update ant*.
- The *backward ant* is responsible for informing to *query ant* the amount of resources in a node found by the *Forward Ant*. In addition, it updates the values of some learning structures that are the bases of the *survival rule* which will be explaining later (Section 4.2.3).
- The *update ant* drops pheromone on the nodes of the path generated by the *Forward Ant*. The amount of pheromone deposited depends on quantity of found resources (*hits*) and number of edges traveled (*hops*) by the *Forward Ant*.

The system is subdivided into four parts: the structures A to adapt to the environment (called *agents*), the adaptation plan P , the memory M , and the operators O . Typically, A has various alternative states A_1, A_2, A_3, \dots among which one is to be chosen for the system, according to the observations made on the environment. On the other hand, P is typically a set of rules, one or more which can be applied. These rules apply the operations in the set O . An operator is either a deterministic function, denoted as $(A_i, P_j) \rightarrow A_k$, or a stochastic function to a probability distribution over a set of states for selecting A_k . The memory M permits the system to collect information on the condition of the environment and the system itself, to use it as a base for the decision making. The observations of the environment are taken as stimuli that trigger the operators.

The routing process implemented in the *Forward Ant* is required to be adaptive, thus A is defined in function of this agent. The possible states for A are five:

- A_1 : No route has been assigned and the *Forward Ant* is at the initial node. The ant can be only activated when the *query ant* send it a query and can only receive once time each stimulus.
- A_2 : A route has been assigned and TTL has not reached zero.
- A_3 : TTL is zero.
- A_4 : *Forward Ant* used the survival rule to extend TTL.
- $A_5=X$: Terminal state is reached by the *Forward Ant*.

The Figure 1 shows the AdaNAS adaptive model. According to the stimuli -the number of documents found (dotted line, I_1) and degree of the node (solid line, I_2) - an operator is selected. The line style for state transitions follows that of the stimuli: dotted line for transitions proceeding from I_1 and solid for I_2 .

The memory M is basically composed of four structures that store information about previous queries. The first of these structures is the three dimensional pheromone table τ . The element $\tau_{ij,t}$ is the preference for moving from node i to a neighboring node j when searching by a keyword t . In this work, we assume that each query contains one keyword and the total number of keywords (or *concepts*) known to the system is denoted by C .

The pheromone table $M_1 = \tau$ is split into n bi-dimensional tables, τ_j, t , one for each node. These tables only contain the entries $\tau_{j,t}$ for a fixed node i and hence have at most dimensions $C \times |\Gamma(i)|$. The other three structures are also three-dimensional tables $M_2 = D, M_3 = N$ and $M_4 = H$, each splits into n local bi-dimensional tables in the same manner. The information in these structures is of the following kind: currently being at node i and searching for t , there is a route of distance $D_{ij,t}$ starting at the neighbor j that leads to a node identified in $N_{ij,t}$ that contains $H_{ij,t}$ hits or matching documents.

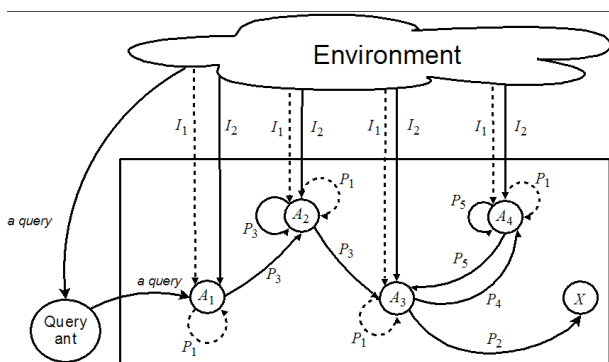


Fig. 1. AdaNAS Adaptive Model General.

The adaptive plans P are the following:

- P_1 : **The backward ant.** Created when a resource is found; modifies M_2, M_3 and M_4 .
- P_2 : **The update ant.** Modifies the pheromone table $M_1 = \tau$ when the *Forward Ant* reached 0 and *survival rule* can not to proceed.
- P_3 : **The transition rule.** Selects the next node applying the inherent learning stored in pheromone trails and in the memory structure $M_2 = D$.
- P_4 : **The survival rule.** Proceeds when the learning

stored in M_2, M_3 and M_4 permits to extend TTL and determines how much TTL must be extended.

- P_5 : **The modified transition rule.** A variation of transition rule that eliminates the pheromone and degree effects.

The operators O of AdaNAS are the following:

- $O_1: (A_1, I_1) - P_1 \rightarrow A_1$:
The ant has been just created and documents were found from the initial node (I_1), so *backward ant* updates the short-time memory (P_1) and no change in the agent state of the system is produced.
- $O_2: (A_1, I_2) - P_3 \rightarrow A_2$:
The ant has been just created and must select a neighbor node (I_2) according with the transition rule (P_3), this will produce a tracked route by the ant (A_2).
- $O_3: (A_2, I_1) - P_1 \rightarrow A_2$:
The ant has assigned a route (A_2) and documents were found from initial node (I_1), so *backward ant* updates the short-time memory (P_1) and no change in the agent state of the system is produced.
- $O_4: (A_2, I_2) - P_3 \rightarrow A_n | A_n \in \{A_2, A_3\}$:
When the ant has a partial route (A_2), it must select the next node from the neighborhood (I_2), so it applies the transition rule (P_3). The application of the transition rule can cause than TTL is over (A_3) or not (A_2).
- $O_5: (A_3, I_1) - P_1 \rightarrow A_3$:
Idem O_3 , but now the ant has $TTL = 0$ (A_3).
- $O_6: (A_3, I_2) - P_4 \rightarrow A_4$:
The ant is over TTL (A_3) and with neighborhood information (I_2), so it applies the survival rule (P_4). When P_4 is applied the ant performs its activity in an extended time to live (A_4).
- $O_7: (A_3, I_2) - P_2 \rightarrow X$:
The ant is over TTL (A_3) and with neighborhood information (I_2), so it decides that the path must end. In order to reach its final state (X), the *forward ant* must create an *updating ant* which performs the pheromone update (P_2).
- $O_8: (A_4, I_1) - P_1 \rightarrow A_4$:
Idem O_3 , but now the ant performs its activity in an extended time to live (A_4).
- $O_9: (A_4, I_2) - P_5 \rightarrow A_1 | A_1 \in \{A_3, A_4\}$:
In an extended TTL by the modified transition rule (P_5), the ant must choose a vertex from the neighborhood (I_2) like the next node in the route. The application of transition rule can cause than TTL is over (A_3) or not (A_2) again.

The general model is illustrated in Figure 1 where can be observed the transitions among states of the *Forward Ant*.

4.2. Behavior Rule

An ant-colony algorithm has rules that determine its behavior. These rules define why the ants construct and evaluate the solution and why the pheromone is updated and used. Although the pheromone is the main learning structure, AdaNAS has three more: D, N and H , for know the distances toward the nodes that contain in its repository matching documents. AdaNAS own several behavior rules: the *transition rule*, the *update rules*, the *survival*

rule and the modified *transition rule*.

4.2.1. Transition Rule

The transition rule P_3 considers two structures to determine the next state: τ and D . This transition rule is used by an ant x that is searching the keyword t and is located in the node r . The rule is formulated in the following Equation 1:

$$l(x, r, t) = \begin{cases} \operatorname{argmax}_{i \in (\Gamma(r) \setminus \Lambda x)} \{ \psi(r, i, t) \}, & \text{si } p < q \\ L(x, r, t) & \text{otherwise;} \end{cases} \quad (1)$$

where p is a pseudo-random number, q is a algorithm parameter that defines the probability of using of the exploitation technique, $\Gamma(r)$ is the set of neighbors nodes of r , Λx is the set of nodes previously visited by x , and Equation 2, defined by:

$$\psi(r, i, t) = (w_d \cdot \kappa(r, i) + w_i \cdot (D_{r,i,t})^{-1})^{\beta_1} \cdot (\tau_{r,i,t})^{\beta_2}, \quad (2)$$

where w_d is the parameter that defines the degree importance, w_i defines the distance importance toward the nearest node with matching documents ($D_{r,i,t}$), β_1 intensifies the local metrics contribution (degree and distance), β_2 intensifies pheromone contribution ($\tau_{r,i,t}$), $\kappa(r, i)$ is a normalized degree measure expressed in Equation 3:

$$\kappa(r, i) = \frac{k_i}{\max_{j \in \Gamma(r)} \{k_j\}}, \quad (3)$$

and L is the exploration technique expressed, in Equation 4:

$$L(x, r, t) = f(\{p_{x,r,i,t} \mid i \in \Gamma(r)\}), \quad (4)$$

where $f(\{p_{x,r,i,t} \mid i \in \Gamma(r)\})$ is a roulette-wheel random selection function that chooses a node i depending on its probability $p_{x,r,i,t}$ which indicates the probability of the ant x for moving from r to i searching by keyword t and it is defined in Equation 5:

$$P_{x,r,i,t} = \frac{\psi(r, i, t)}{\sum_{i \in (\Gamma(r) \setminus \Lambda x)} \psi(r, i, t)} \quad (5)$$

The tables D and τ were described in the previous section. The exploration strategy L is activated when $p \geq q$ and stimulates the ants to search for new paths. In case that $p < q$, the exploitation strategy is selected: it prefers nodes that provide a greater amount of pheromone and better connectivity with smaller numbers of hops toward a resource. As is shown in the transition rule, β_2 is the intensifier of the pheromone trail, and β_1 is the intensifier of the local metrics, this means that the algorithm will be only steered by the local metrics when $\beta_2 = 0$, or by the pheromone when $\beta_1 = 0$. In this work the initial values are $\beta_1 = 2$ and $\beta_2 = 1$.

4.2.2. Update Rules

There are two basic update rules in an ant colony algorithm: the evaporation and increment of pheromone. The evaporation method of AdaNAS is based on the technique

used in SemAnt [3], while the increment strategy is based on the proposed in NAS [7]. Both update rules are described below.

Pheromone Evaporation Rule, the pheromone evaporation is a strategy whose finality is avoid that the edges can take very big values of pheromone trail causing a greedy behavior on the algorithm. Each unit time the query ant makes smaller the pheromone trail of the node where the query ant is, by multiplying the trail by the evaporation rate ρ , which is a number between zero and one. To avoid very low values in the pheromone the rule incorporates a second term consisting of the product $\rho\tau_0$, where τ_0 is the initial pheromone value. The Equation 6 expresses mathematically the evaporation pheromone rule.

$$\tau_{r,s,t} \leftarrow (1 - \rho) \cdot \tau_{r,s,t} + \rho \cdot \tau_0 \quad (6)$$

Pheromone Increment Rule, when a *Forward Ant* finishes, it must express its performance in terms of pheromone by means of an *update ant* whose function is to increase the quantity of pheromone depending on amount of documents found and edges traversed by *Forward Ant*. This is done each time that an *update ant* passes on one node. The Equations 7 and 8 describe the *pheromone increment rule*.

$$\tau_{r,s,t} \leftarrow \tau_{r,s,t} + \Delta\tau_{r,s,t}(x) \quad (7)$$

where $\tau_{r,s,t}$ is the preference of going to s when the *Forward Ant* is in r and is searching by keyword t , $\Delta\tau_{r,s,t}(x)$ is the amount of pheromone dropped on $\tau_{r,s,t}$ by a *backward ant* generated by the *Forward Ant* x and can be expressed like:

$$\Delta\tau_{r,s,t}(x) \leftarrow \left[w_h \frac{\operatorname{hits}(x, s)}{R} + (1 - w_h) \frac{1}{\operatorname{hops}(x, r)} \right] \quad (8)$$

where $\operatorname{hits}(x, s)$ is the amount of documents found by the *Forward Ant* x from s to end of its path, and $\operatorname{hops}(x, r)$ is the length of the trajectory traversed by the *Forward Ant* x from r to the final node in its route passing by s .

4.2.3. Survival Rules

P_1 (the backward ant) updates the memory structures $M_2 = D$, $M_3 = N$, and $M_4 = H$. These structures are used in the survival rule (P_4) to increase time to live. This survival rule can be only applied when TTL is zero. The survival rule can be expressed mathematically in terms of the structures H , D and N as see in Equation 9:

$$\Delta\text{TTL}(x, i, t) = \begin{cases} D_{i, \omega(x, i, t), t}, & \text{si } \Omega(x, i, t) > Z_x \\ 0, & \text{en otro caso} \end{cases} \quad (9)$$

where $\Delta\text{TTL}(x, i, t)$ is the increment assigned to the TTL of ant x (that is, number of additional steps that the ant will be allowed to take) when searching for resources that match to t , currently being at node i . The number of additional steps $D_{i, \omega(x, i, t), t}$ for arriving in the node $\omega(x, i, t)$ is determined from the shortest paths generated by previous ants, and is taken when its associated efficiency $\Omega(x, i, t)$ is better than Z_x which is a measure of current performance

of the ant x . The auxiliary functions are shown in Equations 10 and 11:

$$\Omega(x, i, t) = \max_{j \in (\Gamma(i) \setminus \Lambda_x)} \left\{ \frac{H_{i,j,t}}{D_{i,j,t}} \mid N_{i,j,t} \notin \Lambda_x, \right. \quad (10)$$

$$\omega(x, i, t) = \arg \Omega(x, i, t), \quad (11)$$

where $\Gamma(i)$ is the set of neighbors of node i and Λ_x is the set of nodes previously visited by the ant x . The tables of hits H , of distances D , and of nodes N were explained in the previous section. The function $\omega(x, i, t)$ determines which node that is neighbor of the current node i and that has not yet been visited has previously produced the best efficiency in serving a query on t , where the efficiency is measured by $\Omega(x, i, t)$.

4.2.4. Modified Transition Rule

The *modified transition rule* is a special case of *transition rule* (see Equations 4 and 5) where $\beta_2 = 0$, $W_d = 0$ and $q = 1$. This rule is greedy and provokes the replication of paths generated by previous ants. This rule takes place when TTL has been extended canceling the normal *transition rule*. Mathematically can be express in Equations 12 and 13, like:

$$l_m(x, r, t) = \left\{ \arg \max_{i \in (\Gamma(r) \setminus \Lambda_x)} \left\{ \psi(r, i, t) \right\} \right\} \quad (12)$$

where l_m is the *modified transition rule*, r is the current node in the path, t is the searched keyword, Λ_x is the set of nodes visited by the *Forward Ant* x and

$$\psi(r, i, t) = (w_i \cdot (D_{r,i,t})^{-1})^{\beta_1} \quad (13)$$

where w_i is a parameter that defines the influence of $D_{r,i,t}$ that is the needed distance for arriving in the known nearest node with documents with keyword t , from r passing by i and β_1 is the distance intensifier.

5. AdaNAS Algorithm

AdaNAS 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. AdaNAS guides the queries toward nodes that have better connectivity using the local structural metric degree [26]; in addition, it uses the well known *lookahead* technique [25], which, by means of data structures, allows knowing the repository of the neighboring nodes of a specific node.

The AdaNAS algorithm performs in parallel all the queries using query ants. The process done by *query ant* is represented in Algorithm 1. Each node has only a query ant, which generates a *Forward Ant* x for attending only one user query, assigning the searched keyword t to the *Forward Ant*. Moreover, the *query ants* realize periodically the local pheromone evaporation of the node where it is.

In the Algorithm 2 is shown the process realized by the *Forward Ant*, as can be observed all *Forward Ants* act in parallel. In an initial phase (lines 4-8), the ant checks the local repository, and if it finds matching documents then

creates a *backward ant* y . Afterwards, it realizes the search process (lines 9-25) while it has live and has not found R documents.

The search process has three sections: Evaluation of results, evaluation and application of the extension of TTL and selection of next node (lines 24-28).

Algorithm 1: Query ant algorithm

```

1 in parallel for each query ant  $w$  located in the node  $r$ 
2 While the system is running do
3   if the user queries to find  $R$  documents with
   keyword  $t$  then
4     create Forward Ant  $x(r, t, R)$ 
5     activate  $x$ 
6   End
7   apply pheromone evaporation
8 End
9 end of in parallel

```

The first section, the evaluation of results (lines 10-15) implements the classical *Lookahead* technique. That is, the ant x located in a node r , checks the *lookahead* structure, that indicates how many matching documents are in each neighbor node of r . This function needs three parameters: the current node (r), the keyword (t) and the set of known nodes (*known*) by the ant. The set *known* indicates what nodes the *lookahead* function should ignore, because their matching documents have already taken into account. If some resource is found, the *Forward Ant* creates a *backward ant* and updates the quantity of found matching documents.

The second section (lines 16-23) is evaluation and application of the extension of TTL. In this section the ant verifies if TTL reaches zero, if it is true, the ant intends to extend its life, if it can do it, it changes the normal *transition rule* modifying some parameters (line 21) in order to create the *modified transition rule*.

The third section (lines 24-30) of the search process phase is the selection of the next node. Here, the *transition rule* (normal or modified) is applied for selecting the next node r and some structures are updated. The final phase occurs when the search process finishes; then, the *Forward Ant* creates an *update ant* z for doing the pheromone update.

The Algorithm 3 presents the parallel behavior for each *backward ant* which inversely traverses the path given by the *Forward Ant*. In each node that it visits, it tries to update the structures D , H and N , which will be used for future queries (lines 7-11). The update is realized if the new values point to a nearer node (line 7). After that, it informs to *ant query* of the initial node of the path how many documents the *Forward Ant* found and which path used (line 13).

The Algorithm 4 presents the concurrent behavior for each *update ant* which inversely traverses the path given by the *Forward Ant*. In each node that it visits, it updates the pheromone trail using the Equation 6 (line 5).

6. Experiments

In this section, we describe the experiments we carried during the comparisons of the AdaNAS and NAS algorithms.

Algorithm 2: Forward ant algorithm

```

1 in parallel for each Forward Ant  $x(r,t,R)$ 
2 initialization:  $TTL = TTL_{max}$ ,  $hops = 0$ 
3 initialization:  $path = r$ ,  $\Lambda = r$ ,  $known = r$ 
4  $Results = get\_local\_documents(r)$ 
5 if  $results > 0$  then
6   create backward ant  $y(path, results, t)$ 
7   activate  $y$ 
8 End
9 while  $TTL < 0$  and  $results < R$  do
10   $La\_results = look\_ahead(r,t,known)$ 
11  if  $la\_results > 0$  then
12    create backward ant  $y(path, la\_results, t)$ 
13    activate  $y$ 
14     $results = results + la\_results$ 
15  End
16  if  $TTL > 0$  then
17     $TTL = TTL - 1$ 
18  Else
19    if  $(results < R)$  and  $(\Delta TTL(x, results, hops) > 0)$  then
20       $TTL = TTL + \Delta TTL(x, results, hops)$ 
21      change parameters:  $q=1$ ,  $W_d=0$ ,  $\beta_2=0$ 
22    End
23  End
24  $Hops = hops + 1$ 
25  $Known = known \cup ((r \cup \Gamma(r)))$ 
26  $\Lambda = \Lambda \in r$ 
27  $r = l(x,r,t)$ 
28 add to  $path(r)$ 
29 End
30 create update ant  $z(x, path, t)$ 
31 activate  $z$ 
32 kill  $x$ 
33 end of in parallel

```

6.1. Generation of the test data

A SQRP instance is formed by three separate files: topology, repositories, and queries. We generated the experimental instances following largely those of NAS reported by Cruz *et al.* [7] in order to achieve comparable results. The structure of the environment in which is carried out the process described is called *topology*, and refers to the pattern of connections that form the nodes on the network. The generation of the topology (T) was based on the method of Barabási *et al.* [27] to create a scale-free network. We created topologies with 1024 nodes; the number of nodes was selected based on recommendations in the literature [3], [28].

The *local repository* (R) of each node was generated using “topics” obtained from ACM Computing Classification System taxonomy (ACMCCS). This database contains a total of 910 distinct topics. Also the content are scale-free: the nodes contain many documents in their repositories on the same topic (identified by keywords) and only few documents on other topics.

Algorithm 3: Backward ant algorithm

```

1 initialization:  $hops = 0$ 
2 in parallel for each backward ant  $y(path, results, t)$ 
3 for  $I = |path| - 1$  to 1 do
4    $R = path_{(i-1)}$ 

```

```

5    $s = path_i$ 
6    $hops = hops + 1$ 
7   if  $Dr, s, t > hops$  then
8      $D_{r,s,t} = hops$ 
9      $H_{r,s,t} = result$ 
10     $N_{r,s,t} = path_h$ 
11  End
12 End
13 Send  $(results, path)$  to the query ant located in  $path_l$ 
14 kill  $y$ 
15 end of in parallel

```

Algorithm 4: Update ant algorithm

```

1 in parallel for each update ant  $z(path, t, x)$ 
2 for  $i = |path| - 1$  to 1 do
3    $R = path_{(i-1)}$ 
4    $s = path_i$ 
5    $\tau_{r,s,t} = \tau_{r,s,t} + \Delta\tau_{r,s,t}(x)$ 
6 End
7 kill  $z$ 
8 end of in parallel

```

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 repository. 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 in average.

The topology and the repositories were created static, whereas the queries were launched randomly during the simulation. Each simulation was run for 15,000 queries during 500 time units, each unit has 100 ms. The average performance was studied by computing three performance measures of each 100 queries:

- **Average hops**, defined as the average amount of links traveled by a Forward Ant until its death, that is, reaching either the maximum amount of results required R or running out of TTL.
- **Average hits**, defined as the average number of resources found by each Forward Ant until its death.
- **Average efficiency**, defined as the average of resources found per traversed edge (hits/hops).

6.2. Parameters

The configuration of the algorithms used in the experimentation is shown in Tables 1 and 2. The first column is the parameter, the second column is the parameter value and the third column is a description of the parameter. These parameter values were based on recommendations of the literature [3], [6], [7], [29], [30].

6.3. Results

The goal of the experiments was to examine the effect of the strategies incorporated in the AdaNAS algorithm and determine whether there is a significant contribution to the average efficiency. The main objective of SQRP is to find a set of paths among the nodes launching the que-

Table 1. Parameter configuration of the NAS algorithm.

PARAMETER	VALUE	DEFINITION
α	0.07	Global pheromone evaporation factor
ρ	0.07	Local pheromone evaporation factor
β	2	Intensifier of pheromone trail
τ_0	0.009	Pheromone table initialization
q_0	0.9	Relative importance between exploration and exploitation
R	10	Maximum number of results to retrieve
TTL_{max}	10	Initial TTL of the Forward Ants
W	0.5	Relative importance of the resources found and TTL

Table 2. Parameter configuration of the AdaNAS algorithm.

PARAMETER	VALUE	DEFINITION
ρ	00.07	Local pheromone evaporation factor
β_1	2	Intensification of local measurements (degree and distance) in transition rule.
β_2	1	Intensification of pheromone trail in the in the transition rule.
T_0	0.009	Pheromone table initialization
q_0	0.9	Relative importance between exploration and Exploitation in the transition rule.
R	10	Maximum number of results to retrieve
TTL_{max}	10	Initial TTL of the Forward Ants
w_h	0.5	Relative importance of the hits and hops in the increment rule
w_d	1	Degree's influence in the transition rule
w_i	1	Distance's influence in the transition rule

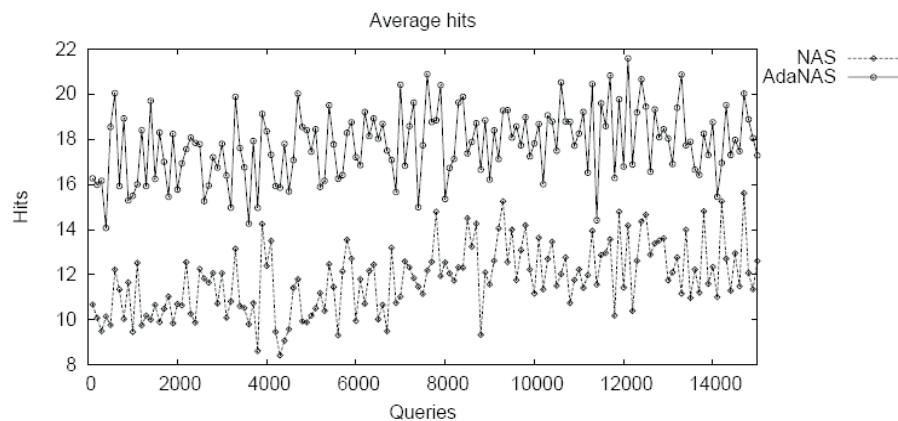


Fig. 2. Learning evolution in terms of the number of resources found for AdaNAS and NAS algorithms.

ries and the nodes containing the resources, such that the efficiency is greater, this is, the quantity of found resources is maximized and the quantity of steps given to find the resources is minimized.

Figure 2 shows the *average hits* performed during 15,000 queries with AdaNAS and NAS algorithms on an example instance. NAS starts off approximately at 13.4 hits per query; at the end, the average hit increases to 14.7 hits per query. For AdaNAS the average hit starts at 16 and after 15,000 queries the average hit ends at 18.3. On the other hand, Figure 3 shows the *average hops* performed during a set of queries with NAS and AdaNAS. NAS starts approximately at 17.4 hops per query; at the end, the average hops decrease to 15.7 hops per query. For AdaNAS the average hops starts at 13.7 and after 15,000 queries the average hops ends at 9.1. Finally, Figure 4 shows the *average efficiency* performed during a set of queries. NAS starts approximately at 0.76 hits per hop; at the end, it increases to 0.93 hits per hop. For AdaNAS the average efficiency starts at 1.17 hits per hop and after 15,000 que-

ries the average efficiency ends at 2.

The adaptive strategies of AdaNAS show an increment of 24.5% of found documents, but the biggest contribution is a reduction of hops in 40%, giving efficiency approximately twice better on the final performance of NAS. This observation suggests that the use of degree instead of DDC was profitable. In addition, the incorporation of the survival rule permits to improve the efficiency, because it guides the Forward Ants to nodes that can satisfy the query. Moreover, in future works it will be important to study adaptive strategies for other parameters as well as the initial algorithm parameter configuration in search of further improvement in the efficiency.

Figure 5 shows the results of the different experiments applied to NAS and AdaNAS on thirty runnings for each ninety different instances generated with the characteristics described in Section 6.1. It can be seen from it that on all the instances the AdaNAS algorithm outperforms NAS. On average, AdaNAS had efficiency 81% better than NAS.

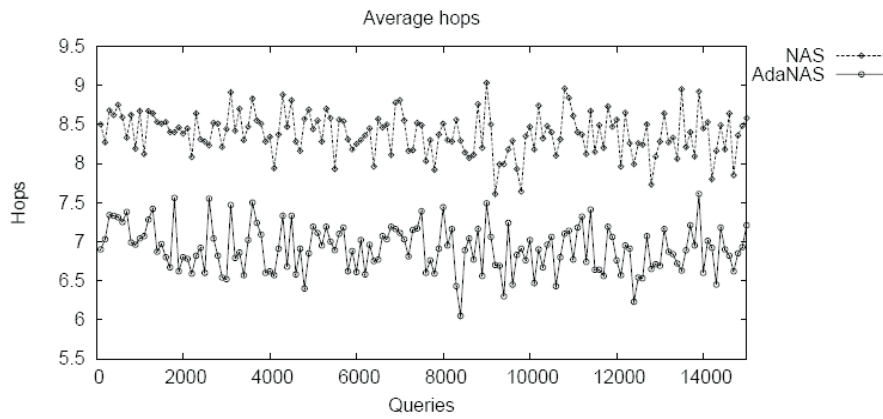


Fig. 3. Learning evolution in terms of the length of the route taken for AdaNAS and NAS algorithms.

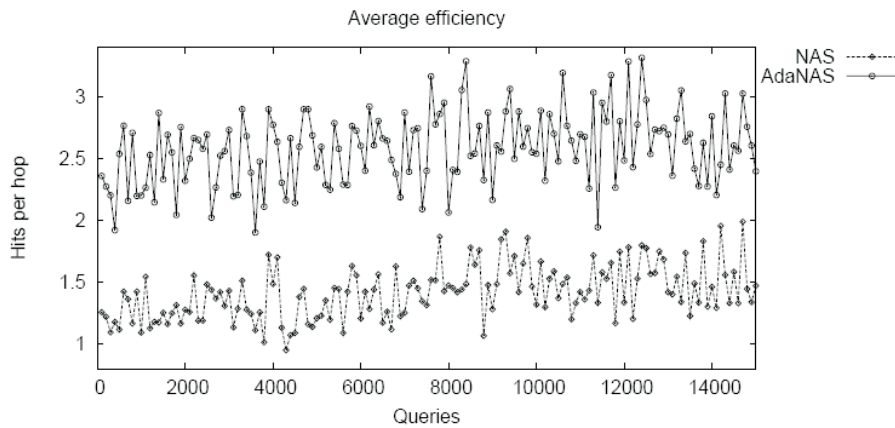


Fig. 4. Learning evolution in terms of the efficiency (hits/ hop) for AdaNAS and NAS algorithms.

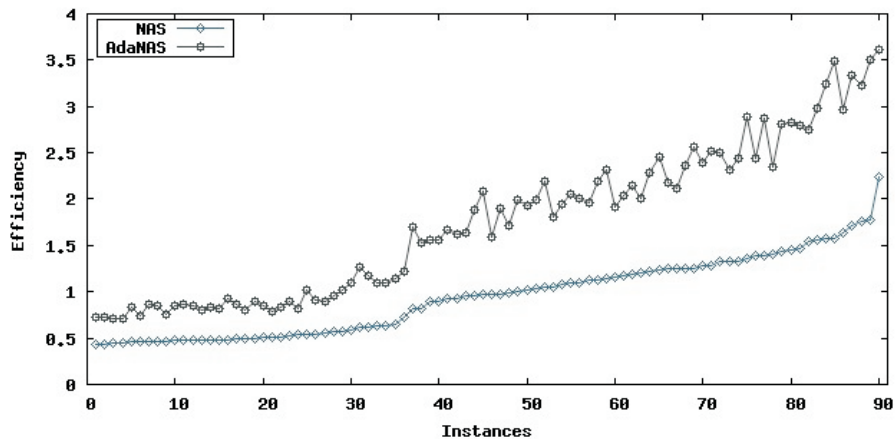


Fig. 5. Comparison between NAS and AdaNAS experimenting with 90 instances.

7. Conclusions

For the solution of SQRP, we proposed a novel algorithm called AdaNAS that is based on existing ant-colony algorithms, which is a state of art algorithm. AdaNAS algorithm incorporates parameters adaptive control techniques to estimate a proper TTL value for dynamic text query routing.

In addition, it incorporates local strategies that take advantage of the environment on local level; three functions were used to learn from past performance. This combination resulted in a lower hop count and an improved hit count, outperforming the NAS algorithm. Our experiments confirmed that the proposed techniques are more effective at improving search efficiency. Specifically the

AdaNAS algorithm in the efficiency showed an improvement of the 81% in the performance efficiency over the NAS algorithm.

As future work, we plan to study more profoundly the relation among SQRP characteristics, the configuration of the 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.

ACKNOWLEDGMENTS

This research was supported in part by CONACYT, DGEST and IPN.

AUTHORS

Claudia Gómez Santillán - Instituto Tecnológico de Ciudad Madero (ITCM). 1ro. de Mayo y Sor Juana I. de la Cruz s/n CP. 89440, Tamaulipas, México. Tel.: (52) 833 3574820 Ext. 3024 and Instituto Politécnico Nacional, Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada (IPN-CICATA). Carretera Tampico-Puerto Industrial Alt., Km.14.5. Altamira, Tamps., México. Tel.: 018332600124. E-mail: cgggs71@hotmail.com.

Laura Cruz Reyes*, **Gilberto Rivera Zarate** - Instituto Tecnológico de Ciudad Madero (ITCM). 1ro. de Mayo y Sor Juana I. de la Cruz s/n CP. 89440, Tamaulipas, México. Tel.: (52) 833 3574820 Ext. 3024. E-mails: lcruzreyes@prodigy.net.mx and riveragil@gmail.com.

Eustorgio Meza, Elisa Schaeffer - CIIDIT & FIME, Universidad Autónoma de Nuevo León (UANL), Av. Universidad s/n, Cd. Universitaria, CP.66450, San Nicolás de los Garza, N.L., México. Phone: (52)8113404000 Ext.1509. E-mails: elisa@yalma.fime.uanl.mx, emezac@ipn.mx.

* Corresponding author

References

- [1] Sakarayan G., *A Content-Oriented Approach to Topology Evolution and Search in Peer-to-Peer Systems*. PhD thesis, University of Rostock, 2004.
- [2] Wu L.-S., Akavipat R., Menczer F., "Adaptive query routing in peer Web search". In: *Proc. 14th International World Wide Web Conference*, 2005, pp. 1074-1075.
- [3] Michlmayr E., *Ant Algorithms for Self-Organization in Social Networks*. PhD thesis, Vienna University of Technology, 2007.
- [4] Mihail M., Saberi A., Tetali P., "Random walks with look ahead in power law random graphs", *Internet Mathematics*, no. 3, 2004.
- [5] Tempich C., Staab S., Wranik A., "REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphers", in: *13th World Wide Web Conference (WWW)*, 2004.
- [6] Dorigo M., Gambardella L.M. "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, 1997, pp. 53-66.
- [7] Cruz L., Gómez C., Aguirre M., Schaeffer S., Turrubiates T., Ortega R., Fraire H., NAS algorithm for semantic query routing systems in complex networks. In: *DCAI, Advances in Soft Computing*, vol. 50, 2008, pp. 284-292.
- [8] Diestel R., "Graph Theory", *Graduate Texts in Mathematics*, vol. 173, Springer-Verlag: New York, USA, 2000.
- [9] Costa L., Rodríguez F.A., Travieso G., Villas P.R., "Characterization of complex networks: A survey of measurements", *Advances in Physics*, no. 56, 2007, pp. 167-242.
- [10] Erdos P., Rényi A., *On the evolution of random graphs*, vol. 2, Akademiai Kiadó, Budapest, Hungary, 1976. First publication in MTA Mat. Kut. Int. Kozl. 1960, pp. 482-525.
- [11] Gilbert E., "Random graphs", *Annals of Mathematical Statistics*, 30(4), Dec. 1959, pp. 1141-1144.
- [12] Bollobás B., *Random Graphs, Cambridge Studies in Advanced Mathematics*, vol. 73, Cambridge University Press, Cambridge, UK, 2nd edition, 2001.
- [13] Adamic L., Huberman B., "Power-law distribution of the World Wide Web". *Science*, no. 287(5461), 2000, p. 2115.
- [14] Albert R., Jeong H., Barabási A., "Error and attack tolerance of complex networks". *Nature*, no. 506, 2000, pp. 378-382.
- [15] Faloutsos M., Faloutsos P., Faloutsos C., "On power-law relationship on the internet topology". *ACM SIGCOMM Computer Communication Review*, 1999, pp. 251-262.
- [16] Barabási A., *Emergence of scaling in complex networks*, Wiley VHC, 2003, pp. 69-82.
- [17] Newman M. E. J., "Power laws, pareto distributions and zipf's law", *Contemporary Physics*, vol. 46(5), 2005, pp. 323-351.
- [18] Birattari M., *The Problem of Tuning Metaheuristics as Seen From a Machine Learning Perspective*. PhD thesis, Bruxelles University, 2004.
- [19] Glover F., Laguna M., *Tabú Search*. Kluwer Academic Publishers, 1986.
- [20] Holland J.H., *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992.
- [21] Amaral L., Ottino J., "Complex systems and networks: Challenges and opportunities for chemical and biological engineers". *Chemical Engineering Scientist*, no. 59 2004, pp. 1653-1666.
- [22] Liu L., Xiao Long J., Kwock C.C., "Autonomy oriented computing from problem solving to complex system modeling". In: *Springer Science + Business Media Inc*, 2005, pp. 27-54,
- [23] Wu C.-J., Yang K.-H., Ho. J.M., "AntSearch: An ant search algorithm in unstructured peer-to-peer networks". In: *ISCC*, 2006, pp. 429-434.
- [24] Goldberg P., Papadimitriou C., "Reducibility among equilibrium problems". In: *Proceedings of the 38th annual ACM symposium on Theory of Computing*, New York, NY, USA, 2005, pp. 61-70.
- [25] Michlmayr E., Pany A., Kappel G., "Using Taxonomies for Content-based Routing with Ants". In: *Proceedings of the Workshop on Innovations in Web Infrastructure, 15th International World Wide Web Conference (WWW2006)*, May 2006.
- [26] Ortega R., *Estudio de las Propiedades Topológicas en Redes Complejas con Diferente Distribución del Grado y su Aplicación en la Búsqueda de Recursos Distribuidos*. PhD thesis, Instituto Politécnico Nacional, México, 2009. (in Spanish)
- [27] Barabási A., Albert R., Jeong H., "Mean-field theory for scale-free random networks". *Physical Review Letters*, no. 272, 1999, pp. 173-189.
- [28] Babaoglu O., Meling H., Montresor A., "Anthill: An framework for the development of agent-based peer to peer systems". In: *22nd International Conference On Distributed Computing Systems*. ACM, 2002.
- [29] Ridge E., *Design of Experiments for the Tuning of Optimization Algorithms*. PhD thesis, University of York, 2007.
- [30] Ridge E., Kudenko D., "Tuning the Performance of the MMAS Heuristic in Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics". *Lecture Notes in Computer Science*, vol. 4638, T. Stutzle, M. Birattari, Eds. Berlin / Heidelberg: Springer, 2007, ISBN 978-3-540-74445-0, pp. 46-60.

Find out how to access preview-only content

Book inside Get Access
Hybrid Artificial Intelligent Systems

Lecture Notes in Computer Science Volume 6679, 2011, pp 26-33

Enhancing Accuracy of Hybrid Packing Systems through General-Purpose Characterization

Citations

Abstract

Some Hybrid Packing Systems integrate several algorithms to solve the bin packing problem (BPP) based on their past performance and the problem characterization. These systems relate BPP characteristics with the performance of the set of solution algorithms and allow us to estimate which algorithm is to yield the best performance for a previously unseen instance. The present paper focuses on the characterization of NP-hard problems. In related work, characterization metrics are traditionally oriented towards problem structure. In this work, we propose metrics based on descriptive statistics for the Bin Packing Problem (BPP). The proposed metrics are of general purpose, meaning that the metrics do not depend on problem structure and can be applied to BPP and other problems to complement existent metrics. The “enhanced” Hybrid Packing System outperforms the version that does not take advantage of the general-purpose metrics; the results obtained show a 3%-improvement with respect to the reference Packing System.

Page %P

Page 1

Enhancing Accuracy of Hybrid Packing Systems through General-Purpose Characterization

Laura Cruz-Reyes¹, Claudia Gómez-Santillán¹, Satu Elisa Schaeffer²,
Marcela Quiroz-Castellanos¹, Victor M. Alvarez-Hernández¹,
and Verónica Pérez-Rosas¹

¹ Instituto Tecnológico de Ciudad Madero, ITCM

² Facultad de Ingeniería Mecánica y Eléctrica, UANL

lcruzreyes@prodigy.net.mx, cggs71@hotmail.com,
elisa@yalma.fime.uanl.mx, qc.marcela@gmail.com,
{mantorvicuel,x_filesrecords}@hotmail.com

Abstract. Some Hybrid Packing Systems integrate several algorithms to solve

Abstract. Some Hybrid Packing Systems integrate several algorithms to solve the bin packing problem (BPP) based on their past performance and the problem characterization. These systems relate BPP characteristics with the performance of the set of solution algorithms and allow us to estimate which algorithm is to yield the best performance for a previously unseen instance. The present paper focuses on the characterization of NP-hard problems. In related work, characterization metrics are traditionally oriented towards problem structure. In this work, we propose metrics based on descriptive statistics for the Bin Packing Problem (BPP). The proposed metrics are of general purpose, meaning that the metrics do not depend on problem structure and can be applied to BPP and other problems to complement existent metrics. The “enhanced” Hybrid Packing System outperforms the version that does not take advantage of the general-purpose metrics; the results obtained show a 3%-improvement with respect to the reference Packing System.

Keywords: Hybrid Solution Systems, Bin Packing Problem, Problem Characterization, Heuristic Algorithms, Algorithm Selection.

1 Introduction

The benefits and shortcomings of *approximate solution algorithms* for complex optimization problems are widely studied. However, the lack of formal methods to characterize the performance of such algorithms makes it difficult to evaluate and select among them [1, 2, 3, 4].

In this work we seek to characterize NP-hard optimization problems in order to select the best approximation algorithm for the solution of a given problem instance; the algorithm portfolio is part of a hybrid solution system. As a practical example, we attend the *BPP*. The proposed characterization process is based on *statistical techniques* and *machine learning*, and forms part of a methodology oriented to the construction of prediction models of algorithmic performance [5]. A prediction model is constructed based on a learning set of problem instances and is then applied to infer the best-performing algorithm for new, previously unseen problem instances.

E. Corchado, M. Kurzyński, M. Woźniak (Eds.): HAIS 2011, Part II, LNAI 6679, pp. 26–33, 2011.
© Springer-Verlag Berlin Heidelberg 2011

No Body Text -- translate me!

Emilio Corchado
Marek Kuciński
Michał Woźniak (Eds.)

LNAI 6679

Hybrid Artificial Intelligent Systems

18th International Conference, HAIS 2011
Wrocław, Poland, May 2011
Proceedings, Part II

2
Part II

 Springer



1. Papadimitriou, C., Steiglitz, K.: *Combinatorial Optimization: Algorithms and Complexity*. Mineola. Dover Publications, New York (1998)
2. Reeves, C.R.: *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, New York (1993)
3. Smith-Miles, K., James, R., Giffin, J., Tu, Y.: Understanding the relationship between scheduling problem structure and heuristic performance using knowledge discovery. In: *Learning and Intelligent Optimization, LION*, vol. 3 (2009)
4. Messelis, T., Haspeslagh, S., Bilgin, B., De Causmaecker, P., Vanden Berghe, G.: Towards prediction of algorithm performance in real world optimisation problems. In: *Proceedings of the 21st Benelux Conference on Artificial Intelligence, BNAIC*, Eindhoven pp. 177–183 (2009)
5. Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Romero, D., Cruz, L.: A Statistical Approach for Algorithm Selection. In: Ribeiro, C.C., Martins, S.L. (eds.) *WEA 2004. LNCS*, vol. 3059, pp. 417–431. Springer, Heidelberg (2004) CrossRef
6. Borghetti, B.J.: *Inference algorithm Performance and Selection under Constrained Resources*. M.Sc. Thesis. Faculty of the School of Engineering Air University Force Institute of Technology Air University (1996)
7. Yuan, B.: *Towards Improved Experimental Evaluation and Comparison of Evolutionary Algorithms*. Ph.D. Thesis, School of Information Technology and Electrical Engineering of The University of Queensland, Australia (2006)
8. Gagliolo, M., Schmidhuber, J.: Learning Dynamic Algorithm Portfolios. In: *AI & MATH 2006, Special Issue of the Annals of Mathematics and Artificial Intelligence*, vol. 47, pp. 295–328. Springer, Heidelberg (2007)
9. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the Empirical Hardness of Optimization Problems: the case of combinatorial auctions. In: Van Hentenryck, P. (ed.) *CP 2002. LNCS*, vol. 2470, pp. 556–572. Springer, Heidelberg (2002) CrossRef
10. Coffman, E.G., Courboubetis, C., Garey, M.R., Johnson, D.S., Shor, P.W., Weber, R.R.: Perfect Packing Theorems and the Average Case Behavior of Optimal and Online Bin Packing. *ACM-SIAM Review. Society for Industrial and Applied Mathematics* 44, 95–108 (2002)
11. Coffman, J., Garey, M., Jonson, D.: Approximation Algorithms for Bin-Packing, a Survey. In: *Approximation Algorithms for NP-hard Problems*, pp. 46–93. PWS, Boston (1997)
12. Coffman, J.E., Galambos, G., Martello, S., Vigo, D.: *Bin Packing Approximation Algorithms: Combinatorial*

Analysis. In: Du, D.-Z., Pardalos, P.M. (eds.) *Handbook of Combinatorial Optimization*, pp. 151–207. Kluwer Academic Publishers, Dordrecht (1999)

13. Pérez, J., Pazos, R.A., Vélez, L., Rodríguez, G.: Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. In: Coello, C.A., Albornoz, A., Sucar, L.E., Cairó, O.B. (eds.) *LNCS*, vol. 4128, pp. 119–127. Springer, Heidelberg (2002)
14. Ducatelle, F., Levine, J.: Ant Colony Optimisation for Bin Packing and Cutting Stock Problems. In: *Proceedings of the UK Workshop on Computational Intelligence*, Edinburgh (2001)
15. Ross, S.M.: *Simulación*. Segunda edición. Prentice Hall, EEUU (1999)
16. Weisstein, E.W.: *CRC Concise Encyclopedia of Mathematics*, 2nd edn. CRC Press, FL (2002) CrossRef
17. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning Multi-label Scene Classification. *Pattern Recognition* 37, 1757–1771 (2004) CrossRef
18. Wiczorkowska, A., Synak, P., Ras, Z.: Multi-label classification of emotions in music. In: *Proceedings of IIS 2006 Symposium, Intelligent Information Processing and Web Mining, Advances in Soft Computing*, pp. 307–315. Springer, Heidelberg (2006)
19. Alsabti, K., Ranka, S., Singh, V.: An Efficient K-Means Clustering Algorithm. In: *IPPS/SPDP Workshop on High Performance Data Mining*, Orlando, Florida (1998)
20. Beasley, J.E.: *Bin Packing - one dimensional*. Brunel University. The Beasley's OR-Library, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html>
21. Scholl, A., Klein, R.: *Bin Packing*, Jena University, <http://www.wiwi.uni-jena.de/Entscheidung/binpp/metric.htm>
22. Quinlan, J.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo; *Book Review*, vol. 16(3), pp. 235–240. Springer, Heidelberg (1993)
23. Dallas, J.: *Métodos Multivariados Aplicados al Análisis de Datos*. In: Thomson (ed.) *Internacional*, México (2000)

About this Chapter

Title

Enhancing Accuracy of Hybrid Packing Systems through General-Purpose Characterization

Book Title

Hybrid Artificial Intelligent Systems

Book Subtitle

6th International Conference, HAIS 2011, Wroclaw, Poland, May 23-25, 2011, Proceedings, Part II

Pages

pp 26-33

Copyright

2011

DOI

10.1007/978-3-642-21222-2_4

Print ISBN

978-3-642-21221-5

Online ISBN

978-3-642-21222-2

Series Title

Lecture Notes in Computer Science

Series Volume

6679

Series ISSN

0302-9743

Publisher

Springer Berlin Heidelberg

Copyright Holder

Springer Berlin Heidelberg

Additional Links

- [About this Book](#)

Topics

- [Artificial Intelligence \(incl. Robotics\)](#)
- [Algorithm Analysis and Problem Complexity](#)
- [Information Systems Applications \(incl. Internet\)](#)
- [Computation by Abstract Devices](#)
- [Database Management](#)
- [Information Storage and Retrieval](#)

Keywords

- [Hybrid Solution Systems](#)
- [Bin Packing Problem](#)
- [Problem Characterization](#)
- [Heuristic Algorithms](#)
- [Algorithm Selection](#)




Industry Sectors

- [Electronics](#)
- [Telecommunications](#)
- [IT & Software](#)

eBook Packages

- eBook Package english Computer Science
- eBook Package english full Collection

Editors

- Emilio Corchado  ⁽¹⁹⁾
- Marek Kurzyński  ⁽²⁰⁾
- Michał Woźniak  ⁽²⁰⁾

Editor Affiliations

- 19. GICAP Research Group, University of Burgos
- 20. Wroclaw University of Technology

Authors

- Laura Cruz-Reyes ⁽²¹⁾
- Claudia Gómez-Santillán ⁽²¹⁾
- Satu Elisa Schaeffer ⁽²²⁾
- Marcela Quiroz-Castellanos ⁽²¹⁾
- Victor M. Alvarez-Hernández ⁽²¹⁾
- Verónica Pérez-Rosas ⁽²¹⁾

Author Affiliations

- 21. Instituto Tecnológico de Ciudad Madero, ITCM, Mexico
- 22. Facultad de Ingeniería Mecánica y Eléctrica, UANL, Mexico

Continue reading...

To view the rest of this content please follow the download PDF link above.

Get Read. Publish
with Springer.

Global Reach

High Quality

Fast
Publication

**PLUS Author
Tools &
Services**



Hyperheuristic for the Parameter Tuning of a Bio-Inspired Algorithm of Query Routing in P2P Networks

Paula Hernández¹, Claudia Gómez¹, Laura Cruz¹, Alberto Ochoa²,
Norberto Castillo¹ and Gilberto Rivera¹

¹División de Estudios de Posgrado e Investigación,
Instituto Tecnológico de Ciudad Madero. Juventino Rosas y Jesús Urueta s/n,
Col. Los mangos, C.P. 89440, Cd. Madero, Tamaulipas, México
{paulahdz314, cggs71}@hotmail.com, lcruzreyes@prodigy.net.mx,
{norberto_castillo15, grivera984}@hotmail.com

²Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez. Henry
Dunant 4016, Zona Pronaf, C.P. 32310, Cd. Juárez, Chihuahua, México
doctor_albertoochoa@hotmail.com

Abstract. The computational optimization field defines the parameter tuning problem as the correct selection of the parameter values in order to stabilize the behavior of the algorithms. This paper deals the parameters tuning in dynamic and large-scale conditions for an algorithm that solves the *Semantic Query Routing Problem* (SQRP) in *peer-to-peer* networks. In order to solve SQRP, the HH_AdaNAS algorithm is proposed, which is an ant colony algorithm that deals synchronously with two processes. The first process consists in generating a SQRP solution. The second one, on the other hand, has the goal to adjust the *Time To Live* parameter of each ant, through a hyperheuristic. HH_AdaNAS performs adaptive control through the hyperheuristic considering SQRP local conditions. The experimental results show that HH_AdaNAS, incorporating the techniques of parameters tuning with hyperheuristics, increases its performance by 2.42% compared with the algorithms to solve SQRP found in literature.

Keywords: Parameter Tuning, Hyperheuristic, SQRP.

1 Introduction

Currently, the use of evolutionary computation has become very popular as a tool to provide solutions to various real-world problems. However, different tools proposed in the evolutionary field require careful adjustment of its parameters, which is usually done empirically, and is also different for each problem to be solved. It should be mentioned that specialized adjustment leads to an increase in the development cost.

The parameter tuning problem has received a lot of attention, because the efficiency of the algorithms is significantly affected by the assigned value to its parameters.

There are few papers which deal the parameter tuning in dynamic and large-scale conditions, such as the *Semantic Query Routing Problem* (SQRP) in *peer-to-peer* (P2P).

SQRP is a complex problem that has characteristics that are challenging for search algorithms. Due to its difficulty this problem has been partially developed under different perspectives [1][2][3]. The works mentioned above, use as solution technique ant colony algorithms. In these algorithms the TTL parameter, which indicates the maximum allowed time for each query in the network, begins with a static value and is decreased gradually by a fixed rule. More recent works such as Rivera [4] and Gomez [5] have focused on using adaptive techniques for adjusting this parameter considered significant [6]. In this work, when the TTL runs out, the algorithm uses an adaptive strategy to decide whether or not to extend the time to live.

In this paper, the main motivation was to create an algorithm called HH_AdaNAS with adaptive techniques through hyperheuristic strategies. The adaptation is performed throughout the search process. This feature makes the difference with such works, because the hyperheuristic defines itself and during its execution, the appropriate TTL values.

So HH_AdaNAS is an ant colony algorithm that deals synchronously with two processes. The first process consists in generating a SQRP solution. The second one, on the other hand, has the goal to adjust the *Time To Live* parameter of each ant, through of the hyperheuristic proposed.

Moreover, after a literature search, we found that SQRP has not been dealt with hyperheuristic techniques, these techniques have been used in other application domains, some of them are: Packing [7] and Vehicle Routing Problem [8]. It should be mentioned that few researchers have tackled the adaptation of parameters in hyperheuristics [9][10].

2 Background

This section describes the information related to research. First hyperheuristic term is defined, after the parameter tuning, the semantic query routing and P2P networks are described.

2.1 Hyperheuristic

A hyperheuristic is a high-level algorithm that acts as a planner on a set of heuristics that makes the programming in a deterministic or nondeterministic form [11]. The most appropriate heuristic is determined and is automatically applied by the hyperheuristic technique at each step to solve a given problem [12].

2.2 Parameter Tuning

Each one of the combinations of parameter values is called *parametric configuration*, and the problem of selecting appropriate values for the parameters to regulate the behavior of algorithms is called parameter tuning [13][14].

The classification proposed by Michalewicz & Fogel [15] divides the parameter tuning in two stages depending on what part of the experiment is applied. If applied before the execution of the experiment it is called *parameter control*.

The parameter control is divided into deterministic, adaptive and self-adaptive control. *Adaptive control*, which is performed in this work, is done when there is some form of feedback from the past that determines a change in direction and magnitude of the parameter.

2.3 Routing of Semantic Consultation and Peer to Peer Nets

The problem of searching for textual information through keywords on Internet is known as *Semantic Query Routing* (SQRP). Its objective is to determine the shortest path from a node that issues a query to the location of the nodes that can answer it appropriately providing the required information. Complex systems such as SQRP involve elements such as the environment (topology), entities that interact in the system (nodes, repositories and queries) and an objective function (minimizing steps and maximizing results) [2][16]. This problem has been taking a great relevance with the growth of the peer-to-peer communities.

Peer to peer systems are defined as distributed systems consisting of interconnected nodes that have equal role and responsibility. These systems are characterized by decentralized control, scalability and extreme dynamism of their operating environment [17][18]. Some examples include academic P2P networks, such as LionShare [19] and military networks, such as DARPA [20].

3 Description of HH_AdaNAS

This section presents the architecture of the system, data structures, the description of the proposed algorithm HH_AdaNAS and the description of hyperheuristic HH_TTL implemented.

3.1 Architecture of HH_AdaNAS

HH_AdaNAS is adaptive metaheuristic algorithm, based on AdaNAS [4], but incorporates a hyperheuristic called HH_TTL; it adapts the parameter of time to live during the execution of the algorithm. HH_AdaNAS uses as solution algorithm an Ant Colony.

This algorithm has two objectives: it seeks to maximize the number of resources found by the ants and to minimize the number of steps that the ants take it. The general architecture of the multi-agents system HH_AdaNAS is shown in Figure 1, and comprises two main elements:

1. Environment E , which is a static P2P complex network.
2. Agents $\{w, x, y, z, x_{hh}, z_{hh}\}$. HH_AdaNAS has six types of agents, each of which have a specific role. They are represented as ants of the algorithm HH_AdaNAS proposed, these ants modify the environment and the hyperheuristic ants x_{hh} and z_{hh} adapts the TTL parameter. The function of each agent is described in Section 3.2.

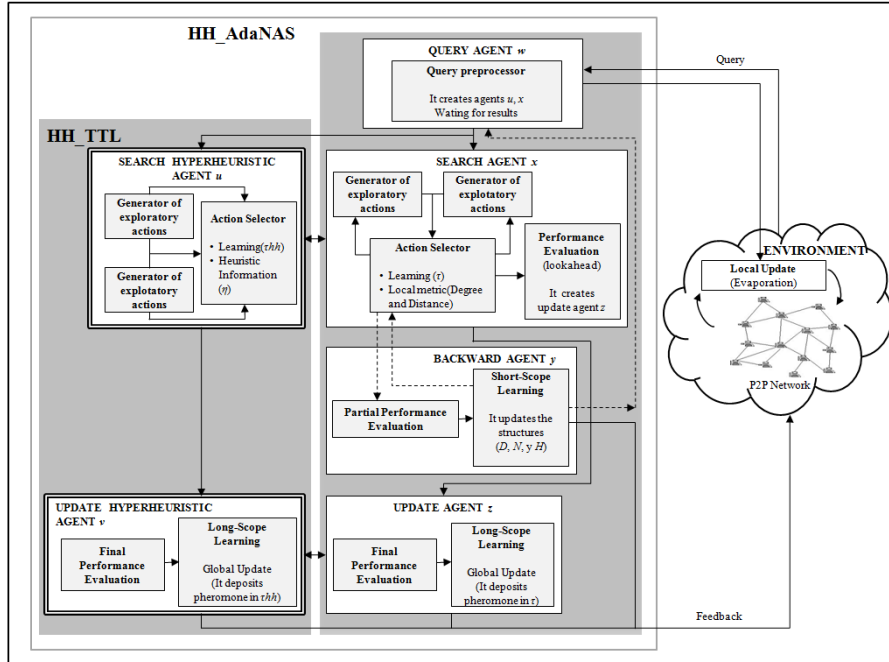


Fig. 1. General Architecture of HH_AdaNAS

3.2 Data Structures of HH_AdaNAS

The proposed algorithm HH_AdaNAS consists of six data structures, in which are stored heuristic information or gained experience in the past. The relationship of these structures is shown in Figure 2.

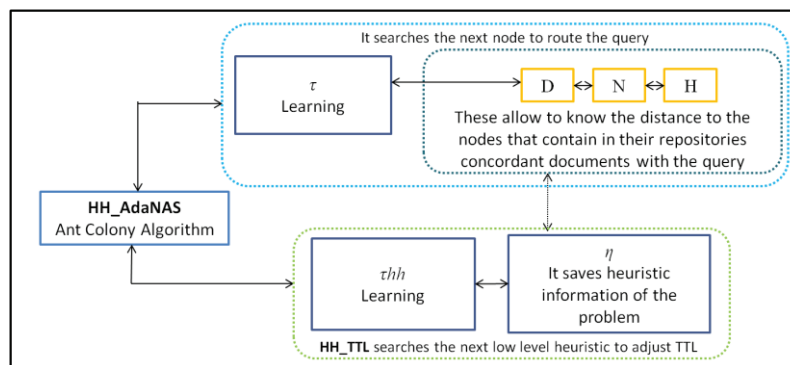


Fig. 2. Data structures of HH_AdaNAS

When HH_AdaNAS searches for the next node, in the routing process of the query, is based on the pheromone table τ and tables D , N y H [21]. Also, when HH_TTL chooses the following low level heuristic through Equation 1 is based on the following tables:

1. The *pheromone* table τhh is divided into n two-dimensional tables, corresponding one τhh_i for each node i in the network. Each $\tau hh_{i,j,l}$ in turn contains a two-dimensional table $l m_l \times l n_l$, where m is the number of visibility states of the problem and n is the total number of heuristics; an example of this can be seen in Figure 3a.
2. The table of *visibility states* η is of size $l m_l \times l n_l$ and is shown in the Figure 3b. The values of the table η were assigned according to knowledge of the problem and they are static.

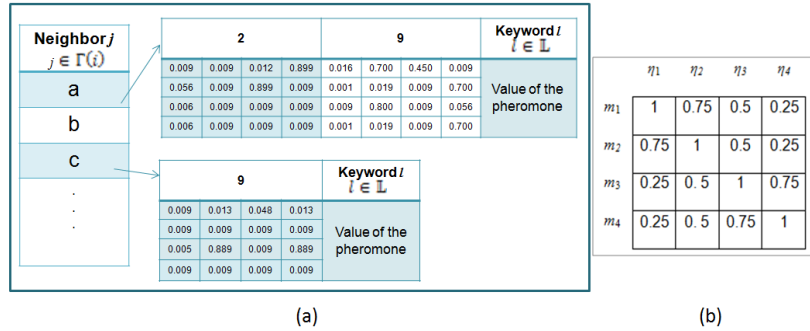


Fig. 3. Data structures of the hyperheuristic HH_TTL. a) Pheromone table τhh and b) Table of the visibility states η .

3.3 Algorithmic Description of HH_AdaNAS

In parallel all the queries in the HH_AdaNAS use *Query Ants* w . Each ant w generates a *Forward Ant* x (It generates a solution for SQRP) and *Hyperheuristic Forward Ant* u (It adjusts adaptively the TTL parameter), besides, this ant updates the pheromone tables τ and τhh through the evaporation.

Algorithm 2 shows the routing process, which is performed by the *Forward Ant* x and *Hyperheuristic Forward Ant* u , these ants work synchronously (see Figure 1). All the ants work in parallel.

In the beginning, ant u has a time to live of TTL_{inic} . The operation of the algorithm can be divided into three phases. In an initial phase (lines 4-8), the ant x checks the local repository of the issuing node of the query and, if documents are consistent, creates a *Backward Ant* y , the algorithm followed by the *Backward Ant* y is found in Gomez et al. [21]. The *Backward Ant* y informs to *Query Ant* w the amount of found resources on a node by the *Forward Ant* x and updates the values of some learning structures (D , N and H).

Subsequently, the next phase is the search process (lines 9-22), which is performed until the time to live runs out and are not R consistent documents. R is the number of documents required by users.

During the search process results are evaluated (lines 10-15) [3], next node is selected (lines 16-18 and 20) [4] and the time to live parameter is adjust by proposed hyperheuristic HH_TTL (lines 19 and 21).

HH_TTL, through *Hyperheuristic Forward Ant* u selects the low level heuristic that best adapts TTL, this by Equation 1 (Line 19). *Sequence_TTL* structure is the sequence of heuristics that make adapting the TTL parameter, this structure is updated in line 21.

In the final phase of the algorithm HH_AdaNAS (lines 23-28) the *Forward Ant* x creates *Update Ant* z and evaluates the solution generated for SQRP, the rule is described in Gomez et al. [21]. Also *Hyperheuristic Forward Ant* u creates *Hyperheuristic Update Ant* v and the last one deposits the pheromone on the path traveled by the ant u (line 24), that is, the sequence of low level heuristics selected for the adaptation of TTL. The deposit rule for the table τ_{hh} is shown in Equation 6.

Algorithm 2. HH_AdaNAS Algorithm that show the routing process with hyperheuristic

```

1  Process in parallel for each Forward Ant  $x$  ( $r$ ,  $l$ ) and each Hyperheuristic
   Forward Ant  $u$  ( $m$ ,  $n$ )
2    Initialization:  $path \leftarrow \langle r \rangle$ ,  $\Lambda \leftarrow \{r\}$ ,  $known \leftarrow \{r\}$ 
3    Initialization:  $TTL = TTL_{init}$ ,  $sequence\_TTL \leftarrow \langle n \rangle$ 
4     $results \leftarrow$  get local documents( $r$ )
5    If  $results > 0$  then
6      Create Backward Ant  $y$  ( $path$ ,  $results$ ,  $l$ )
7      Activate  $y$ 
8    End
9    While  $TTL > 0$  and  $results < R$  do
10      $la\_results \leftarrow$  lookahead( $r$ ,  $l$ ,  $known$ )
11     If  $la\_results > 0$  then
12       Create Backward Ant  $y$  ( $path$ ,  $results$ ,  $l$ )
13       Activate  $y$ 
14        $results \leftarrow results + la\_results$ 
15     End
16      $known \leftarrow known \cup \Gamma(r)$ 
17      $\Lambda \leftarrow \Lambda \cup r$ 
18     Apply transition rule:  $r \leftarrow \ell(x, r, l)$ 
19     Apply Adaptation_TTL rule:  $n \leftarrow \phi(u, r, s, l, m)$ 
20     add_to_path( $r$ )
21     add_to_sequece_TTL( $n$ )
22   End
23   Create Update Ant  $z$  ( $x$ ,  $path$ ,  $l$ )
24   Create Hyperheuristic Update Ant  $v$  ( $u$ ,  $path$ ,  $sequence\_TTL$ ,  $l$ )
25   Activate  $z$ 
26   Activate  $v$ 
27   Kill  $x$ 
28   Kill  $u$ 
29 End of the Process in parallel

```

3.4 Description of HH_TTL

The hyperheuristic, which adapts the time to live (*Hyperheuristic Time To Live*, HH_TTL), was designed with online learning [12], and uses an Ant Colony metaheuristic as high level heuristic.

As shown in Figure 4, the low level heuristics are related with SQRP. It also notes that there is a barrier between the hyperheuristic and the set of low level heuristics; this allows the hyperheuristic to be independent of the problem domain. In this context, hyperheuristic would ask how each of the low-level heuristics would work, so it can decide which heuristic to apply at each time to adapt the TTL parameter, according to the current state of the system, in this case, of performance achieved.

The design of the hyperheuristic was done so that while the solution is built for SQRP, low-level heuristics adapt the TTL parameter, this working synchronously.

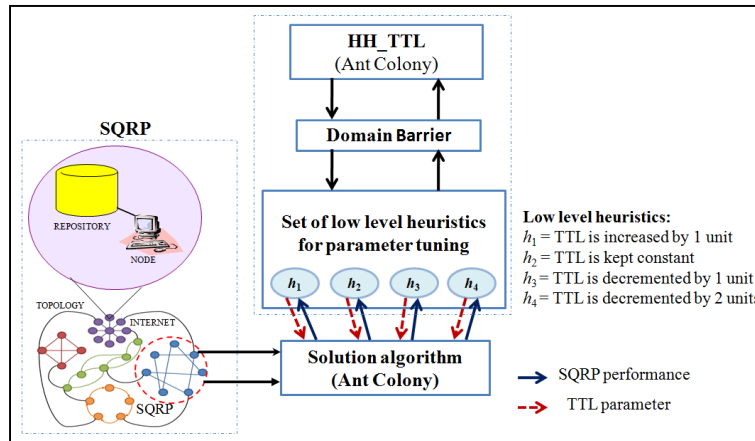


Fig. 4. General diagram of HH_TTL

3.5 Rules of Behavior of HH_TTL

The hyperheuristic HH_TTL has two rules of behavior, which interact with data structures: the selection rule and update rules.

1. Selection Rule of the Heuristics

In this stage the *Hyperheuristic Forward Ant* u , selects the low level heuristic to adapt TTL. This movement is realized following a selection rule that uses local information, which includes heuristic information η and learning (table τ_{hh}) to guide the search.

First HH_TTL determines the state m of SQRP, in which is the *Hyperheuristic Forward Ant* u , after that selects the best low level heuristic n that adapts the TTL parameter.

The selection rule for *Hyperheuristic Forward Ant* u , which is consulted through keyword l , located at node r and it decided to route the query to node s , with the visibility state m is the following:

$$\phi(u, r, s, l, m) = \begin{cases} \arg \max_{n \in H} \{\delta(r, s, l, m, n)\}, & \text{si } \varphi \leq q \\ \gamma(u, r, s, l, m), & \text{otherwise} \end{cases} \quad (1)$$

where $\phi(u, r, s, l, m)$ is the function that selects the next low level heuristic, φ is a number pseudorandom, q is an algorithm parameter which defines the probability of using the exploitation or exploration technique, φ and q acquires values between zero and one. H is the set of low level heuristics of the visibility state m and the Equation 2 shows the exploration technique,

$$\delta(r, s, l, m, n) = (\eta_{m,n})^{\beta_1} (\tau h h_{r,s,l,m,n})^{\beta_2} \quad (2)$$

where β_1 is a parameter that intensify the contribution of the visibility ($\eta_{m,n}$) and β_2 intensify the contribution of the pheromone ($\tau h h_{r,s,l,m,n}$). The table η has heuristic information of the problem and the pheromone table $\tau h h$ saves the gained experience in the past.

In the Equation 1, γ is exploration technique, which selects the next low level heuristic. This technique is expressed as:

$$\gamma(u, r, s, l, m) = f(\{p_{u,r,s,l,m,n} | n \in H\}) \quad (3)$$

where $f(\{p_{u,r,s,l,m,n} | n \in H\})$ is the roulette-wheel random selection function which selects low level heuristic n depending on its $p_{u,r,s,l,m,n}$, which indicates the probability that the *Hyperheuristic Forward Ant* u , which is the visibility state m , selects the heuristic n as the following in the adaptation of TTL. It can define $p_{u,r,s,l,m,n}$ as:

$$p_{u,r,s,l,m,n} = \frac{\delta(r, s, l, m, n)}{\sum_{n \in H} \delta(r, s, l, m, n)} \quad (4)$$

2. Update Rules of the hyperheuristic

The proposed hyperheuristic HH_TTL applies deposit and evaporation rules on its pheromone table $\tau h h$.

Evaporation Rule of the Pheromone

When choosing a low level heuristic, the proposed hyperheuristic algorithm implements a local update on the table $\tau h h$, in each unit of time (typically 100 ms), which is the following:

$$\begin{aligned} \tau h h_{r,s,l,m,n} &= (1 - \rho) \tau h h_{r,s,l,m,n} + \rho \tau_0 \\ \forall (r, s, l, m, n) &\in \{r\} \times \Gamma(i) \times \mathbb{L} \times \mathbb{S} \times H, \end{aligned} \quad (5)$$

where r is the current node, s is the selected node to route the query by the keyword l , m is the current visibility state, n is the select heuristic, ρ is the evaporation rate of pheromone (number between zero and one) and τ_0 is the initial value of pheromone. \mathbb{L} is the dictionary for the queries, \mathbb{S} is the set of the visibility state, H is the set of low level heuristics and $\{r\} \times \Gamma(i) \times \mathbb{L}$ is the Cartesian product between sets $\{r\}$, $\Gamma(i)$, \mathbb{L} , \mathbb{S} and H .

Deposit Rule of the Pheromone

Once each *Hyperheuristic Forward Ant* u has generated a solution, it is evaluated and an amount of pheromone is deposited, that is based on the quality of its solution. This process is realized by a *Hyperheuristic Update Ant* v .

When the *Hyperheuristic Update Ant* v is created runs in reverse the route generated by the *Hyperheuristic Forward Ant* u , whenever it reaches a different heuristic modifies the pheromone trail according to the formula:

$$\tau h_{r,s,l,m,n} = \tau h_{r,s,l,m,n} + \Delta \tau h_{r,s,l,m,n}(u) \quad (6)$$

In the Equation 6, $\tau h_{r,s,l,m,n}$ is the preference of selecting the low level heuristic n , in the state m , for *Hyperheuristic Forward Ant* u located in the node r , which has selected the node s to route the query by l . $\Delta \tau h_{r,s,l,m,n}(u)$ is the amount of pheromone deposited by *Hyperheuristic Update Ant* v and

$$\Delta \tau h_{r,s,l,m,n}(u) = (w_h) \frac{hits(x,s)}{R} + (1 - w_h) \frac{1}{hops(x,r)} \quad (7)$$

where R is the amount of required resources, w_h is an parameter that represents the goodness of the path and takes a value between zero and one, $hits(x,s)$ is the amount of found resources by the *Forward Ant* x from node s until the end of its route, $hops(x,r)$ is the length of the generated route by *Forward Ant* x from node r to the end of its route.

4 Experimental Results

This section presents the performance of the algorithm and is compared with an algorithm of the literature in the area. It also describes the experimental setup and test instances used.

4.1 Experimental Environment

The following configuration corresponds to the experimental conditions that are common to the test described.

Software: Operative system Microsoft Windows 7 Home Premium; Java programming language, Java Platform, JDK 1.6; and integrated development, Eclipse 3.4.

Hardware: Computer equipment with processor Intel (R) Core (TM) i5 CPU M430 2.27 GHz and RAM memory of 4 GB.

Instances: It has 90 different SQRP instances; each of them consists of three files that represent the topology, queries and repositories. The description of the features can be found in Cruz et al. [6].

Initial Configuration of HH_AdaNAS

Table 1 shows the assignment of values for each HH_AdaNAS parameter. The parameter values were based on values suggested of the literature as Dorigo [22], Michlmayr [2], Aguirre [3] and Rivera [4].

Table 1. Values for the parameters of HH_AdaNAS

Parameter	Description	Value
τ_0	Pheromone table initialization	0.009
D_0	Distance table initialization	999
ρ	Local pheromone evaporation factor	0.35
β_1	Intensification of local measurements (degree and distance)	2.0
β_2	Intensification of pheromone trail	1.0
q	Relative importance between exploration and Exploitation	0.65
W_h	Relative importance of the hits and hops in the increment rule	0.5
W_{deg}	Degree's influence in the selection the next node	2.0
W_{dist}	Distance's influence in the selection the next node	1.0
TTL_{inic}	Initial Time To Live of the Forward Ants	10

4.2 Performance Measurement of HH_AdaNAS

In this section we show experimentally that our HH_AdaNAS algorithm outperforms the AdaNAS algorithm. Also HH_AdaNAS outperforms NAS, SemAnt and random walk algorithms, inasmuch as in Gomez et al. [21] and Rivera [4] reported that AdaNAS surpasses the NAS performance. Also Gomez et al. [16] reported that NAS outperforms SemAnt and random walk algorithms [3], so our algorithm is positioned as the best of them.

In this experiment, in the HH_AdaNAS and AdaNAS algorithms, the performance achieved by the *Forward Ant* x , which is the agent performing the query, is measured by the rate of found documents by traversed edge. The larger number of found documents by edge that runs the *Forward Ant* x , the better algorithm's performance will have.

To measure the performance of the entire ant colony, the average performance of 100 queries is calculated. The average performance of the latest 100 ants is called *final efficiency of the algorithm*; this measure was used to compare the HH_AdaNAS algorithm with the AdaNAS algorithm.

Each algorithm was run thirty times per instance with the configuration described in Table 1. Figure 5 shows a comparison chart between the resulting performance of the HH_AdaNAS algorithm and the reference algorithm AdaNAS, for each ninety different test instances. It is observed that the HH_AdaNAS algorithm outperforms AdaNAS algorithm. This is because the HH_AdaNAS algorithm achieved an average performance of 2.34 resources by edge resources, while the average performance reached achieved by AdaNAS algorithm was of 2.28 resources by edge. That is, HH_AdaNAS using hyperheuristic techniques had an improvement of 2.42% in average efficiency over AdaNAS. It is because the hyperheuristic HH_TTL in HH_AdaNAS defines itself and during its execution, the appropriate TTL values; and on the other hand, AdaNAS defines the TTL values in a partial and deterministic way.

Additionally, to validate the performance results of these two algorithms, non-parametric statistical test of Wilcoxon was performed [23]. The results of this test reveals that the performance of the algorithm HH_AdaNAS shows a significant improvement over the algorithm AdaNAS, on the set of the 90 test instances, at a confidence level above 95%.

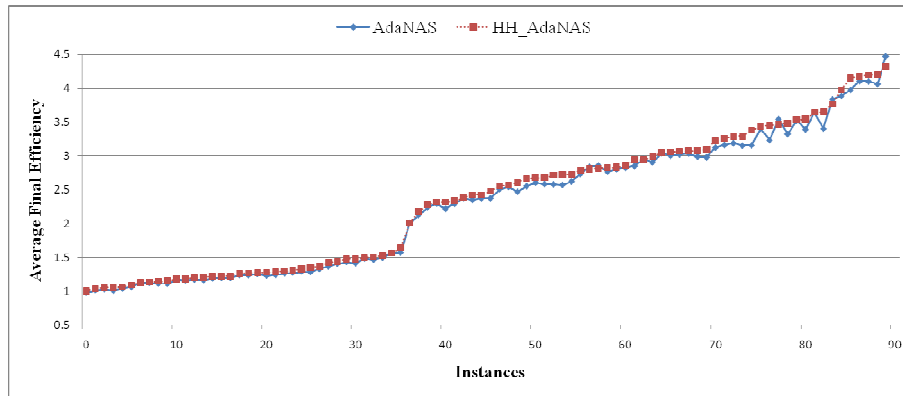


Fig. 5. Comparison of performance between the algorithms HH_AdaNAS and AdaNAS

5 Conclusions

In this work the semantic query routing process was optimized by creating a hyperheuristic algorithm whose main characteristic was its adaptability to the environment. The HH_AdaNAS algorithm was able to integrate the routing process that AdaNAS algorithm performs and the HH_TTL hyperheuristic, which adapts the TTL parameter.

The HH_AdaNAS algorithm has better average performance than his predecessor AdaNAS in 2.42%, taking into account the final efficiency of the algorithms. In the process of adaptation hyperheuristic agents (hyperheuristic ants) do not depend entirely on TTL_{inic} parameter, but is able to determine the necessary time to live while the query is routed to the nodes that satisfy it.

The main difference in the adaptation of the TTL parameter between the algorithms AdaNAS and HH_AdaNAS is that the first one does it in a partial and deterministic form, while the second one does it through the learning acquired during the solution algorithmic process.

References

1. Yang, K., Wu, C., Ho, J.: AntSearch: An ant search algorithm in unstructured peer-to-peer networks. *IEICE Transactions on Communications* 89(9), 2300–2308 (2006)
2. Michlmayr, E.: Ant Algorithms for Self-Organization in Social Networks. PhD thesis, Women's Postgraduate College for Internet Technologies, WIT (2007)
3. Aguirre, M.: Algoritmo de Búsqueda Semántica para Redes P2P Complejas. Master's thesis, División de Estudio de Posgrado e Investigación (2008)
4. Rivera, G.: Ajuste Adaptativo de un Algoritmo de Enrutamiento de Consultas Semánticas en Redes P2P. Master's thesis, División de Estudio de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero (2009)
5. Gómez, C.: Afinación Estática Global de Redes Complejas y Control Dinámico Local de la Función de Tiempo de Vida en el Problema de Direccionamiento de Consultas Semánticas. PhD thesis, Instituto Politécnico Nacional, Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada, Unidad Altamira (2009)

6. Cruz, L., Gómez, C., Aguirre, M., Schaeffer, S., Turrubiates, T., Ortega, R., Fraire, H.: NAS algorithm for semantic query routing systems in complex networks. In: DCAI. Advances in Soft Computing, vol. 50, pp. 284–292. Springer, Heidelberg (2008)
7. Garrido, P., Riff, M.-C.: Collaboration Between Hyperheuristics to Solve Strip-Packing Problems. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 698–707. Springer, Heidelberg (2007)
8. Garrido, P., Castro, C.: Stable Solving of CVRPs Using Hyperheuristics. In: GECCO 2009, Montréal, Québec, Canada, July 8-12 (2009)
9. Han, L., Kendall, G.: Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. In: Congress on Evolutionary Computation, Canberra, Australia, pp. 2230–2237 (2003)
10. Cowling, P., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
11. Özcan, E., Bilgin, B., Korkmaz, E.: A Comprehensive Analysis of Hyper-heuristics. Journal Intelligent Data Analysis. Computer & Communication Sciences 12(1), 3–23 (2008)
12. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: Exploring Hyper-Heuristic Methodologies With Genetic Programming. In: Mumford, C.L., Jain, L.C. (eds.) Computational Intelligence. ISRL, vol. 1, pp. 177–201. Springer, Heidelberg (2009)
13. Eiben, A., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
14. Birattari, M.: The Problem of Tuning Metaheuristics as seen from a machine learning perspective. PhD thesis, Universidad libre de Bruxelles (2004)
15. Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. segunda edición. Springer, Heidelberg (2004)
16. Gómez, C.G., Cruz, L., Meza, E., Schaeffer, E., Castilla, G.: A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks. Computación y Sistemas 13(4), 433–448 (2010) ISSN 1405-5546
17. Montresor, A., Meling, H., Babaoglu, Ö.: Towards Adaptive, Resilient and Self-organizing Peer-to-Peer Systems. In: Gregori, E., Cherkasova, L., Cugola, G., Panzieri, F., Picco, G.P. (eds.) NETWORKING 2002. LNCS, vol. 2376, pp. 300–305. Springer, Heidelberg (2002)
18. Ardenghi, J., Echaiz, J., Cenci, K., Chuburu, M., Friedrich, G., García, R., Gutierrez, L., De Matteis, L., Caballero, J.P.: Características de Grids vs. Sistemas Peer-to-Peer y su posible Conjunción. In: IX Workshop de Investigadores en Ciencias de la Computación (WICC 2007), pp. 587–590 (2007) ISBN 978-950-763-075-0
19. Halm M., LionShare: Secure P2P Collaboration for Academic Networks. In: EDUCAUSE Annual Conference (2006)
20. Defense Advanced Research Project Agency (2008), <http://www.darpa.mil>
21. Santillán, C.G., Reyes, L.C., Schaeffer, E., Meza, E., Zarate, G.R.: Local Survival Rule for Steer an Adaptive Ant-Colony Algorithm in Complex Systems. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Recognition Based on Biometrics. SCI, vol. 312, pp. 245–265. Springer, Heidelberg (2010)
22. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
23. García, S., Molina, D., Lozano, F., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 Special Session on Real Parameter Optimization. Journal of Heuristics (2008)

Impact of Initial Tuning for Algorithm That Solve Query Routing

Claudia Gómez Santillán, Laura Cruz Reyes, Gilberto Rivera Zarate,
Juan González Barbosa, and Marcela Quiroz Castellanos

Instituto Tecnológico de Ciudad Madero, México
{cggs71, lauracruzreyes, grivera984,
jjgonzalezbarbosa}@hotmail.com

Abstract. The algorithms are the most common form of problem solving in many science fields. Algorithms include parameters that need to be tuned with the objective of optimizing its processes. This work uses Hoeffding race techniques, with the objective to obtain the best initial combination of variables to use it as an input configuration. Hoeffding race quickly discard less promising candidates as soon as there are evidences enough to remove them from the competition. These evidences are based on the use of any statistical test that, at a given confidence level, would set a range of expected performance for configuration. All the experiment was applied in AdaNAS (Adaptive Neighboring-Ant Search), an algorithm that was developed to route queries through the Internet. Our results show that there is a significant gain in efficiency of the AdaNAS algorithm by using the simple, but powerful, technique of initial setting of parameters presented in this paper. In our experiments, the average efficiency was improved 50% by using a good initial configuration.

Keywords: Algorithms Optimization, Parameter Setting, Race Techniques, Ant Algorithms, Query Routing.

1 Introduction

All businesses need to share resources and collaboratively work, for these reasons their information exchange systems need adjust to the changing needs for resource management, this has impacted on turning them into complex systems in their structure, organization, distribution and access. Hence there is a necessity for creating algorithms that help to users find the information that they request within a reasonable processing time and with a higher quality of the obtained information.

New communication models have emerged in the Internet that manage information in a distributed manner and offer significant advantages for the business. Examples of such systems are peer-to-peer (P2P) networks that consist of a set of computers interconnected to offer its resources to other peers within the network.

The P2P systems together with the underlying communication network (Internet) form a complex system that requires autonomous operation through mechanisms of intelligent navigation [8]. To achieve this it is necessary to set appropriate values to its parameters.

Find the initial setting of parameters for an optimization algorithm is a non-trivial task, and it can consume, according to Adenso [1], to 90% of development time for solving a problem. Whenever we want to do a setting of parameters, we have two questions to answer: 1) How many runs of the algorithm will be needed? 2) How many instances should I run? It is commonly carried out 30 runs of the algorithm on the entire set of test instances. However, could be fewer runs or fewer instances?

In such scenarios is desirable to use a method to select a configuration of parameters without having to do all runs over all test instances but to ensure, with a certain level of confidence, that the chosen configuration is reliably the best.

In this paper we present a method based on statistical tests that complies with such features, called Hoeffding Race [2][3]. Hoeffding Race is a long studied technique, that even has been used in previous works [4][5] for parameter setting, however in recent years has lacked attention and its benefits have been denied in the area of automatic optimization. In this paper we present the use of Hoeffding Race applying it to an algorithm called AdaNAS.

AdaNAS [5][6] is based on the ACS (Ant Colony System) metaheuristic and NAS[7] algorithm, hybridized with local strategies such as: learning, characterization, and exploration. All the strategies were developed to resolve for the semantic query routing problem (SQRP) in peer to peer networks (P2P).

SQRP consists to discover routes as short as possible between a node that issues a query by the user, and a node (or several) that has the resources to satisfy that request. AdaNAS, as many other algorithms, has a lot of parameters that need to be properly tuned in order to yield a fully functioning algorithm. [1][4][8].

We describe the impact of initial tuning for AdaNAS algorithm by finding a good initial configuration through experimental evaluations statistically guided that reduce the number of experiments. In addition, the initial configuration helps to obtain a better efficiency from the beginning to the end of the execution of the algorithm and thus achieve the goal of SQRP, which is to maximize the amount of resources found in the P2P network and minimize the distance between the query node and the node with matching resources.

2 Parameter Setting

Determining the best combination of variables to use as input for a common practical problem is hard, since the input values have to be chosen in a way that the cost function is optimized. Parameter setting can be classified into: parameter tuning and parameter control. Michalewicz [9] and Angeline [10] define them as: Parameter Control is the setting done during the algorithm execution. This type of parameter setting supervises the local environmental changes and the current state of the algorithm to adapt locally the configuration to the local conditions; and Parameter tuning is the setting done before the algorithm runs, and provides a

global initial configuration. It evaluates the general performance of the algorithm but, it does not assure that the values for the parameters will be the best in each instant of the run of the algorithm.

The parameter tuning can be applied through three techniques: a) by hand, doing a sequence of experiments with different values of the parameters, and choosing the configuration with the best performance, b) Meta-Evolution, using an auxiliary metaheuristic algorithm to improve the performance of the main metaheuristic algorithm and c) Design of Experiment (DOE), this technique provides a great variety of statistics tests to make useful decisions [11][12]. DOE is a statistics tool set, useful on making plans, running and interpreting an experiment, while searching for valid and impartial deductions. An experiment can be defined, as a planned test which introduces checked changes in the process or system variables, with the aim of analyzing changes that could happen over the system outputs. Race is an algorithm family whose aim is to select from among a set of models, one that is considered best based on established criteria. When applied to the problem of parameter setting, racing algorithms can select from a set of configurations, the one whose associated cost is minimized by solving a set of instances [2][3][4].

2.1 Hoeffding Race

The Hoeffding race is a technique for finding out a good model for data by quickly discarding bad models and concentrating the computational effort at differentiating between the better ones [2][3][4].

The first stage of the method is to establish the necessary elements, such as: N points with which to test a given model, E_{true} is a real average error if we were to test a model on N points. But if you only want to test a model on n points ($n < N$), then you only have an estimate E_{est} of the average true error E_{true} . Hoeffding's bounds are useful when n points are tested with an identical independent distribution from the set of N original test point. In this case, you can say that the probability of E_{est} being more than ϵ away from E_{true} is: $\Pr(|E_{true} - E_{est}| > \epsilon) < 2e^{(-2n \epsilon^2)/B^2}$, where B bounds the greatest possible error that a model can make. Now, to calculate the parameter ϵ , which tell us how close the estimated mean is to the true mean after n points with confidence $1-\delta$, the Equation 1 is used.

$$\epsilon(n) = \sqrt{\frac{B^2 \log(2/\delta)}{2n}} \quad (1)$$

The next stage of the process assumes the dataset has N data points and, for each iteration of the algorithm, a point from test set is randomly selected. Each data point is a box whose boundaries are $E_{est} + \epsilon$ (upper bound) and $E_{est} - \epsilon$ (lower bound), and the center is the average of the n points, E_{est} .

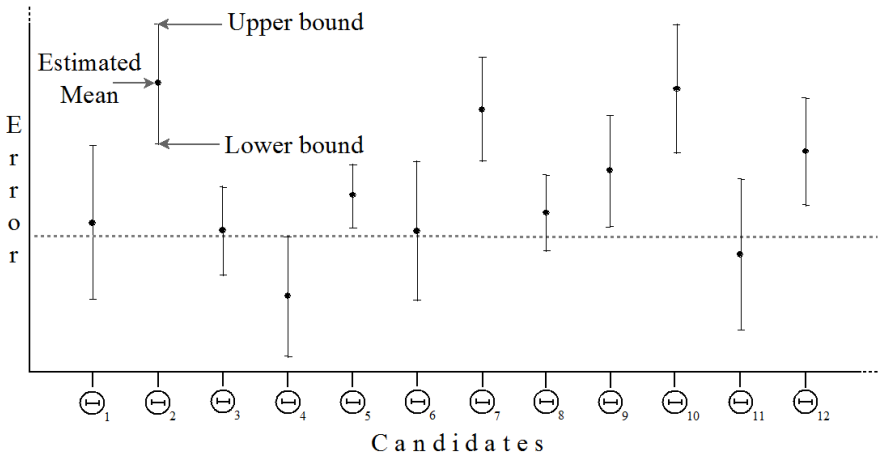


Fig. 1 Race Iteration example

For creating each box: 1) Compute E_{est} , depending on the n test points, 2) Compute e , according to Equation 1, and 3) Calculate the width of the box.

Each box now has a bound within which the true average error lies. Then those boxes whose best possible error (lower bound) is still greater than the worst error of the best box (upper bound) can be eliminated. At each iteration, n is increased causing e is decreased.

The algorithm continues picking test points until one condition occurs: a) All but one of the boxes has been eliminated or b) The algorithm can be stopped once e has reached a certain threshold [2][3].

Figure 1 presents an example of race iteration, where θ_4 is the best configuration (with minimal error), because it has the minor estimated mean. In the worst case, θ_4 is still better than the best cases of θ_5 , θ_7 , θ_9 , θ_{10} and θ_{12} . Because this, they will be eliminated from the competition, and the remainder will continue to compete.

3 Ant Algorithms for Semantic Query Routing

The basis of the ant algorithms can be found in a metaheuristic called ACO (Ant Colony Optimization). The ACO algorithm and its variants (e.g. ACS) were proposed to solve problems modeled as graphs. Each component of the network is represented by a *node* (or a vertex) and the interactions among them represent the *connections* (the edges). ACS needs to know information about all nodes in the network to select the destination node. Ant algorithms were inspired by the ant's behavior, while searching for food. Because when they perform the search, each ant drops a chemical called pheromone which provides an indirect communication among the ants [13].

However, in the Semantic Query Routing Problem (SQRP) the goal is to find one or more destination nodes for a query without having information from the

complete network, requiring operating with local information. The SQRP 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 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 [8][14]. 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.

3.1 Adaptive Neighboring Ant Search

AdaNAS 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.

AdaNAS parameters

During the search process several parameters are used, which are a total of nine: 1) Local pheromone evaporation factor ρ , 2) Importance of local measures (Degree and distance) β_1 , 3) Importance of pheromone β_2 , 3) Relative importance between exploration and exploitation q , 4) Relative importance of the resources found during the time-to-live W_h , 5) Degree weight W_{deg} , 6) Distance weight W_{dist} , 7) Pheromone table initialization τ_0 , 8) Initial value for distances D_0 , and 9) Initial Time-to-live for the search agents TTL_{inic} .

Before using Hoeffding Race to find a valid assignment of parameters, we chose to perform a causal analysis [15][16] to identify which parameters significantly influenced the performance of the algorithm. PC algorithm was used [15][16] to perform causal analysis. This is used for identifying a causal order, in other words, the correct direction of the relationship between two variables and the intensity of the causal relationships found. To establish the right direction of causal relationships PC algorithm needs enough information, in this case, several runs of the algorithm using different values in its parameters. For this example, 800 runs were performed.

To accelerate the algorithm Hoeffding Race we thought appropriate to identify significant parameters, because it cannot distinguish between two configurations when they only vary in the value of a non-significant parameter.

Non-significant parameters were β_2 , W_h , W_{dist} , τ_0 , D_0 [5]. These parameters taking the values recommended by the literature [7][8][13]. The five remaining parameters were adjusted by Hoeffding race. Table 1 presents two possible parameter configurations for AdaNAS, the first (titled Non Tuned Values) presents the recommended values by the literature for these parameters; and the second one (titled Tuned Values) presents the obtained by Hoeffding Race. The methodology applied to select the best setting is shown in Figure 2.

Table 1 Configuration parameter of the AdaNas algorithm

Parameters	Non Tuned Values		Tuned Values	
	Value	Obtained By	Value	Obtained By
μ	0.7	[4]	0.35	Hoeffding Race
ζ_1	2.00	[18]	2.00	Hoeffding Race
ζ_2	1.00	[4,18]	1.00	Recommended
q	0.9	[4,18]	0.65	Hoeffding Race
W_h	0.5	[4,18]	0.5	Recommended
W_{deg}	1.00	[18]	2.00	Hoeffding Race
W_{dist}	1.00	[18]	1.00	Recommended
μ_0	0.009	[4,18]	0.009	Recommended
D_0	999	[18]	999	Recommended
TTL_{inc}	25	[4,18]	10	Hoeffding Race

4 Experiments and Results

In this section, we describe the experiments carried out on the AdaNAS algorithm. We use two versions of the same algorithm, is to say a) AdaNAS algorithm with initial configuration and b) AdaNAS algorithm without initial configuration. The objective of the first experiment is to examine the contribution of initial configuration in an instance, and in the second experiment AdaNAS algorithm is test over 90 different instances.

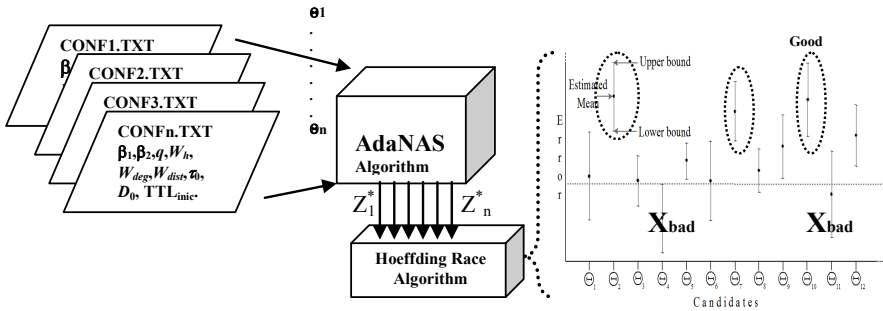


Fig. 2 Methodology for selecting the best configuration

4.1 Experiments Setup

In our implementation, an SQRP instance is determinate by three separate files: topology, repositories, and queries. The description of the instances used can be found at [5][6][7]. Each P2P simulation was run for 20,000 time units (queries).

The average performance was studied by computing performance measures each 100 units of time, called *Average efficiency*, defined as the *hits* (found matching resources) divided by the *hops* (distance between the query node and the node with resources).

The initial configuration of the algorithms is specified in a file containing a *global static configuration*. The configuration of the AdaNAS algorithm used in the experimentation is shown in Table 1.

4.2 The Experiment Results

The performance of the AdaNAS algorithm is analyzed experimentally in order to determine the contribution of the initial configuration that was obtained through Hoeffding Race Technique. For the evaluation of AdaNAS algorithm, the parameters of the algorithms were shown in Table 1.

The first experiment (see Figure 3) is an example of the behavior of the algorithm using the configuration found by Hoeffding race technique. In the second experiment (see Figure 4) are shown the algorithm performance on ninety different instances.

The Figure 3 shows the *average efficiency* reached during the execution of 14000 queries in an instance. For the first configuration –Non Tuned Values–, the algorithm started about at 2.1 average efficiency and at the end the average efficiency is 2.7 hits per hop; and for the second configuration –Tuned Values–, the algorithm start approximately at 2.9 average efficiency; at the end the average hit-rate increases to 3.5 hits per hop. We can see that the average performance always is better using Tuned AdaNAS.

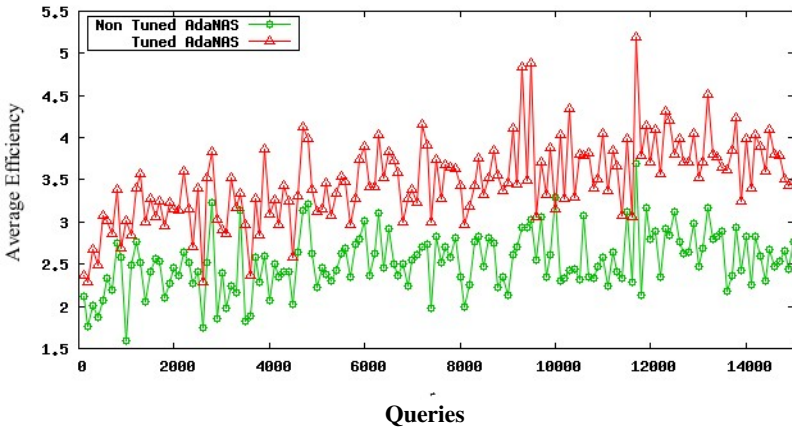


Fig. 3 Average Efficiency of the AdaNAS algorithm, during 14000 queries

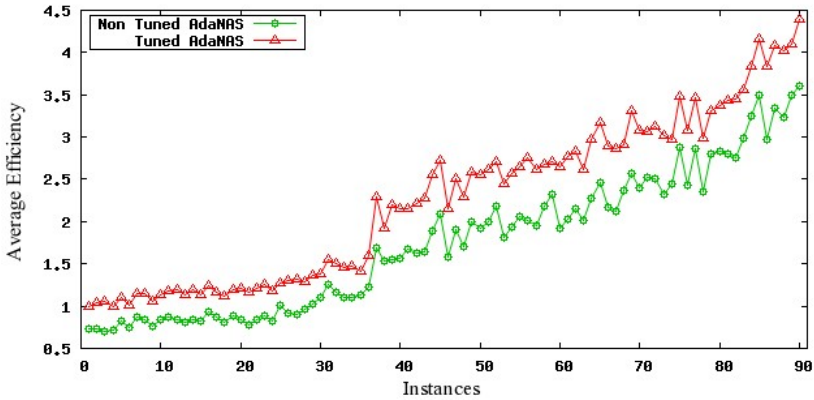


Fig. 4 Average Efficiency of the AdaNAS algorithm, during 90 different instances

Finally in the Figure 4 we show the results with 90 instances and, like in Figure 2, we can see that using the algorithm with the tuned parameters gives better results. It can be saw that, regardless of the hardness of the instances, Hoeffding race provided a better configuration.

AdaNAS with initial configuration –Tuned Values–, show the bigger contribution, giving an average efficiency of 1 hits per hop at the beginning and 4.5 hits per hop at the end, while the Non Tuned AdaNAS algorithm give an efficiency of 0.5 hits per hop at the beginning and 2.3 hits per hop at the end. 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.

5 Conclusions and Future Works

In previous works [5,6,7] we proposed a query routing algorithm that seeks to minimize the search time in the system and maximize the amount of information requested. To improve this algorithm were fundamental the parameter setting.

A simple methodology to do it was presented. The first step consists of finding algorithm's significant parameters, and the second one adjusts them through the Hoeffding race.

The experiment results demonstrated the effectiveness of this methodology as the version that uses the initial configuration outperforms the version with a non-tuned configuration. The improvement in average efficiency of the algorithm was up to 50%.

We are planning to analyze the parameter control of the AdaNAS algorithm, including more features of the problem and test instances, searching to further improve the performance of the algorithm.

References

1. Adenso-Díaz, B., Laguna, M.: Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operation Research*, 99–114 (2004)
2. Maron, O., Moore, A.: Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In: *Advances in Neural Information Processing System*, vol. 6, pp. 59–66 (1994)
3. Maron, O., Moore, A.: The Racing Algorithm: Model Selection for Lazy Learners. *Artificial Intelligent Review* 11, 193–225 (1997)
4. Birattari, M., Stützle, T.: A Racing algorithm for Configuring Metaheuristics. *Artificial Life*, 11–18 (2002)
5. Gómez, C.: *Afinación Estática Global de Redes Complejas y Control Dinámico Local de la Función Tiempo de Vida en el Proceso de Direccionamiento de Consultas Semánticas*, Doctoral Thesis, Instituto Politécnico Nacional, México (2010)
6. Gómez, C., et al.: A Self Adaptive Ant Colony System for Semantic Query Routing Problem. *Computación y Sistemas, Revista Iberoamericana de Computación* 13(4) (2010), ISSN 1405-5546
7. Aguirre, M.: *Algoritmo de Búsqueda Semántica en Redes P2P Complejas*. Master Thesis, Instituto Tecnológico de Ciudad Madero (2008)
8. Michlmayr, E.: *Ant Algorithms for Self-Organization in Social Networks*. Doctoral Thesis, Women's Postgraduate College for Internet Technologies (WIT), Institute of Software Technology and Interactive Systems, Vienna University of Technology (2007)
9. Michalewicz, Z.: *How to solve it: Modern Heuristics*. Springer, NY (2000)
10. Angeline, P.: *Adaptative and Self-Adaptative Evolutionary Computations*. *IEEE Computational Intelligence*, 152–163 (1995)
11. Montgomery, D.C.: *Design and Analysis of Experiments*. John Wiley & Sons, New York (2001)
12. Barr, R., Golden, J., Kelly, M.: Designing and Reporting Computational Experiments with Heuristics Methods. *Journal of Heuristics* 1, 9–32 (1995)
13. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press (2004)
14. Sakaryan, G.: *A Content-Oriented Approach to Topology Evolution and Search in Peer-to-Peer Systems*, PhD Thesis, University of Rostock, Germany (2004)
15. Pérez, V.: *Modelado Causal del Desempeño de Algoritmos Metaheurísticos en Problemas de Distribución de Objetos*. Master Thesis, Instituto Tecnológico de Ciudad Madero (2007)
16. Quiroz, M.: *Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP*. Master Thesis, Instituto Tecnológico de Ciudad Madero (2009)

Improving Distributed Resource Search through a Statistical Methodology of Topological Feature Selection

Claudia Gómez Santillán

Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada, IPN, México

Instituto Tecnológico de Ciudad Madero, Cd. Madero, México

Email: cgomezsa@ipn.mx, cggs71@hotmail.com

Laura Cruz-Reyes, Eustorgio Meza, Tania Turrubiates López, Marco A. Aguirre Lam, and Elisa Schaeffer

Instituto Tecnológico de Ciudad Madero, Cd. Madero, México, Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada, IPN, México, Instituto Tecnológico de Alamo Temapache, Veracruz, México, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, México

Email: lcruzreyes@prodigy.net.mx, emeza@ipn.edu.mx, tania_251179@acm.org, marco@marcoaguirre.com.mx, elisa@yalma.fime.uanl.mx

Abstract—The Internet is considered a complex network for its size, interconnectivity and rules that govern are dynamic, because of constantly evolve. For this reason the search of distributed resources shared by users and online communities is a complex task that needs efficient search method. The goal of this work is to improve the performance of distributed search of information, through analysis of the topological features. In this paper we described a statistical methodology to select a set of topologic metrics that allow to locally distinguish the type of complex network. In this way we use the metrics to guide the search towards nodes with better connectivity. In addition we present an algorithm for distributed search of information, enriched with the selected topological metric. The results show that including the topological metric in the Neighboring-Ant Search algorithm improves its performance 50% in terms of the number of hops needed to locate a set of resources. The methodology described provides a better understanding of why the features were selected and aids to explain how this metric impacts in the search process.

Index Terms— Internet, search process, query routing, random walk, ant colony system, scale free, topology, experiment designs, statistical analysis, metrics

I. INTRODUCTION

Complex systems can be modeled by means of complex networks, because of have a non-trivial topological structure. These features have motivated the study of topological features of real-world networks such as the Internet. Knowledge on such features can be used to optimize the performance of processes carried out on the Internet, for example: the search of distributed resources, traffic management, and design of routing queries [1], among others. The main goal is to help the users to find the information that they request with a reasonable processing time and with a higher quality of the information obtained [2].

Over the past years, new communication models have emerged in the Internet that manage information in a distributed manner and offer significant advantages over centralized information management systems. These systems are known as *peer to peer* networks (P2P). In a P2P network, a set of nodes form connections to offer their resources to the other nodes within the network. The P2P systems, together with the underlying communication network (typically the Internet), form a complex system that requires autonomous operation through mechanisms of intelligent search [1].

Until now, a great number of topologic metrics have been developed to characterize the complex networks, but the majority of these metrics are global. This implicates a great computational effort (processing time and memory). For this reason, it is necessary to identify a topological metric that locally allows to obtain information about the type of network. In this way the distributed search process would take advantage of the topology. This point requires sufficient empirical evidence that supports the use of a topological metric to locally identify the type of network. This raises important questions: what topological metric to select in order to locally identify the types of complex networks? Does the topologic metric allow to improve the performance of the distributed search process? If so, how much the performance of the distributed search process is improved? Why the topological metric can identify the type of network?

In this paper, a methodology based in statistical analysis is described. The goal of this methodology is to identify, by the means of an experimental design and a series of statistical tests, a set of topologic metrics that allow to locally recognize the type of a complex network. This minimum set is used to analyze the performance of a distributed search algorithm for textual information,

enriched with a topological metric to characterize local topology. The study of such distributed algorithms is important as the quality of the retrieved information as well as the time necessary for its retrieval are key factors in the performance of a P2P system.

II. THEORIC FRAME

A. Peer-to-Peer Networks Modeled as Complex Networks

A *system* is a set of interrelated components that seeks a common goal. Any system that can be understood as a set of components whose connections follow a certain rule can be modeled as a network: each component is represented by a node of the network and all existing interactions are captured by the connections of the network [3].

Complex systems are those systems that have a very large number of components and the connections among the components 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 [3].

A *P2P network* is a distributed system where all the nodes are equal in terms of functionality and tasks performed in the P2P system [4]. The objective of a P2P network is to share resources such as information (documents, music, videos), hardware resources (computational capacity, memory), or peripheral devices (printers, cameras).

Such structure formed by pairs of connected nodes can be modeled as the edges of a dynamic network, where edges and nodes may appear and disappear at any time. Hence the structure of a P2P network can be modeled as a complex network [1, 5]. One of the main motivations for modeling systems as complex networks is the flexibility and generality of the abstract representation that allows handling properties such as dynamic topology in a natural way [6].

B. Information Search

The problem of locating textual information in a P2P network over the Internet is known as *semantic query routing* (SQR), where the goal is to determine the shortest paths from a node that issues a query to those nodes that can appropriately respond to the query (by providing the requested information) [1]. The query traverses the network moving from the initiating node to a neighboring node and then to a neighbor of a neighbor and so forth until locating the requested resource (or giving up in its absence).

The challenge lies in the design of algorithms to traverse the Internet in search of resources – modeled as a

complex network – in an intelligent and autonomous manner. In order to reach this goal, the algorithms proposed for this problem include the selection of the next node to visit, using information of near-by nodes of the current node, that is, information on the *local topology* of the current node.

C. Random Walk

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 [8]. Let G be a graph that models the network and v a vertex in G . A T -hop random walk from v in G is a sequence of dependent random variables X_0, \dots, X_T defined as follows: $X_0 = v$ with probability 1 and for each $i = 1, \dots, T$, the value for X_i is selected uniformly at random among the vertices $\Gamma(X_{i-1})$, that is, among the neighbors of the vertex of the preceding step. Simply put, a random walk begins at a certain vertex and on each step, moves to a neighbor of the current vertex, until it arrives to a vertex that meets to goal. In our network, that would be a vertex that represents a node that contains the requested resource [7].

D. Neighboring-Ant Search

In the area of classification, feature selection has great benefits such as improving the performance of classification procedures and constructing simple and comprehensible classification models. These are achieved eliminating irrelevant and redundant features that may introduce noise that could affect the efficiency of the procedure. The goal of feature selection is to choose a minimum subset of features that can discriminate efficiently among different classes. This minimum subset is known as the *optimal* subset [24]. The majority of the feature selection methods involve the search in the feature space to predict the best class and the evaluation of the features to measure the fitness of a subset [25].

E. Experimental Design

Understanding a particular system or process demands observation, modeling, and experimentation. The experimentation aims to generalize away from context specific measurements and to build insight into fundamental structures and properties of a system. In this way the experimentation provides knowledge of the domain of interest [11, 12, 13].

Learning involves the encapsulation of knowledge, checking that the knowledge is correct, and evolving that knowledge over time. The experimental paradigm is used in many fields, including physics, medicine, and manufacturing. Like other sciences, many disciplines within computer science likewise require an empirical paradigm [11, 14]. Since the experimental subject and the research questions are somewhat unusual compared to

other problem domains, much more work is needed to identify the statistical and data analysis tools most appropriate to these types of problems [12, 13, 14].

F. Degree Disperion Coefficient

The DDC measures the differences between the degree of a vertex and the degrees of its neighbors. A node i is said to be a *neighbor* of a node j if they are connected in the network. The *degree* k_i of a node i is the number of neighbors it has. The DDC of node i is defined in equation (1), $\sigma(i)$ is the degree variation among i and its neighbors and $\mu(i)$ is their average degree [26].

$$DDC(i) = \frac{\sigma(i)}{\mu(i)};$$

where $\sigma(i) = \sqrt{\frac{\sum_{j \in \Gamma(i)} [k_j - \mu_i]^2 + [k_i - \mu_i]^2}{k_i + 1}};$ (1)

and $\mu(i) = \frac{\sum_{j \in \Gamma(i)} [k_j] + k_i}{k_i + 1}$

III. RELATED WORKS

Existing methods for classifying real-world networks using topological features [9, 10, 15] do not provide a detailed statistical analysis to determine if all of the features used are necessary or optimal to efficiently discriminate among types of networks. Costa *et. al.* [6] used statistical techniques to identify the type of a network with unknown nature. The results show that the type of network assigned to the networks, varies according to the topological features selected, and that excessive number of features can compromise the quality of the classification.

In the above mentioned methods the topological features used are *global*: computing each feature requires processing the entire network. This involves a great computational effort with respect of both time and memory.

One of the problems of interest is the semantic query routing on the Internet. The most relevant works in this area address this problem using ant-colony algorithms [1, 22, 23]. The principal difference of the present work with existing methods is the incorporation of a strategy that takes advantage of the environment where the search takes place, in terms of a local structural metric is. The structural metric was selected through a statistical methodology described in Section 3.

IV. STATISTICAL METHODOLOGY

The main goal of the statistical methodology proposed is to identify which topological features are relevant and non-redundant. Let us first define relevancy feature. Let

us suppose there are k different populations of networks, from which topological features are extracted. Three possible cases can be identified, c.f Figure 1:

- **Strong Relevance:** the differences among the k types of networks are sufficient to be able to classify a new network based on the features measured and the populations do not overlap.
- **Weak Relevance:** the differences among the k types are strong but insufficient to perfectly classify a new network due to population overlap.
- **Irrelevancy:** the differences among the k types of networks are insufficient for classification.

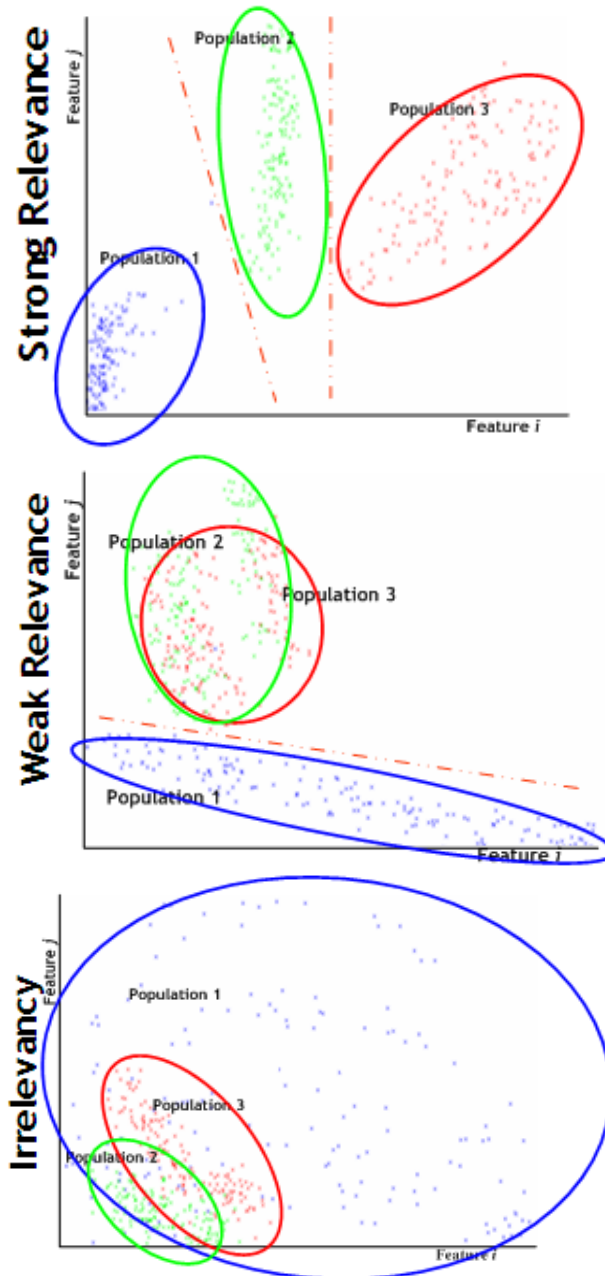


Figure 1. Three possible cases of relevancy feature.

The statistical methodology proposed to select relevant and non-redundant features has four basic steps; *c.f.* Figure 2:

1. **Identify relevant and irrelevant features:** This step consist in determining which features differ significantly according to the type of network, regardless of the number of nodes in the network. The features with different means are considered as relevant features whereas the features with equal means are irrelevant. The experimental design used to discriminate between relevant and irrelevant features is described by Cruz *et al.* [16] and Turrubiates *et al.* [17]
2. **Identify features with strong relevance and weak relevance:** In this step the relevant features are analyzed by means of a multiple comparison method to determine which of them are strong relevant features and which are weak relevant features.
3. **Redundancy elimination:** The goal in this step is to eliminate the weak relevant features correlated with the strong relevant features in such a way that the set of selected features contain the majority of the strong features and some of the weak features.
4. **Identify the minimal set:** Combinations of selected features are made and a discriminant analysis is carried out to determine the

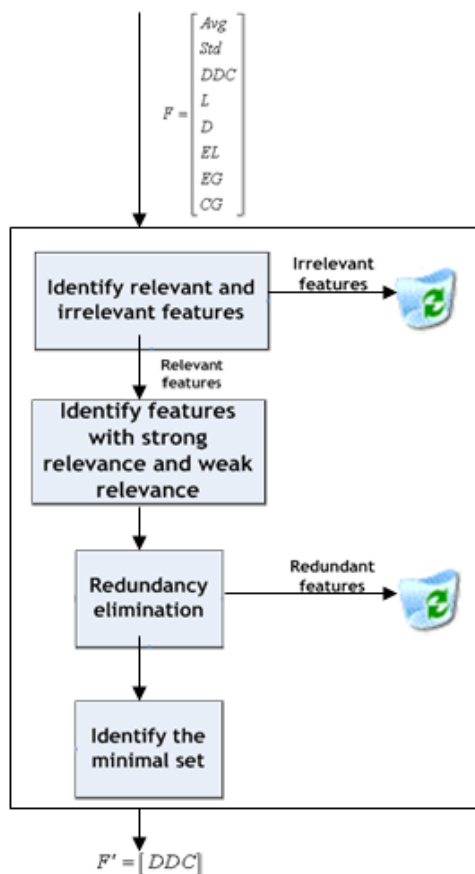


Figure 2. Steps of feature selection

combination with the lowest number of features that produces the best performance in the discriminant analysis. This combination is called the *minimal set*.

V. EXPERIMENTS

A. Statistical Methodology

The topological features analyzed were: average degree Avg , standard deviation of the degree Std , clustering coefficient CG , global efficiency E_{glob} , local efficiency E_{loc} , shortest path length L , diameter D and the degree dispersion coefficient DDC . A detailed description of these features could be found in Costa *et al.* [6]. The methodology considers that the features could be affected by the type of network and the number of nodes; we generated three kinds of complex networks with different sizes [18].

B. Distributed Resource Search

We generated complex networks with the scale-free network method of Barabási *et al.* [20], where nodes are added one at a time with a fixed number of connections each. The newly-arriving node chooses preferentially at random among the existing nodes to which to connect, giving preference to high-degree nodes. The resulting network has a small number of highly connected nodes while the majority of the nodes have degree close to average degree. The network size was set to 1,024 nodes.

The studied algorithms were the proposed Neighboring-Ant Search (NAS) and the random-walk (RW) algorithm that serves as a base case for comparison. We experimented on two versions of both algorithms: with and without the DDC. Only one ant was used per query and the time-to-live of the ants was set to 25 hops. The number of results (hits) needed to satisfy the query was set to five.

The time steps of the experiments were of 100 ms and the simulations were ran for 10,000 steps. During each time step, each node has a probability of 0.1 to launch a query. The “topics” of the resources were modeled as integer values from zero to 1,024, generated using the uniform-distribution generator of Repast [21]. Each node was assigned ten resources with possibly repeated topics. The queries were generated to search a topic uniformly at random from 0 to 1,024. The experiments were carried out on a workstation with an Intel Xeon a 3GHz processor with 4GB of RAM.

C. Random Walk

Optionally, one can include the DDC function into the random-walk algorithm. A simple modification to include structural preferentiality is to choose uniformly at random two neighbors, calculate their DDC values, and move on to the neighbor with higher DDC.

D. Neighboring-Ant Search

The NAS algorithm has two objectives: directing the

TABLE I. NAS ALGORITHM PSEUDO-CODE

```

01 for each query
02 repeat while the forward ant is active
03 if Hits < maxResults and TTL > 0 // Phase 1
04 if the neighbor from edge s_k has results
05 append s_k to Path_k
06 TTL_k = TTL_k - 1
07 globalUpdate // backward ant
08 else // Phase 2
09 s_k = apply the transition rule
10 if path does not exist or node was visited,
11 remove the last node from Path_k
12 else,
13 append s_k to Path_k
14 TTL_k = TTL_k - 1
15 localUpdate
16 endif
17 endif
18 else
19 Kill the forward ant
20 endif
21 endif
    
```

queries towards the nodes that have good connectivity using the DDC while minimizing the number of hops needed to respond to queries. This latter objective is achieved by a function called *importance of hops* that is the inverse of the sum of all connections traversed during the search. The number of hops is the lifetime of the ant

TABLE II. STATISTICAL TOOLS USED IN THE PROPOSED METHODOLOGY

Step	Statistical test	Results
Identify relevant and irrelevant features	Experimental Design: Two factor mixed factorial Statistical Test: MANOVA, Residuals Analysis, Interaction Plots.	✓ Relevant features: <i>Std</i> , <i>DDC(G)</i> , <i>L(G)</i> , <i>D(G)</i> , <i>E_{loc}</i> , <i>E_{glob}</i> × Irrelevant features: <i>Avg</i> , <i>CG(G)</i>
Identify features with strong relevance and weak relevance	Multiple comparison using the Tukey test	Strong relevant features: <i>Std</i> ,, <i>DDC(G)</i> , <i>L(G)</i> , <i>D(G)</i> Weak relevant features: <i>E_{loc}</i> , <i>E_{glob}</i>
Redundancy elimination	Correlation Analysis	✓ Not redundant features: <i>Std</i> ,, <i>DDC(G)</i> , <i>L(G)</i> , <i>E_{loc}</i> × Redundant features: <i>D(G)</i> , <i>E_{glob}</i>
Identify the minimal set	Discriminant Analysis	Minimal set: <i>DDC(G)</i> .

represented by *TTL* and the maximum value is set at 25 hops. A pseudo-code for the algorithm is given in Table 1 [19].

VI. RESULTS

In Table 2, the statistical tests used in each step of the methodology are described together with the obtained results, *c.f* [16, 18].

The subset obtained in the steps that correspond with

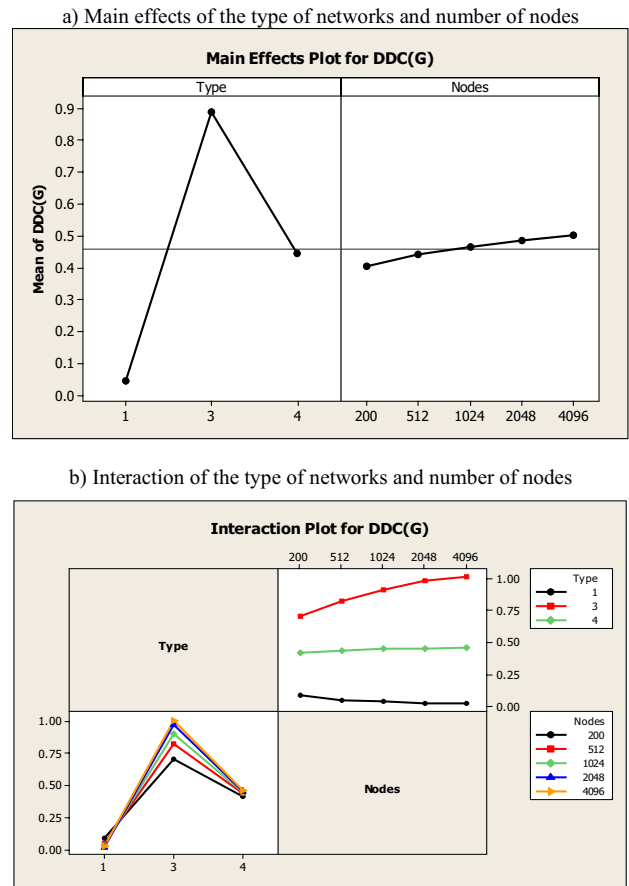


Figure 3. Factor effects for the *DDC (G)* feature.

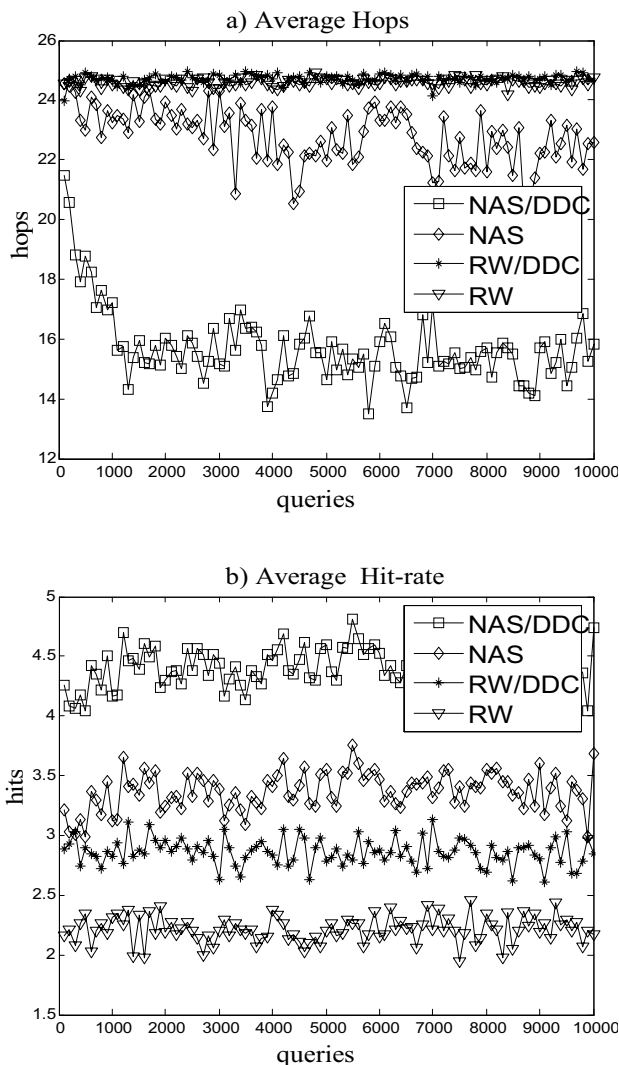
the search in the feature space is formed by the standard deviation of the degree $\sigma_{(k)}$, local efficiency E_{loc} , shortest path length L , and the degree dispersion coefficient *DDC*. The evaluation of this subset identified the *DDC* as the feature that by its own has a best performance in the discriminant analysis.

The *DDC* was selected (*c.f* Figure. 3) as the relevant and not redundant feature that can discriminate efficiently among the three types of complex networks. The analysis of the topological features selected shows that the type of network has a greater effect on the results than the number of nodes in the network. This is important in locally identifying the type of a network; this feature captures sufficient information about the types of networks [18].

After having selected the *DDC* as the relevant and not redundant feature, was included into the two search algorithms, obtaining the following results. In Figure 4(a), it can be seen that in scale-free networks, the

number of hops used by the RW algorithm is very close to the TTL of 25 hops, with or without the DDC. For the NAS algorithm without DDC, approximately 19 hops are needed. Incorporating DDC into NAS reduces the number of hops even further down to 10.

Figure 4(b) shows the average hit-count. On scale-free topologies, the RW algorithm obtains 2 – 2.5 results per query without using the DDC, and with DDC, the number of hits rises to 2.5 – 3. In the same networks, the NAS algorithm obtains 3 – 3.5 hits per query without the DCC, and 4 – 4.5 hits when DCC is used.



These observations confirm the intuition that the DDC in the presence of a scale-free distribution allow a significant improvement to the search performance, which also implies that the NAS algorithm outperforms in such topologies the existing methods that do not incorporate local structural information.

VII. CONCLUSIONS AND FUTURE WORK

In this work a statistical methodology was defined and developed to identify the minimal set of topological features that allows to discriminate among three different types of complex networks. The use of this methodology

allows us to justify why the features were selected and provide information of the influence of the type of network and the number of nodes in the prediction power of the selected features.

The result of applying the methodology to a set of eight topological functions resulted on minimal set containing the DDC metric [18]. Subsequently, this metric was used in semantic query routing algorithms. We observe that upon including DDC in the algorithm NAS, the hop count decreases by 50% and the hit count is improved by 15%. The random-walk algorithm used as a comparison gains no advantage of DDC in terms of the hop count, and a benefits very little in terms of hit count (3% improvement).

As future work, we plan to study more profoundly the impact of the metrics employed in the learning curve of ant-colony algorithms as well as the effect on the performance measures of hop and hit counts. We also contemplate using more than one ant per query to parallelize the algorithm in hopes of improved performance.

REFERENCES

- [1] Michlmayr E.: *Ant Algorithms for Self-Organization in Social Networks*. Ph.D. Thesis, Vienna University of Technology, Vienna, Austria, 2007.
- [2] Arenas A., Danon, L., Díaz-Guilera, A., and Guimera, R.: Search and Congestion in Complex Networks. In *Statistical Mechanics of Complex Networks: XVIII SITGES Conference on Statistical Mechanics*, Lecture Notes in Physics, vol. 625, 175 - 194. Springer Verlag, 2003.
- [3] Amaral, L. A. N., and Ottino, J. M.: *Complex Systems and Networks: Challenges and Opportunities for Chemical and Biological Engineers*. Chemical Engineering Scientist (59): 1653–1666, 2004.
- [4] Androutsellis-Theotokis, S., and Spinellis, D.: *A Survey of Peer-to-Peer Content Distribution Technologies*. ACM Computing Surveys, 36(4): 335–371, 2004.
- [5] Ortega, R.: *Estudio de las Propiedades Topológicas en Redes Complejas con Diferente Distribución de Grado y su aplicación en la búsqueda de recursos distribuidos*. Technical Report, Centro de Investigación en Ciencia Aplicada y Tecnología Aplicada, Altamira, Mexico, 2005.
- [6] F. Costa, L., Rodriguez, F. A., Travieso, G., and Villas Boas, P. R.: *Characterization of Complex Networks: A survey of measurements*, Advances in Physics, 56(1): 167-242, 2007.
- [7] Arora, S., and Barak, B.: *Complexity Theory: A Modern Approach*. Book in preparation, 2007.
- [8] Dorigo, M., and Gambardella, L. M.: *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*. IEEE Transactions on Evolutionary Computation, 1(1): 53-66, 1997.

- [9] Airoidi E.M., and Carley K.M.: *Sampling algorithms for pure network topologies: a study on the stability and the separability of metric embeddings*. ACM SIGKDD Explorations Newsletter 7:13-22, 2005
- [10] Middendorf M., Ziv E., Carter A., Hom J., Koytcheff R., Levovitz C., Woods G., Chen L., and Wiggins C.: *Discriminative Topological Features Reveal Biological Network Mechanisms*. BMC Bioinformatics 5:181, 2004.
- [11] Basil V.R., and Zelkowitz M.V.: *Empirical Studies to Build a Science of Computer Science*. Communications of the ACM 50:33-37, 2007.
- [12] McGeoch C.C.: *Experimental Algorithms*. Communications of the ACM 50:27-31, 2007.
- [13] Montgomery D.C.: *Design and Analysis of Experiments*. John Wiley & Sons. New York, 2001.
- [14] Hooker J.N.: *Needed: An Empirical Science of Algorithms*. Operations Research 42:201-212, 1994.
- [15] Ali W., Mondragón R.J., and Alavi F.: *Extraction of topological features from communication network topological patterns using self-organizing feature maps*. arXiv:cs/0404042v2, 2004.
- [16] Cruz R.L., Meza C. E., Turrubiates L.T., Gomez S.C., and Ortega I.R.: *Experimental Design for Selection of Characterization Functions that Allows Discriminate Among Random, Scale Free and Exponential Networks*. Polish Journal of Environmental Studies 16:67-71, 2007.
- [17] Turrubiates L.T., Gómez S.C., Cruz R.L., Ortega I.R., and Meza C. E.: *Diseño Experimental para Selección de Funciones de Caracterización que Permitan Discriminar entre Redes Aleatorias, de Escala Libre, y Exponenciales*. 14th International Congress on Computer Science Research CIICC'07. 161-171, 2007.
- [18] Turrubiates L.T.: *Clasificación de Redes Complejas usando Funciones de Caracterización que Permitan Discriminar entre Redes Aleatorias, Power-Law y Exponenciales*. M.Sc. Thesis, IT Cd. Madero, Madero, 2007.
- [19] Cruz-Reyes L., Gómez S.C., Aguirre M.A.L., Schaeffer E.S., Turrubiates L.T., and Ortega I.R.: *NAS Algorithm for Semantic Query Routing Systems in Complex Networks*, accepted in DCAI'08.
- [20] Barabási A.L., Albert R., and Jeong H.: *Mean-Field theory for Scale-free Random Networks*. *Physica A*, 272: 173–189, 1999.
- [21] ROAD: *Repast Organization for Architecture and Design*. Repast Home Page. Chicago, IL, USA. <http://repast.sourceforge.net>, 2005.
- [22] Yang K.H., Wu C., and Ho J.M.: *AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks*. IEICE Transactions on Communications, 89(9): 2300-2308, 2006.
- [23] Babaoglu O., Meling H., and Montesor A.: *Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems*. In Proceedings of the 22nd International Conference on Distributed Computer Systems (ICDCS 02). IEEE, 2002.
- [24] Yu, L., Liu, H.: *Efficient Feature Selection via Analysis of Relevance and Redundancy*. Journal of Machine Learning Research 5. 1205 – 1224, 2004
- [25] Witten, H.I., Frank, E.: *Data Mining. Practical Machine Learning Tools and Techniques*. Second Edition. Elsevier, 2005.
- [26] Ortega, R., Meza, E., Gómez, G., Cruz, L., Turrubiates, T.: *Impact of Dynamic Growing on the Interner Degree Distribution*. In Frontiers of High Performance Computing and Networking ISPA 2007 Workshops, Lecture Notes in Computer Science. vol. 4743. Springer Verlag. 119-122, 2007.

Claudia Gómez S. 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) degree 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.

Tania Turrubiates López is a doctoral student in Autonomous University of Nuevo Leon. She obtained her MS degree in Computer Science from Madero City Institute of Technology, Mexico in 2007. Her research interests are optimization techniques and complex network.

Marco Aguirre Lam. is a MS student in Computer Science at Madero City Institute of Technology, Mexico. Her research interests are optimization technique, complex network and autonomous agents.

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.

Multiple Local Searches to Balance Intensification and Diversification in a Memetic Algorithm for the Linear Ordering Problem

Héctor Joaquín Fraire Huacuja, Guadalupe Castilla Valdez,
Claudia G. Gómez Santillan, Juan Javier González Barbosa, Rodolfo A. Pazos R.,
Shulamith S. Bastiani Medina, and David Terán Villanueva

Instituto Tecnológico de Ciudad Madero, México
1o. de Mayo y Sor Juana I. de la Cruz S/N C.P. 89440
Cd. Madero, Tamaulipas, México
hfraire@prodigy.net.mx, gpe_cas@yahoo.com.mx,
cgs71@hotmail.com, jjgonzalezbarbosa@yahoo.com.mx,
r_pazos_r@yahoo.com.mx, b_shulamith@hotmail.com,
david_teran01@yahoo.com.mx

Abstract. The Linear Ordering problem (LOP) is an NP-hard problem, which has been solved using different metaheuristic approaches. The best solution for this problem is a memetic algorithm, which uses the traditional approach of hybridizing a genetic algorithm with a single local search; on the contrary, in this paper we present a memetic solution hybridized with multiple local searches through all the memetic process. Experimental results show that using the best combination of local searches, instead of a single local search, the performance for XLOLIB instances is improved by 11.46% in terms of quality of the solution. For the UB-I instances, the proposed algorithm obtained a 0.12% average deviation from the best known solutions, achieving 17 new best known solutions. A Wilcoxon test was performed, ranking the proposed memetic algorithm as the second best solution of the state of the art for LOP. The results show that the multiple local searches approach can be more effective to get a better control in balancing intensification/diversification than the single local search approach.

Keywords: Metaheuristics, Memetic algorithm, Linear Ordering Problem, Local Search, Intensification/diversification balance.

1 Introduction

Given a C_{ij} matrix of weights of size $n \times n$, the linear ordering problem (LOP) consists in finding a permutation p of columns (and rows) such that the sum of the weights in the upper triangle is maximized. Its mathematical expression is:

$$\max \left(C_{LOP}(p) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n C_{p(i), p(j)} \right) \quad (1)$$

Here p_i is the index of column (and row) i in the permutation. In LOP, the permutation p provides the ordering of rows and columns, as described by Laguna [1].

The Linear Ordering Problem has been described as an NP-hard problem by Karp and Thatcher [2] and Garey and Johnson [3].

Important applications of LOP exist in scheduling, social sciences, electronics, archeology, and particularly in economics, where it is applied to solve the problem of triangulating the input-output tables in the economic model by Leontief [4].

2 Related Work

The memetic algorithm by Schiavinotto and Stutzle [5] currently offers the best solution of the state of the art for LOP. In this work a genetic algorithm was hybridized with a single local search on an insertion neighborhood. The single local search included in the memetic algorithm uses a criterion between first and best. The final configuration of memetic includes a population size of 25 individuals, an offspring formed by 11 children generated by OB crossover, a diversification by reinitializing of population (after 30 iterations have occurred without changes in fitness average). The final memetic algorithm didn't include a mutation operator.

One of the best solutions for LOP is the Tabu search by Laguna et al. [1]. This solution includes an intensification phase using short-term memory based on a tabu criterion, a diversification process through a long-term memory that uses a frequency register, and an additional intensification process that applies a path relinking strategy based on elite solutions. The Tabu search algorithm uses two different local searches (first and best) in its different processes. These local searches explore an insertion neighborhood through consecutive swap movements.

A benchmark library and a comprehensive study of heuristic methods were developed by Martí [6]. In this work, under the same experimental conditions and on the same standard sets of instances (OPT-I and UB_I), 10 of the top performing heuristic algorithms are assessed. The highest performing algorithms were the memetic algorithm and Tabu search.

Memetic algorithms are commonly implemented as evolutionary algorithms endowed with a single local search component [7]. The use of multiple local searches in the memetic algorithm is associated to learning approaches [8], but this strategy has a high computational cost. On the contrary, we propose a less expensive new approach by incorporating in a memetic algorithm multiple local searches with different levels of intensification, according to the requirements of the processes included in the algorithm.

3 Local Searches

The insertion neighborhood used by the local searches implemented, is defined by the insert operation: $\pi \times \{1, \dots, n\}^2 \rightarrow \pi$.

$$\text{insert}(\pi, i, j) \triangleq \begin{cases} (\dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_j, \pi_i, \pi_{j+1}, \dots) & i < j \\ (\dots, \pi_{j-1}, \pi_i, \pi_j, \dots, \pi_{i-1}, \pi_{i+1}, \dots) & i > j \end{cases} \quad (2)$$

This neighborhood was implemented applying the exploration strategy inspired on the Dynasearch method proposed by Congram [9]. The insertion is made by a series of swap movements between adjacent elements, where $i = j \pm 1$. The change in the objective function value is given by:

$$\Delta_s(\pi, i, j) \triangleq \begin{cases} c_{\pi_i \pi_j} - c_{\pi_j \pi_i} & i = j + 1; \\ c_{\pi_j \pi_i} - c_{\pi_i \pi_j} & i = j - 1. \end{cases} \quad (3)$$

Since the change evaluation for each movement is constant, the cost to evaluate the neighborhood is $O(n^2)$. For optimizing the execution time, the costs of all the swap moves are pre-calculated like in the Tabu search by Laguna [1]. The expression for this calculation is:

$$d_{ij} = c_{ij} - c_{ji} \quad \forall i, j = 0..n-1. \quad (4)$$

In this work two local search algorithms with different levels of intensification/diversification were implemented. The algorithm in Figure 3 corresponds to LS_f local search [5], which has an external cycle to examine all the sectors in the permutation order. For each sector, an internal cycle evaluates all the neighbors searching for the best (if none is better it returns the less worse neighbor solution).

```

LSf Algorithm( $\pi$ )
for ( $i = 0..n-1$ ) do
   $r^- \leftarrow \arg \max_{r, r=i} f(\text{insert}(\pi, i, r^-))$ 
   $\pi^2 = \text{insert}(\pi, i, r^-)$ 
  if  $f(\pi^2) > f(\pi)$  then
    return ( $\pi^2$ )
  end_if
end_for
return ( $\pi$ )
end_LSfAlgorithm( $\pi$ )

```

Fig. 1. LS_f algorithm

On the other hand the *Best* local search [1] is much more intensive: it starts from a feasible solution which is improved over the time. All sectors are scanned in the permutation order; all the consecutive positions forward and backward from the current position are examined, choosing the position which produces the highest increment (or the lowest decrease) in the objective function value for carrying out the movement. The process is repeated as long as the current solution is improved, Figure 4 shows the algorithm for the *Best* local search. It is worth mentioning that the *Best* local search is more intensive and its cost is greater than the cost of the LS_f algorithm. As we can see, both searches have a variable cost.

```

Best Algorithm ( $\pi$ )
do
  for ( $i = 1..nsectors$ ) hacer
     $r^- \leftarrow \arg \max_{r, r \neq i} f(\text{insert}(\pi, i, r^-))$ 
     $\pi^2 = \text{insert}((\pi, i, r^-))$ 
  end_for
  while ( $f(\pi^2) > f(\pi)$ )
  return ( $\pi^2$ )
end Best Algorithm ( $\pi$ )

```

Fig. 2. Best algorithm

4 Proposed Memetic Algorithm (MLSM)

The proposed memetic algorithm that we denote by the acronym MLSM (multiple local search memetic) is hybridized with different local searches. Figure 1 shows the pseudo-code for MLSM.

```

MLSM Memetic Algorithm
Population  $\leftarrow$  RandomGeneratepopulation();
ImprovePopulation  $\leftarrow$  BestLocal Search(Population);
while (not stop condition)
  if stagnation then
    Population  $\leftarrow$  Selectbestindividual(Population);
    Population  $\leftarrow$  RandomGeneratepopulation(); //diversification
    ImprovePopulation  $\leftarrow$  LSFLocalSearch(Population);
  end_if
  for  $i \leftarrow 1..#crossovers$  do
    select  $\pi_a, \pi_b$  from Population();
    offsprings  $\leftarrow$  OB_crossover();
    ImprovePopulation  $\leftarrow$  BestLocalSearch(Population);
  end_for
  Population  $\leftarrow$  SortPopulation(Population);
  best_individual  $\leftarrow$  Select_best(SortedPopulation)
end_while (reach stop condition);
end_Memetic Algorithm

```

Fig. 3. Proposed memetic algorithm MLSM

In the proposed memetic algorithm, the initial population is obtained by generating randomly 36 individuals, to which the *Best* local search is applied.

For subsequent generations, two parents are randomly selected for applying crossover. From the first parent a set of positions are randomly selected ($0.4 * \text{instance size}$), then the values in the non-selected positions are copied directly to the corresponding positions of the offspring. In the next step, the values in the selected positions are ranked according to the order in the second parent and copied to the vacant positions of child. Figure 2 shows this crossover process. In the selection

process, the first father is randomly chosen according to a uniform distribution and the second parent is chosen giving preference to the best elements of the population. A general condition for the selection of parents is that they haven't recently generated offspring. The *Best* local search is applied to the new population.

To build the new population, the 25 best individuals are chosen from the current population and from the generated offspring, duplicates are eliminated.

An extra diversification by randomly regenerating the individuals in the current population is applied, remaining only the best individual. This process is triggered to avoid premature convergence when the average of the population objective function does not change during five consecutively generations. In this case an LS_f Local search is applied to all the individuals in the new population.



Fig. 4. An OB crossover example

5 Experimental Results

Experimental work was conducted on a Dell Power Edge 2600 with a XEON processor at 3.06 GHZ and 4GB in RAM. The algorithm was coded in C++, and compiled in Visual Studio 2005. Each algorithm was executed using two time limits (10 and 600 seconds) for each instance and a seed of 1471. The instances were grouped into four sets: RandAII, Spec, RandB, XLOLIB, and RandAI.

A first experiment to assess different combinations of the local search strategies (*Best* (B) and LS_f (LSF)) on the memetic algorithm was performed. They are applied to the initial population, to the population after the crossover and to the population after the diversification process. The experiment was carried out using the XLOLIB instances. Table 1 shows the results for the best combinations. The first column shows the combination of local searches. The combination (B, B, B) corresponds to a single local search memetic algorithm (SLSM, traditional approach), that is used as base case. The third column includes the average deviation from the best known solution (%), and the fourth column contains the difference in performance for the corresponding combination with respect to the performance of the base combination. The best combination (B, LSF, B), highlighted in the table, achieves a performance improvement of 11.4% with respect to the base case. In the table, the acronym NA indicates that no local search was applied.

A second experiment was carried out to assess the performance of the proposed memetic algorithm, which incorporates the best configuration of local searches. The MLSM algorithm was compared average wise with the two best metaheuristics: tabu search by Laguna (TS) [1] and memetic algorithm by Schiavinotto (SM) [5]. The results for these two metaheuristics were obtained from the benchmark library developed by Marti et al. [6] and are shown on Table 2.

Table 1. Average deviation from best known solutions and percentage improvement with respect to the base SLSM algorithm

Combination of local searches	Memetic algorithm	Average deviation from best known (%)	Percentage improvement versus base SLSM
B, B, B	Base SLSM	0.18974923	0
LSF, LSF, B	MLSM	0.19692336	-3.78084802
B, LSF, B	MLSM	0.16805435	11.43344824
LSF, B, B	MLSM	0.19661945	-3.62068399
NA, NA, B	SLSM	0.17349502	8.566153338
LSF, LSF, LSF	SLSM	0.42988623	-126.554927
B, LSF, LSF	MLSM	0.40227557	-112.003796

Table 2. Experimental results for UB-I instances (10 sec.)

Instances	Performance	TS by Laguna [3]	SM by Schiavinotto [5]	MLSM
RandA1	% Error (Avg)	0.12	0.05	0.11
	# Best	5	33	19
RandA2	% Error (Avg)	0.01	0	0.001
	# Best	3	39	33
RandB	% Error (Avg)	0	0	0.27
	# Best	20	20	0
XLOLIB	% Error (Avg)	0.61	0.12	0.169
	# Best	0	2	1
Spec	% Error (Avg)	0.45	0.049	0.073
	# Best	3	3	4
	% Average Error	0.23	0.04	0.12
	#Total Best	31	97	57

As can be seen, the proposed MLSM algorithm achieves a comparable performance to the memetic algorithm by Schiavinotto (SM). Afterwards non-parametric Wilcoxon tests were performed to compare MLSM with respect to the tabu and memetic algorithms, applying time limits of 10 and 600 seconds. MLSM significantly outperforms the Tabu solution for the two different time limits (the results are not shown by space restrictions). The results of the experiment with SM algorithm are shown in the Table 3. In this table, when R^+ is larger than R^- , the SM algorithm is better than MLSM, the opposite is true if R^- is greater than R^+ . If the greatest of R^+ and R^- is greater than the reference value (R), then the performance difference is statistically significant, otherwise both algorithms have the same

performance. In all cases, considered reliability is 0.05. As we can see, the memetic algorithm by Schiavinotto[5] outperforms significantly the performance of MLSM in 6 of 10 cases for 10 seconds. In the test with 600 seconds MLSM was outperformed by SM only in 3 of 10 cases. The performance increment of the MLSM algorithm could be explained because the multiple searches approach permitted to incorporate a better balance of intensification/diversification than the single search approach.

Additionally, with the proposed memetic algorithm MLSM 17 best known solutions were improved. These new best known solutions were obtained when the MLSM algorithm was carried out using a time limit of 600 seconds. Table 4 shows these new best known solutions.

Table 3. The Wilcoxon test (memetic algorithm by Schiavinotto [5] and MLSM)

Execution time limit of 10 seconds										
Prom.	RandA I(100)	RandA I(150)	RandA I(200)	RandA I(500)	RandA II(150)	RandA II(200)	RandB	XLOLIB (150)	XLOLIB (250)	Spe c
N	25	25	25	25	25	25	20	39	39	7
N	7	24	25	25	4	12	20	39	33	3
R ⁺	23	282	246	325	10	55	210	749	558	4
R ⁻	5	18	79	0	0	23	0	31	3	2
R	26	219	236	236	NA	65	158	531	391	NA
Winner	none	SM	SM	SM	NA	none	SM	SM	SM	NA
Dif.Sig.?	No	Yes	Yes	Yes	NA	No	Yes	Yes	Yes	NA
Execution time limit of 600 seconds										
N	25	25	25	25	25	25	20	39	39	7
N	0	8	16	15	0	3	0	39	1	3
R ⁺	0	36	128	88	0	6	0	537	1	4
R ⁻	0	0	8	32	0	0	0	243	0	2
Ref	NA	33	107	95	NA	NA	NA	531	NA	NA
Winner	SM=MLSM	SM	SM	SM=MLSM	SM=MLSM	NA	S=MLSM	S	S=MLSM	NA
Dif.Sig.?	No	Yes	Yes	No	No	NA	No	Yes	No	NA

Table 4. New best solutions

Set	Instances	Old best solutions	New best solutions
Rand AI	N-t1d200.13	409234	409270
	N-t1d500.15	2411718	2412400
	N-t1d500.16	2416067	2416446
	N-t1d500.17	2401800	2402438
	N-t1d500.18	2421159	2421511
Spec	N-atp134	1796	1797
	N-atp163	2073	2075
XLOLIB	N - t75d11xx_150.mat	9642140	9643446
	N - tiw56r67_150.mat	2056347	2056446
	N - stabu2_250.mat	11500448	11500845
	N - stabu3_250.mat	11900315	11901939
	N - t59d11xx_250.mat	3841167	3841376
	N - t75d11xx_250.mat	25017059	25022475
	N - t75n11xx_250.mat	4524942	4525197
	N - tiw56n54_250.mat	2098726	2098877
	N - tiw56n62_250.mat	4142745	4143351
	N - tiw56n72_250.mat	11149706	11151094

6 Conclusions

In this work the linear ordering problem is approached. We propose a memetic algorithm that incorporates multiple local searches instead of a single local search. A combination of different local searches is evaluated for the process of improving the individuals.

The MLSM algorithm was tested on the UB-I instances, obtaining 0.12% in the average deviation from the best known solution, comparable with the memetic algorithm by Schiavinotto, and it outperforms the Tabu by Laguna by 48% in solution quality. The Wilcoxon non-parametric test shows that MLSM significantly outperforms the Tabu solution; but the memetic algorithm by Schiavinotto outperforms significantly the performance of MLSM in 6 of 10 cases for 10 seconds. In the test with 600 seconds MLSM was outperformed by SLSM only in 3 of 10 cases. The recovery of the MLSM performance could be explained because the multiple local searches approach permits to get a better control of the balance of intensification/diversification than the single local search approach. With the proposed memetic algorithm MLSM, 17 best known solutions were improved. We are currently applying the multiple local searches approach in other metaheuristics.

Acknowledgements. This work was supported by PROMEP, CONACYT and COTACYT. Also we want to thank Manuel Laguna and Abraham Duarte, for their support.

References

1. Laguna, M., Martí, R., Campos, V.: Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Computers and Operations Research* 26, 1217–1230 (1998)
2. Karp, R., Miller, R., Thatcher, J.: Reducibility among Combinatorial Problems. In: *Complexity of Computer Computation*, pp. 85–103. Plenum Press, New York (1972)
3. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co, New York (1975)
4. Leontief, W.W.: *Input-Output Economics*. Oxford University Press, Oxford (1986)
5. Schiavinotto, T., Stützle, T.: *Search Space Analysis of the Linear Ordering Problem*. Darmstadt University of Technology, Intellectics Group, Darmstadt (2004)
6. Martí, R., Reinelt, G., Duarte, A.: *A Benchmark Library and a Comparison of Heuristic Methods for the Linear Ordering Problem*. *Computational optimization and applications* (2010)
7. Moscato, P., Cota, C.: A Modern Introduction to Memetic Algorithms. In: Gendreau, M., Potvin, J.-Y. (eds.) *International Series in Operations Research & Management Science*, pp. 449–468. Springer, Heidelberg (2010)
8. Ong, Y.-S., Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. *IEEE Trans. Evol. Comput.* 8(2), 99–110 (2004)
9. Congram Richard, K.: *Polynomially Searchable Exponential Neighborhoods for Sequencing Problems in Combinatorial Optimization*. Faculty of Mathematical Studies, University of Southampton, UK (2000)

Performance Analysis of the Neighboring-Ant Search Algorithm through Design of Experiment*

Claudia Gómez Santillán^{1,2}, Laura Cruz Reyes¹, Eustorgio Meza Conde²,
Claudia Amaro Martínez¹, Marco Antonio Aguirre Lam¹,
and Carlos Alberto Ochoa Ortíz Zezzatti³

¹Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada (CICATA), Carretera Tampico-Puerto Industrial Alt., Km.14.5. Altamira,Tamps., México Phone: 018332600124
²Instituto Tecnológico de Ciudad Madero (ITCM). 1ro. de Mayo y Sor Juana I. de la Cruz s/n CP. 89440, Tamaulipas, México, Phone: (52) 833 3574820 Ext. 3024
³Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez Henry Dunant 4016, Zona Pronaf Cd. Juárez, Chihuahua, México C.P. 32310
cggs71@hotmail.com, lcruzreyes@prodigy.net.mx, emezac@ipn.mx, amarito@hotmail.com, marco@marcoaguirre.com.mx.

Abstract. In many science fields such as physics, chemistry and engineering, the theory and experimentation complement and challenge each other. Algorithms are the most common form of problem solving in many science fields. All algorithms include parameters that need to be tuned with the objective of optimizing its processes. The NAS (Neighboring-Ant Search) algorithm was developed to route queries through the Internet. NAS is based on the ACS (Ant Colony System) metaheuristic and SemAnt algorithm, hybridized with local strategies such as: learning, characterization, and exploration. This work applies techniques of *Design of Experiments* for the analysis of NAS algorithm. The objective is to find out significant parameters for the algorithm performance and relations among them. Our results show that the probability distribution of the network topology has a huge significance in the performance of the NAS algorithm. Besides, the probability distributions of queries invocation and repositories localization have a combined influence in the performance.

Keywords: Semantic Query Routing, Metaheuristic Algorithm, Parameter Setting, Design of Experiment, Factorial Design.

1 Introduction

With the advent of Internet and WWW, the amount of information available on the Web grows daily due to the opportunities to access, publish and manage resources. It has become common for Internet users to form online communities to share resources without a centralized management. These systems are known as unstructured peer-to-peer networks (P2P), and together with the underlying Internet are considered complex networks for its size and interconnectivity that constantly evolve. In these circumstances, global information recollection is not a feasible approach to perform

* This research was supported in part by CONACYT, DGEST and IPN.

queries on shared resources. A new approach to build query algorithms is to determine their behavior locally without using a global control mechanism. With this approach, we developed the NAS algorithm to perform text queries. Recently this type of query has been named Semantic Query Routing Problem (SQRP) [8].

NAS and many other algorithms have many parameters. Some researchers confirm [1][2][13], that the cost of not setting parameters with optimization algorithms, consumes a great amount of time and resources. It is well-known that parameter setting mainly depends on the problem characteristics. However, in adaptive complex systems, this is an open research [1][8][13]. The *Design of Experiment* (DOE) is an approach that can be adapted successfully to address the parameter setting problem for algorithms that find approximate solutions to optimization problems [1][2]. The objective in this work is to identify the SQRP main characteristics that affect the NAS algorithm performance.

2 Parameter Setting

The parameter setting can be classified into: parameter tuning and parameter control. *Parameter tuning* is the setting done before the algorithm runs that provides a global initial configuration. It evaluates the general performance of the algorithm but, it does not assure that the values for the parameters will be the best in each instant of the run of the algorithm. *Parameter Control* is the setting done during the run of the algorithm. This type of parameter setting supervises the local environmental changes and the current state of the algorithm to adapt locally the configuration to the local conditions [3][4].

The parameter tuning can be applied through three techniques: a) *by hand*, doing a sequence of experiments with different values of the parameters, and choosing the configuration with the best performance, b) *Meta-Evolution*, using an auxiliary metaheuristic algorithm to improve the performance of the main metaheuristic algorithm and c) *Design of Experiment* (DOE), this technique provides a great variety of statistics tests to make useful decisions [3][4].

2.1 Design of Experiments (DOE)

DOE is a statistics tool set, useful on making plans, running and interpreting an experiment, while searching for valid and impartial deductions. *An experiment* can be defined, as a planned test which introduces checked changes in the process or system variables, with the aim of analyzing changes that could happen over the system outputs. To apply the statistical approach in designing and analyzing an experiment, it is necessary to follow a general scheme of procedures. The items that Montgomery's work [5] suggested are: identifying and enunciating the problem, selecting the factors and levels, selecting response variables, choosing the experimental design, doing the experiment, analyzing statistical data, conclusions and recommendations.

2.2 Experiment Strategy: Factorial Design

In statistics, a *factorial experiment* is an experiment whose design consists of two or more factors, each with discrete possible values or "levels", and whose experimental

units take on all possible combinations of these levels across all such factors. A *factorial design* may also be called a *fully-crossed design*. Such an experiment allows studying the effect of each factor on the response variable, as well as the effects of interactions between factors on the response variable. The effect of the factor is defined as the change in the response variable, produced by a variation in the level of the factor. Its also known as *main effect* because it does reference to the primary factors in the experiment. Some experiments show that the difference in response between factors is not the same for all the factors and levels. When it happens, an interaction between them exists [5][6].

3 Ant Algorithms for Semantic Query Routing

Ant algorithms were inspired by the ants behavior, while searching for food. Because when they perform the search, each ant drops a chemical called pheromone which provides an indirect communication among the ants.

The basis of the ant algorithms can be found in a metaheuristic called ACO (Ant Colony Optimization) [7]. The ACO algorithm and its variants (e.g. ACS) were proposed to solve problems modeled as graphs. The ACS was designed for circuit-switched and packet-switched networks [8]. Each component of the network is represented by a *node* (or a vertex of the graph) and the interactions among themselves represent the *connections* (the edges of the graph). ACS needs to know information about all nodes in the network to select the destination node. However, in the SQRP the goal is to find one or more destination nodes for a query without having information from the complete network, requiring operating with local information.

3.1 Algorithm Neighboring-Ant Search

NAS 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: maximizes the found resources and minimizes the given steps. NAS guides the queries towards nodes that have better connectivity using the local structural metric *Degree Dispersion Coefficient* (DDC) [9]. The DDC measures the differences between the degree of a vertex and the degrees of its neighbors in order to minimize the hop count. Therefore, the more frequent is a query towards a resource, the better path is selected; that is to say, the degree of optimization of a query depends directly on its popularity. The NAS algorithm consists of two main phases [10].

Phase 1: the evaluation of results (lines 03-06 in pseudo code on Table 1). This phase implements the classic Lookahead technique. This is, ant k , located in a node s , queries to the neighboring nodes for the requested resource. If the resource is found, the result is retrieved. In case the evaluation phase fails, phase 2 is carried out.

Phase 2: the state transition (lines 07-17 in pseudo code on Table 1). This Phase selects through q , a neighbor node s . In the case that there is no node towards which to move (that is to say, the node is a leaf or all neighbor nodes had been visited) a hop backward on the path is carried out, otherwise the ant adds the node s to its path, and reduces TTL_k by one hop. The query process ends when the expected result has been satisfied or TTL_k is equal to zero, then the ant is killed indicating the end of the query.

Table 1. NAS algorithm pseudo code

```

01   for each query
02     for each ant
03       if Hits < maxResults and TTL > 0                               // Phase 1
04         if the neighbor from edge  $s_k$  has results                       // Lookahead strategy
05           append  $s_k$  to  $Path_k$  and  $TTL_k = TTL_k - 1$ 
06           Pheromone globalUpdate
07         else                                                            // Phase 2
08            $s_k$  = apply the transition rule with the DDC
09           if path does not exist or node was visited,
10             remove the last node from  $Path_k$ 
11           else,
12             append  $s_k$  to  $Path_k$  and  $TTL_k = TTL_k - 1$ 
13             Pheromone localUpdate
14           endif
15         endif
16       else
17         Kill ant
18       endif
19     endfor
20   endfor

```

4 Case Study: Determining SQRP Main Characteristics That Affect NAS Performance

4.1 Characteristics of SQRP

The problem of locating textual information in a 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 respond to the query [8]. The query traverses the network moving from the source node to a neighboring node and then, to a neighbor of a neighbor and so forth until locating the requested resource, or giving up in its absence. The challenge lies in the design of algorithms that navigate the internet in an intelligent and autonomous manner. In order to reach this goal, the NAS algorithm selects the next node to visit using information of the near-by nodes of the current node, for example, information on the local topology of the current node [10].

We considered three main characteristics of the SQRP and its probability distribution to study their impact in the performance of the NAS algorithm. First is the probability distribution of the node connections, which defines the topology of the network. Next, the probability distribution of the repository texts, which is the frequency of the distribution of the information contained in the nodes. Finally, the probability distribution of the queries, which defines how many times the same query, is repeated into the different nodes on the network. In order to simplify the experiment description, the following short names will be used respectively: topology, repository and query distributions. The most common distributions on Internet are: power law [11] and random [12].

4.2 Research Question and Hypothesis

The research question can be stated as: Does a set of characteristics of the SQRP affect the performance of the NAS algorithm? This question can be defined in the following research hypothesis.

Null Hypothesis H_0 : The distributions of the topology, repositories and queries of the SQRP, have no significant effect on the average performance of the NAS algorithm.

Alternative Hypothesis H_1 : The distributions of the topology, repositories and queries of the SQRP, have a significant effect on the average performance of the NAS algorithm.

4.3 Experimental Design

Hypothesis above require studying all combinations between the problem characteristic and probability distribution types. When it is needed to study the combined effect of two factors, the statistic literature recommends carrying out a full factorial 2^k experiment. In this kind of experiment, it is necessary to identify the factors with its levels, the variable responses and replicates. The impact of the factor levels is measured through the variable response in each replicate.

Two *factors* were studied: the probability distribution type and the problem characteristics. The *levels* of the first factor are: Scale free (SF) and Random (RM) and for the second factor the levels are: Topology, Repositories and Queries.

The *response1* variable is the average amount of discovered text (*hits*) and *response2* variable is the average on the number of visited nodes (*hops*). In the experiment the *replicates* are the different runs of the NAS metaheuristic.

4.4 Performing the Experiment

The NAS algorithm was implemented to solve SQRP instances. Each instance is formed by three files that correspond with the problem characteristics: Topology, Repositories and Queries. The generation of these files is explained below. In this experiment 2^3 instances were generated. These instances are the combinations of probability distributions of the characteristics of the problem. Each instance was replicated 7 times with NAS. The Topology and the Repositories were simulated in a static way, while the set of Queries was dynamic.

For the Topology, each no-uniform network with a *scale-free distribution* was generated using the model of Barabási et al. [11]. Similarly, the networks with *random distribution* were generated using the model of Erdős-Rényi [12]. In the *scale-free or power law* distribution, a reduced set of nodes has a very high degree connection and the rest of the nodes have a small degree. In the *random* distribution, all nodes have a degree very closed to the average degree. All networks were generated with 1,024 nodes and bi-directional links. This number of nodes was obtained from the recommendations of the literature [8][7].

For the repository texts R , the application is a distributed search machine. Each peer manages a local repository of resources, and offers its resources to other peers. The “topics” of the resources were obtained from ACM Computing Classification

System (ACMCCS) in which the database contains 910 topics. The R file was created with the two distributions before mentioned, but now the distributions are applied in how many times the topic was repeated. When the distribution was random the topics were duplicated 30 times in average. When the distribution was scale free, in each node was assigned a reduced set of topics with many copies, and the rest of the topics were assigned with few copies.

For the queries Q , each node has a list of possible resources to search. This list is limited by the total of topics of the ACM classification. During each algorithm step, each node has a probability of 0.1 to launch a query. The topics were selected randomly for each query. The probability distribution of Q determines how many times will be repeated a query in the network. When the distribution is random, the query is duplicated 30 times in average and, when the distribution is scale free, a reduced set of queries is much repeated and the rest of the queries is a few repeated.

Each simulation was run 10,000 steps. The average performance was calculated taking measures of the performance in progress each 100 steps. The performance was measure through two response variables. The *response1* variable measures the average number of hops; in each query the maximal number of *hops* was settled in 25. The *response2* variable measures the average of hits; the maximum number of hits was settled in 5.

The statistical experiment was performed with the package MINITAB. We created a full factorial 2^k design with 2 factors, 3 levels (k), 7 replicates, the significance level $\alpha = 0.05$, and the values of the two response variables for each instance. This factorial design was analyzed in order to determine the most important interactions of the factors, in other words, which combinations of the factors have the biggest effects on the performance. With this information, we ran the General Linear Model (GLM), which is based mainly on the ANOVA test. GLM was used to confirm, if the selected combinations of the factors are *statistically significant*. These results are described in the next section with plots of main effects and interactions.

5 Analyzing Statistics Results

The plots of the Figure 1 reveal that the type of probability distribution of the main factors has *relevant effect* on the average performance of NAS. The Topology is the factor that has the biggest effect on the average of the quality and efficiency of NAS. The other two factors (Repositories and Queries) show *irrelevant effect* on the average of the quality and efficiency of NAS. The size and direction of the graphs confirm this affirmation.

The plots of the Figure 2 reveal that the type of probability distribution of the factor interactions has *relevant effect* on the average performance of NAS. The interaction between *Repositories* and *Queries* has the biggest effect on the average of the quality and efficiency of NAS. The other two interactions show *irrelevant effect* on the average of the quality and efficiency of NAS. We demonstrated the statistical significance of the *relevant effect* with the GLM methodology.

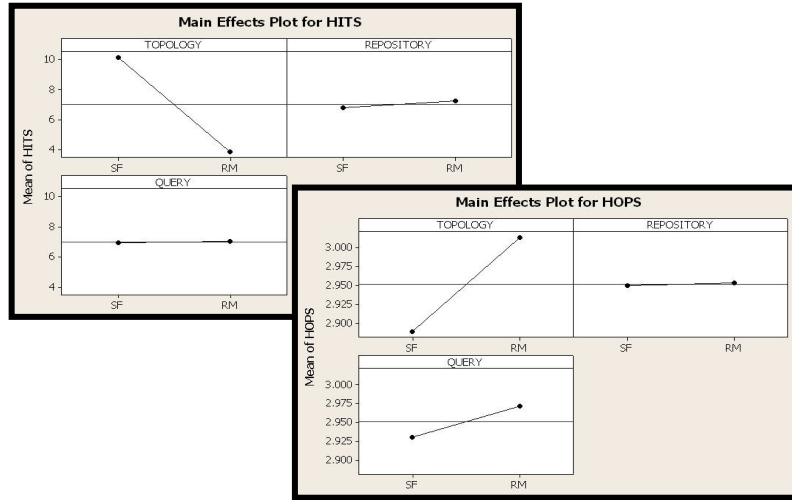


Fig. 1. Result of the Main effects: *response1* (HITS) and *response2* (HOPS)

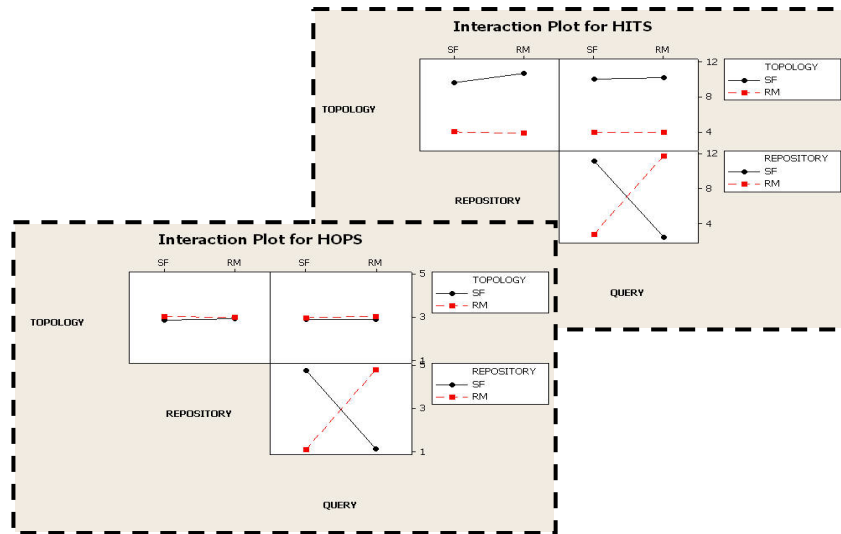


Fig. 2. Result of the characteristic interactions: *response1* (HITS) and *response2* (HOPS)

6 Conclusions and Future Works

A methodology based on Design of Experiment (DOE) was presented to answer the research question. The results obtained in the experimentation show that, the variability in the probability distributions of the SQRP characteristics (topology, query and repository) affects the performance of the NAS algorithm.

The experiment results showed that the topology distribution had a huge influence for the two response variables: number of resources founded for the queries and number of steps taken. We found out that the distributions of queries and repositories have strong relation ship in both response variables.

In addition, with the analysis of the results we identified which type of probability distribution is the most significant to the algorithm performance: a) The Scale-free, for the topological distributions, b) The random, for the query distributions and c) The scale-free for the repository distributions.

We are planning to use DOE, in order to analyze the control parameter of NAS algorithm. The objective of this future work is to find the best configuration to ensure the maximum NAS performance.

References

1. Birattari, M., Stutzle, T.: A Racing algorithm for Configuring Metaheuristics. *Artificial life*, 11–18 (2002)
2. Barr, R., Golden, J., Kelly, M.: Designing and Reporting Computational Experiments with Heuristics Methods. *Journal of Heuristics* 1, 9–32 (1995)
3. Michalewicz, Z.: *How to solve it: Modern Heuristics*. Springer, NY (2000)
4. Angeline, P.: Adaptative and Self-Adaptative Evolutionary Computations. *IEEE, Computational Intelligence*, 152–163 (1995)
5. Montgomery, D.C.: *Design and Analysis of Experiments*. John Wiley & Sons, New York (2001)
6. Manson, R.L., Gunst, R.F., James, L.H.: *Statistical Design and Analysis of Experiments with Applications to Engineering and Science*, 2nd edn. Wiley – Interscience, Chichester (2003)
7. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
8. Michlmayr, E.: *Ant Algorithms for Self-Organization in Social Networks*. Doctoral Thesis, Women's Postgraduate College for Internet Technologies (WIT), Institute of Software Technology and Interactive Systems, Vienna University of Technology (2007)
9. Ortega, R., et al.: Impact of Dynamic Growing on the Internet Degree Distribution. *Polish Journal of Environmental Studies* 16, 117–120 (2007)
10. Cruz-Reyes, L., et al.: NAS Algorithm for Semantic Query Routing System for Complex Network. In: *Advances in Soft Computing*, vol. 50, pp. 284–292. Springer, Heidelberg (2008)
11. Barabási, A.L., Albert, R., Jeong, H.: Mean-Field theory for scale-free random networks. *Physic A* 272, 173–189 (1999)
12. Erdős, P., Rényi, A.: On random graphs. I. *Publ. Math. Debrecen* 6, 290–297 (1959)
13. Adenso-Díaz, B., Laguna, M.: Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operation Research*, 99–114 (2004)

Solution to the Social Portfolio Problem by Evolutionary Algorithms

Gilberto Rivera, Claudia Gómez, Laura Cruz, Rogelio García, Fausto A. Balderas
Instituto Tecnológico de Cd. Madero, México.
riveragil@gmail.com, cggs71@hotmail.com, lcruzreyes@prodigy.net.mx,
rgarciardz@hotmail.com, tono.21@gmail.com

Eduardo R. Fernández
Universidad Autónoma de Sinaloa, México.
eddyf@uas.uasnet.mx

Fernando López
Universidad Autónoma de Nuevo León, México.
flopez65@gmail.com

Abstract. The formation of project portfolio is a multi-objective problem that has a high impact on public and private organizations, and has generally been addressed by evolutionary algorithms. They often seek an approximation of the Pareto front, and then the decision maker must choose an only solution from the set. This is not a difficult task when you have to select a solution from a small set evaluated in two or three objectives. But when the set of solutions grows, or the number of objectives increases, the choice is often a complicated process. It is necessary to present the decision maker only the subset of the Pareto front according to your preferences. This paper describes an optimization algorithm that steers the search process towards such solutions. The performance of the algorithm is evaluated with respect to the most related algorithm found in the state of the art.

Keywords: Project portfolio, evolutionary algorithm, preferences, decision maker, multi-objective optimization.

1 Introduction

Organizations to ensure their growth and permanence invest continuously and simultaneously on projects, however, confront the problem of having more projects than resources to implement them. One of the main tasks of the manager is to select projects that best meet the objectives of the company [1]. Incorrect decisions regarding the selection of projects have two main consequences:

1. Resources, generally limited, are wasted on projects that, although they may be good, not the most appropriate for the company.
2. The organization loses the benefits that could have gotten if it had invested in more suitable projects.

The selection of projects for a portfolio of social projects needs special treatment for the following reasons [2]:

1. The quality of projects is generally described by multiple criteria that are often in conflict.
2. Typically, requirements are not accurately known. Many concepts have no mathematical support for having entirely subjective nature.
3. Heterogeneity, or differences between the objectives of the projects, makes it difficult to compare.

These features of the *Social Portfolio Problem* represent a challenge for multi-objective optimization algorithms [3]. Moreover, although optimal solutions are found, the problem has not been completely resolved. Even the decision maker has the task of implementing just one of the alternatives presented. The decision maker will evaluate the alternatives according to her/his criteria and preferences.

2 Background

This section presents the theoretical foundations on which this paper is based.

2.1 Portfolio Problem

A *project* is a temporary, unique and unrepeatable process which pursues a specific set of objectives [4]. In this work, it is not considered that the projects can be broken down into smaller units such as tasks or activities. In other words, a project cannot be divided to run only a part, however, different versions of the same project can be proposed, each version may vary in amount of activity, time required and requested resources.

A *portfolio* consists of a set of projects that can be performed in the same period of time [4]. For this reason, projects in the same portfolio share available resources in the organization. They can complement each other, which is called synergy. Thus, it is not sufficient to compare the projects individually, but must compare groups of projects to identify what portfolio makes the greatest contribution to the organization objectives.

The proper selection of projects to integrate the portfolio, which will receive the organization's resources, is one of the most important decision problems for both public and private institutions [5, 6]. The main economic and mathematical models to the portfolio problem assume that there is a defined set of n projects, each project well characterized with costs and revenues, of which the distribution over time is known. The *Decision Maker* (DM) is responsible for selecting the portfolio that the company will implement [2].

2.2 Social projects

Social projects are characterized by objectives whose fulfillment benefits society. These objectives are generally intangible, such as social and scientific impact, as well as human resource training, among others, without regard to potential economic benefit as the main element of measure. In addition, the amount of desired objectives in these projects can be of several tens, depending on the level of detail and the conditions under which it is restricted.

It is also important to note that such projects are usually assigned to one area and region. The project area is mainly the social sector, e.g. education, public health, safety, scientific development, among others. The region is primarily concerned with the physical area that will benefit, for example by state, county, district, or similar. Thus, to form social portfolios should be considered:

1. No area/region monopolizes most of the budget, leaving remaining areas/regions with poor resources.
2. All areas/regions receive at least a minimal budget, ensuring its permanence and growth.

2.3 Multi-objective optimization

Real-world optimization problems are extremely complex with many attributes to evaluate and multiple objectives to optimize [7, 8]. The attributes correspond to quantitative values that describe the problem and are expressed in terms of decision variables. The objectives are the directions for improvement of the attributes and can be to maximize or minimize.

In many cases, due to the conflicting nature of attributes is not possible to obtain a single solution and therefore the ideal solution for a multi-objective problem (MOP) cannot be achieved because there is no one solution the problem. Typically, solving a MOP has a set of solutions that reached an aspiration level expected by the DM [7].

Figure 1 graphically illustrates the space of feasible solutions to a maximization problem (Figure 1a) and to a minimization problem (Figure 1b), with two objectives both. Note in Figure 1 that μ_4 can improve their performance in both objectives without leaving the feasible solution space, so μ_4 is not an optimal solution (it can still be improved). It can also see that μ_1 , μ_2 and μ_3 cannot improve one objective without harming the other. Therefore they are solutions that, although different, their performance is mathematically equivalent and cannot be overcome on both objectives simultaneously without leaving the feasible solution space [9]. This set of solutions is called the Pareto front, and find it is one of the main purposes of solving a MOP [10].

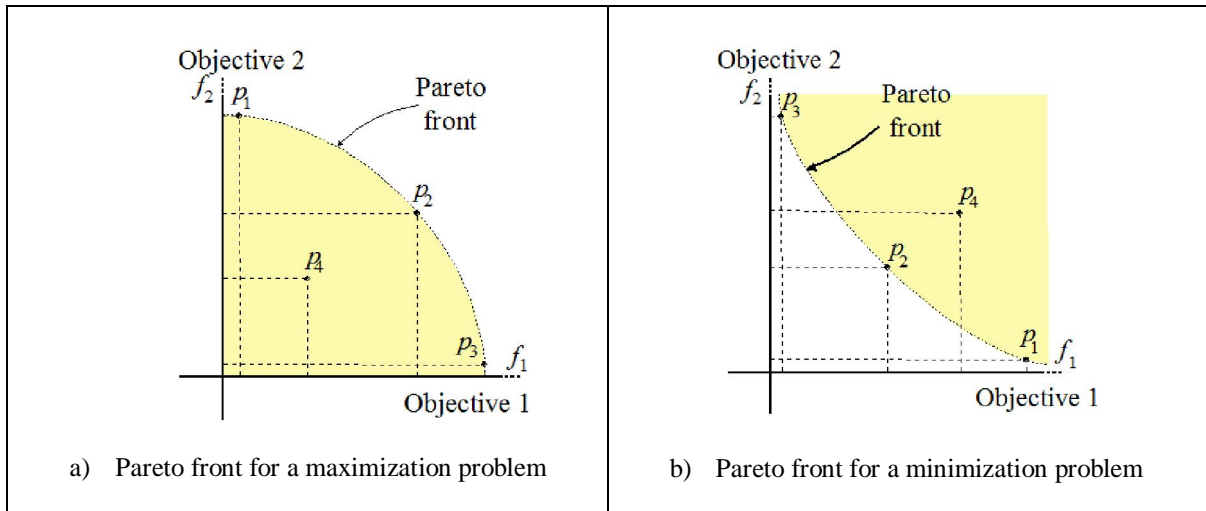


Fig. 1. Pareto front for optimization problems

But finding the Pareto front does not completely solve a MOP. Now the DM should choose a solution from the front, according to his/her own criteria. This is not a difficult task if you are managing two or three objectives. However, when the number of objectives increases, three major difficulties arise:

1. The capacity of algorithms for finding the Pareto front is rapidly degraded [11].
2. It becomes extremely difficult for the DM, and even impossible, to establish valid criteria for comparing solutions when there are conflicting objectives [9].
3. The size of the Pareto front can grow exponentially with respect to the number of objectives. This complicates the task of the DM to choose a solution [9].

A *compromise solution* is understood as a Pareto solution in which the objectives achieved acceptable values for the DM, and therefore could be selected. The *best compromise* is the compromise solution that meets best the preferences of the DM. Thus, the solution to a MOP is not only finding the Pareto front, but also to identify the best compromise.

Identify the Pareto front (or at least an approximation) has been commonly the task of multi-objective algorithms, leaving the identification of the best compromise to the user. However, a typical DM is capable of processing only at most five to ten pieces of information at once [12], thus being unable to identify the best compromise when he/she needs to compare sets of solutions of a MOP over five or nine objectives. To address this problem requires the creation of algorithms for MOP that show a set of solutions as small as possible, but without discarding those that the DM could choose as a final solution.

Since all Pareto solutions are mathematically equivalent, the DM should provide information about his/her preferences to MOP algorithms. Such information can be provided before or after to generate the Pareto solutions, or the process can be interactive, progressively consulting DM preferences [3].

2.4 Formal definition for multi-objective problem

Fernandez [3, 9] proposed a model to describe a multi-objective problem (to identify the best compromise). The model is based on solving Equation 1.

$$x^* = \min_{x \in O} \{ |S(O, x)| \} \quad (1)$$

where x^* is the best compromise, O is the solution space, $S(O, x)$ is a function that returns the set of solutions that exceed on preference to x , which can be defined as in Equation 2.

$$S(O, x) = \{ y \in O \mid \mathcal{P}(y, x) \} \quad (2)$$

where $\mathcal{P}(x, y)$ is a relational operator, called the operator of outranking, which indicates that the solution y is more preferable than solution x . It reads like " y outranks x " and indicates that y is at least as good as x . An outranking relation $\mathcal{P}(x, y)$ can be justified if at least one of the following conditions is true:

1. x dominates y
2. $(x, y) \wedge (y, x) \geq 0.5$.
3. $(x, y) \wedge (0.5 - (y, x)) \wedge ((x, y) - (y, x)) \geq \theta$.

The function (x, y) returns a value in the range $[0, 1]$, indicating how much the DM is agree with the statement " x is at least as good as y ". This value can be calculated using some proven methods such as ELECTRE-III [13, 14] and PROMETHEE [15].

The parameter $\theta \in [0, 1]$ is a threshold that indicates what the minimum magnitude of (x, y) to consider credible the statement " x is at least as good as y ". In general, it is considered that $\theta = 0.5$. The parameter δ is a threshold that indicates what should be the minimum difference between (x, y) and (y, x) for believing that one of them is significantly higher than the other.

The operator of outranking $\mathcal{P}(x, y)$ serves to calculate, using $S(O, x)$, the number of preferred solutions with respect to a given solution. When $S(O, x) = \emptyset$, means that no exist, or not found (in the case of approximate algorithms), no other solution that is preferred over x .

The set of solutions did not outranked in O is a subset of the Pareto front. For a solution $x \in O$, the outranking set is defined as $S(O, x) = \{y \in O, | \mathcal{P}(y, x) \}$. The cardinal of this set, $|S(O, x)|$, is an entire function which depends on x , and if it is equal to zero, x should be presented as a solution that is probably the best compromise.

Figure 2 illustrates how the search process is addressed when incorporating information about the preferences of decision maker. As shown in Figure 2b, the search is directed to privileged regions of the Pareto front, those where $|S(O, x)| = 0$. These regions contain the solutions that were not outranked by any other solution in O . The set of all solutions contained in these regions is called *non-outranked front*. Figure 2a illustrates how a multi-objective algorithm without a preferential model is not directed toward the non-outranked front, but towards the entire Pareto front.

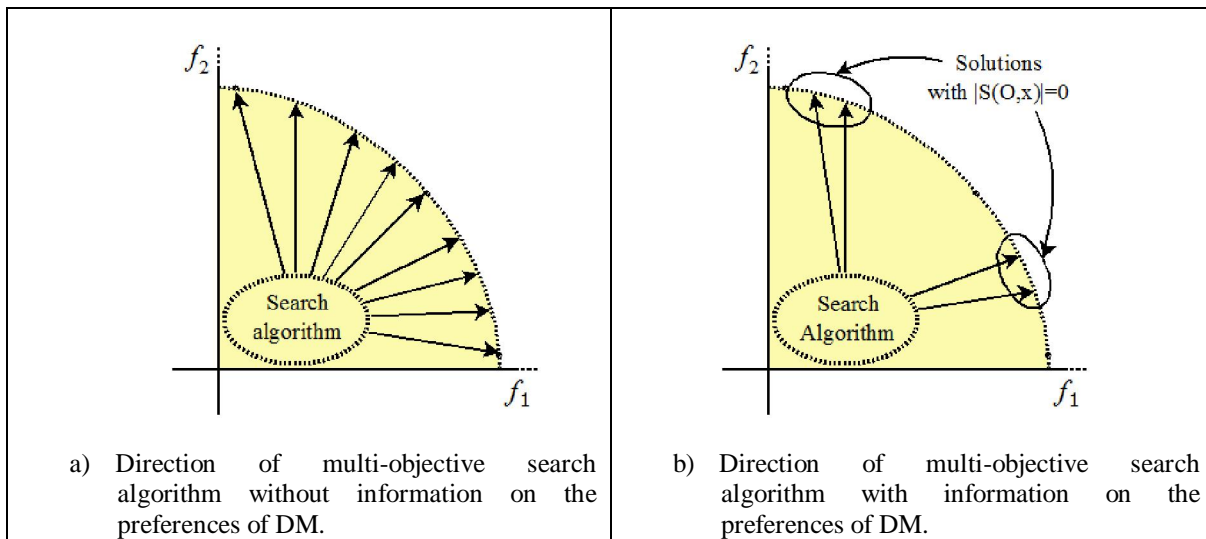


Fig. 2. Differences between an algorithm with DM's preferences information and an algorithm without preferences information

2.5 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEA) have become a popular technique for solving multi-objective problems [7]. The MOEA are very attractive for solving multi-objective problems because they treat simultaneously a set of possible solutions, which allows obtaining an approximation of the Pareto front in a single execution. Thus, using MOEA, the DM does not need to do a series of optimizations for each objective, as is usually done in the methods of operations research [16, 17].

However, a limitation on the MOEA is the fact that only involves the process of finding a solution set without considering the most important aspect, the decision process. Most current approaches to MOEA are focused on finding an approximation to the optimal set of Pareto front, however, identify the best compromise has usually been omitted.

2.6 Hyper-heuristics

The term hyper-heuristic was first used in 1997 by Jorg Denzinger, Matthias Fuchs and Marc Fuchs [18]. They used the term to describe a protocol that selects and combines various methods of artificial intelligence. Later, in 2000, Cowling and Soubeiga used the term hyper-heuristic to describe the idea of a "heuristic that selects heuristics" in the context of combinatorial optimization [19].

The hyper-heuristics are simple heuristics or high level, that given a particular problem and a number of *Low Level Heuristics* (LLH) that act on the problem, select and apply the most appropriate LLH in each decision point. The way why the solution space is managed by Hyper-Heuristics is shown in Figure 3, where we can see that a hyper-heuristic runs a series of LLH, which modify the decision variables of an optimization problem, causing a change in the function objective. In the example in Figure 3, the problem has two decision variables (x_1 and x_2) and three objective functions (f_1 , f_2 and f_3).

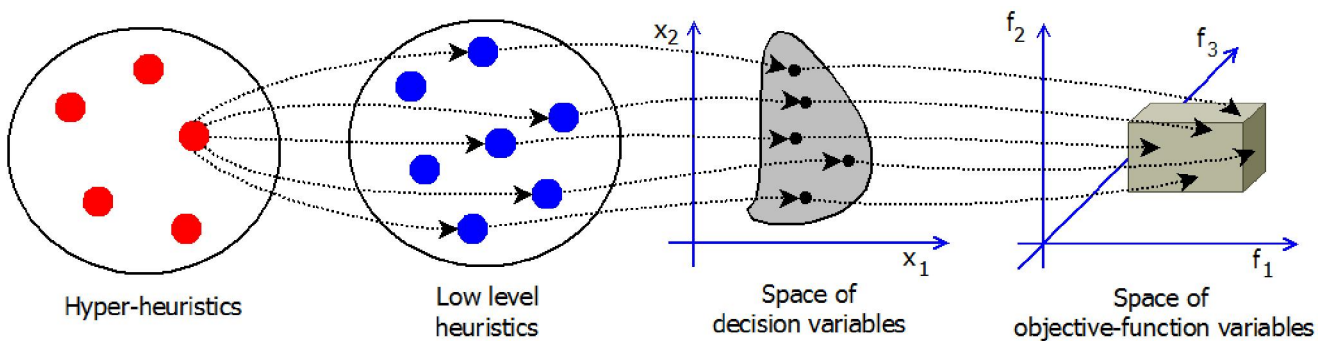


Fig. 3. Space where hyper-heuristics operate

Usually, meta-heuristics work directly with the solution of the problem by modifying the solution directly, while hyper-heuristics modify the solution indirectly using the available LLH. The hyper-heuristics have a level of abstraction and generality higher than most current meta-heuristics. It's called *High Level Heuristics* (HLH) to heuristics that controls and directs the HBLH, and can range from simple search heuristics to more complex meta-heuristics, such as a genetic algorithm [20].

2.7 Hyper-heuristics taxonomy

Burke [21] proposes two criteria to classify hyper-heuristics: Depending on the feedback and depending to the nature of the search space. These taxonomies are shown in Figure 4. According to feedback, the hyper-heuristics may be:

1. Without learning: do not use of feedback during the search process.
2. With off-line learning: learn, from a training instance set, a method that could be generalized to unseen instances.
3. With online learning: learn while solving an instance of the problem.

Depending on the nature of the search space, the hyper-heuristics include: selection heuristic and generation heuristic. The first type has a set of heuristics that solve fully or partially the central problem. The aim is to discover the sequence of how to apply these heuristics to solve the problem efficiently. In the second category are the heuristics that generate heuristics. This kind of hyper-heuristics has a set of constructive heuristics that solve the central problem. First starts with an empty solution and selects intelligently the constructive heuristics for improving gradually the solution. This process continues until to have a complete solution to the central problem.

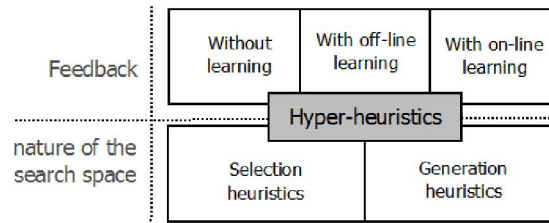


Fig. 4. Kinds of hyper-heuristics

3 State of the art

SS-PPS [4], *Scatter Search for Project Portfolio Selection*, is a scatter search algorithm that addresses the portfolio problem. In [4] not only is proposed the algorithm, but also a mathematical model for portfolio selection problem. SS-PPS has basically two stages, the first one generates a set of initial points efficiently using tabu search, and the second phase improves the initial set by scatter search. The algorithm is tested on 76 instances randomly generated, showing a better performance than SPEA2 [21], a multi-objective highlighted algorithm in the literature in MOP. Each instance contains between 10 and 60 candidate projects evaluated in two, four and six objectives, depending on the instance. SS-PPS searches a solution set as scattered as possible in the Pareto front, leaving the final selection of the portfolio to DM.

Doerner et. Al [22] proposed P-ACO (*Pareto Ant Colony Optimization*), an algorithm based on the known ant colony meta-heuristic for generating the Pareto front for the most efficient portfolios. Each ant in the colony generates a candidate portfolio, and the amount of pheromone deposited by the ant is inversely proportional to the number of solutions that dominate it. The algorithm stores the solutions that have never been dominated, which form an approximation of the Pareto front. This algorithm is tested on 30 test instances evaluated with nine restrictions. Each instance has between 20 and 30 projects evaluated in 4, 6, 8 or 10 objectives (depending on the instance type). P-ACO during the search does not incorporate the preferences of the DM, so that the final selection of the portfolio depends on the effort that the DM spent for finding a solution according to his preferences in the approximation of Pareto.

Reiter [23] proposes APS (*Adaptive Pareto Sampling*), an algorithm that uses a sampling approach for estimation of the Pareto front, based on Monte Carlo simulation method. APS is quick to reduce the computational effort without losing accuracy. It does this through a metric that guides the simulation process called *Importance Sampling*. Reiter compares APS with P-ACO over one hundred test instances, exceeding the performance and runtime. Each project is evaluated on two objectives only. Although APS does not search the non-outranking front, includes techniques to facilitate the dispersion of the solutions in the Pareto front.

NOSGA [9], unlike the works mentioned above, is designed for solving the SPP model formulated in Section 2.4 by Fernandez [3.9]. NOSGA (*Non-Outranked-Sorting Genetic Algorithm*) is based on the well-known algorithm NSGAI [24], the main change being the addition of the outranking operator in order to obtain the best solutions according to the preferences of DM, and not only the Pareto front as NSGA-II does it.

Unlike NSGA-II, NOSGA is not interested in finding a uniform Pareto front, but looks the non-outranking front. It is noteworthy that NOSGA largely outperforms NSGA-II, according to experimental evidence, for the SPP on a set of instances of three and nine objectives. The main differences between NOSGA and NSGA-II are:

1. The use of fitness of individuals, instead of the values in the objective functions.
2. The order based on dominance of the Pareto front is replaced by the strict preference provided by the outranking operator. Thus, zero-order individuals (that have not been outranked by any other solution found) have priority in the crossover and survival to the next generation.

Table 1 presents a comparison among the works presented in this section, where it is seen that NOSGA is the research work with greater affinity to proposal of this article. This is seen in the dimensionality (number of objectives) and that NOSGA takes into account the preferences of the DM during the search for the best portfolio. For this reason, NOSGA is considered as the algorithm which will contrast the performance of the technique presented in this document.

Table 1. Related works to multi-objective portfolio problem

Algorithm	Basic technique	Objectives	Consider the DM's preferences
SS-PPS	Scater search and tabu search	2, 4 and 6	
P-ACO	Ant colony optimization	4, 6, 8 y 10	
APS	Monte carlo simulation	2	
NOSGA	Genetic algorithm	6 and 9	ü

4 Proposed algorithm

This section describes in detail the hyper-heuristic algorithm used for SPP developed by Garcia [6], called *Hyper-Heuristic Genetic Algorithm for Social Portfolio Problem* (HHGA-SPP).

In HHGA-SPP, the chromosome of individuals represents the sequence of how LLH are executed. The LLH will interact with the solution of the problem by adding, removing or exchanging projects. None of these low-level heuristics can solve the SPP itself. The genetic algorithm never directly manipulates the solutions even if individuals have in their associated solutions to the problem. The LLH are responsible for modifying the SPP solutions to generate new solutions.

4.1 Solution representation

For the SPP, the representation of the solution is a binary array of size n where each cell represents a project, assigning 0 if the project does not receive support and 1 if the project receives support. Figure 5 shows an example of the representation of the solution for an instance of SPP with 8 projects. In the example in Figure 5, projects 1, 3, 4, 5 and 8 will be supported.

1	0	1	1	1	0	0	1
1	2	3	4	5	6	7	8

Fig. 5. Representation of a SPP solution

4.2 Low Level Heuristics

The LLH for SPP were a total of seven:

1. Random change: Change a project for another at random.
2. Generate random solution: Generate a new random solution.
3. Left change: Change a randomly-selected project by the first project that makes the solution feasible, the projects are selected from left to right.
4. Right change: Change a randomly-selected project by the first project that makes the solution feasible, the projects are selected from right to left.
5. Change \mathcal{R} region: Take at random one project from each region and replaces it with another project in the same region.
6. Change \mathcal{A} area: Take at random one project from each area and replaces it with another project in the same area.
7. Opposite change: Select at random four projects, if a project is in the portfolio is deleted, otherwise it will chosen.

4.3 High Level Heuristics

The genetic algorithm used as basis for creating the Hyper-Heuristic is the most basic of genetic algorithms. The number of generations is a static parameter, individuals always cross at a cut point and the selection of individuals for the crossover is at random. The mutation rate of HHGA-SPP is 25% and the mutation is to exchange one LLH for another randomly selected. In addition, an elitism of 10% of the population is incorporated.

Each individual in the genetic algorithm represents a sequence of LLH. Figure 6 illustrates a chromosome with eight genes of an individual in HHGA-SPP. In each gene is stored the number corresponding to a LLH (see low-level heuristics in Section 4.3) and each individual is associated with the resulting SPP solution by applying the corresponding sequence of HBN.

1	3	6	2	7	4	3	5
1	2	3	4	5	6	7	8

Fig. 6. Representation of a chromosome in HHGA-SPP

5 Experimental results

This section describes the experiments carried out to evaluate the performance of the proposed algorithm.

5.1 Experimental environment

The following configuration corresponds to the experimental conditions that are common to the tests described in this chapter:

1. Software. Operating system, Linux Ubuntu; programming language, C; compiler, CGG 4.4.4.
2. Hardware. Computer equipment dual-processor Xeon (TM) CPU 3.06 GHz in parallel and 4 GB RAM.
3. Instances. The 30 instances used for this study are randomly generated, consisting of 20 projects evaluated in four objectives.
4. Performance measurement. Performance is measured according to the number of *Non-Outranked Solutions* (NOS) found by the algorithm.
5. Parameter setting. The parameter setting used to evaluate the performance of the algorithm is that proposed by García [6].

5.2 Algorithm performance

The purpose of this section is to verify the quality of the solutions obtained by HHGA-SPP. A comprehensive search algorithm was run to find optimal solutions for each instance. The results of HHGA-SPP are plotted in Figure 7, where averages of 30 runs of algorithm are showed. In this experiment, a standard deviation of 0.77 among the solutions was observed.

Figure 7 shows that the larger is the optimal set of NOS in the instance, the harder is to find it by HHGA-SPP. The Hyper-heuristic found the optimal for 13 of 30 instances, showing an average error of 26% compared to optimal solutions.

5.3 Performance comparison with related work

The purpose of this section is to verify that HHGA-SPP has a competitive performance with respect to the algorithm NOSGA. The results are plotted in Figure 8, in which the order of appearance of instances is increasing with respect to the optimal number of NOS.

As shown in Figure 8, both algorithms achieve the same performance in 15 test instances, in 4 instances HHGA outperforms NOSGA a 20% on average performance, and in 11 instances NOSGA outperforms HHSGA-SPP a 17%.

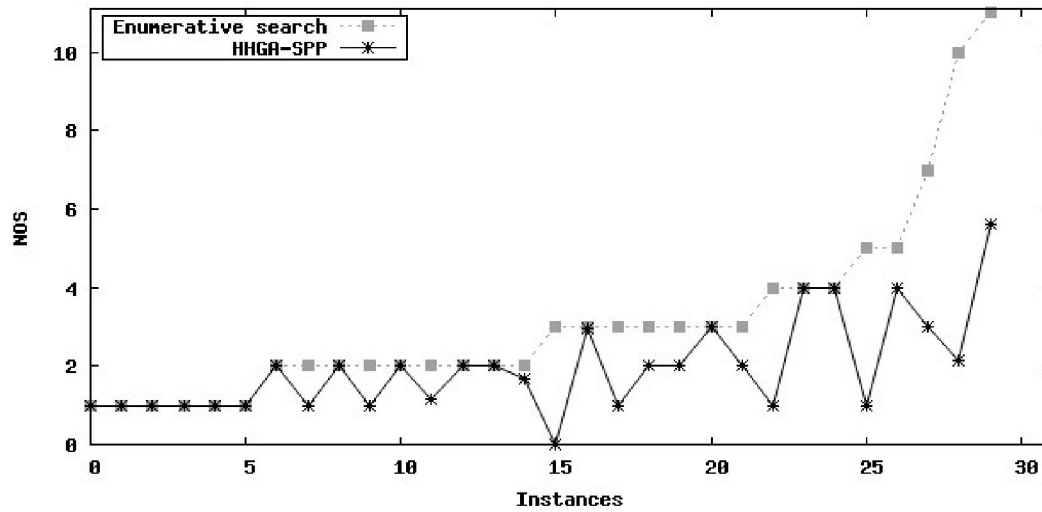


Fig. 7. Comparison between HHGA and an enumerative search

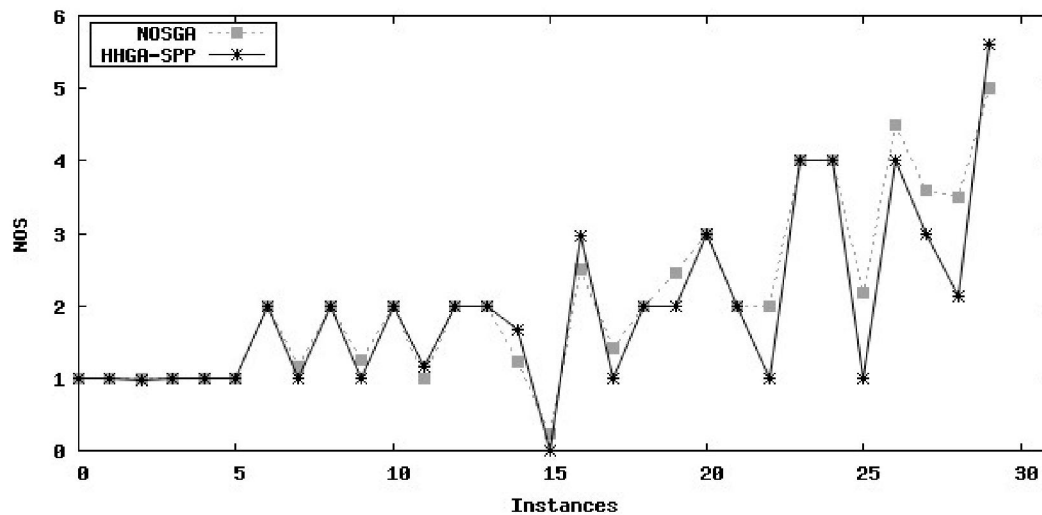


Fig. 8. Comparison between NOSGA and HHGA-SPP.

6 Conclusions and future work

This article provides a solution to the Social Portfolio Problem by Evolutionary Algorithms, creating an hyper-heuristic algorithm called HHGA -SPP, that includes the model of Fernandez et al [3, 9], through which DM’s preferences are modeled during the search process.

The results presented show that HHGA-SPP has a competitive performance. In the first experiment HHGA-SPP was compared with an exhaustive search, achieving an average error of 26% compared to optimal solutions. Next was compared with NOSGA, achieving an average error of 3%.

The following future works could be addressed in further study of the topic:

1. Include new LLH that enhance the intensification and diversification capabilities.
2. Analyze the algorithm performance on real instances.
3. Include SPP variants that did not considered in this work.

References

1. Ghasemzadeh, F., Archer, N., and Iyogun, P.: A zero-one model for project portfolio selection and scheduling. *Journal of the Operational Research Society*, Vol. 50, No. 7 (1999) 745-755.
2. Fernández, E. and Navarro, J.: A genetic search for exploiting a fuzzy preference model of portfolio problems with public projects. *Annals OR*, Vol. 117 (2002) 191-213.
3. Fernández, E., López, E., Bernal, S., Coello Coello, C.A., and Navarro, J.: Evolutionary multiobjective optimization using an outranking-based dominance generalization. *Computers & Operations Research*, Vol. 37, No. 2 (2010a) 390-395.
4. Carazo, A.F., Gómez, T., Molina, J., Hernández-Díaz, A.G., Guerrero, F.M., and Caballero, R.: Solving a comprehensive model for multiobjective project portfolio selection. *Computers & Operations Research*, Vol. 37, No. 4 (2010) 630-639.
5. Castro, M.: Development and implementation of a framework for I&D in public organizations. Master's thesis, Universidad Autónoma de Nuevo León (2007).
6. García, R.: Hyper-Heuristic for solving social portfolio problem. Master's Thesis, Instituto Tecnológico de Cd. Madero (2010).
7. Coello Coello, C.A., Lamont, G.B., and Van Veldhuizen, D.A.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer, 2nd edition (2007).
8. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Series Artificial Intelligence, Springer-Verlag, New York, 3ra. Edition (1996).
9. Fernández, E., López, E., López, F., and Coello Coello, C.A.: Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method. *Information Sciences*, Vol. 181, No. 1 (2010b) 44-56.
10. Durillo, J.J., Nebro, A.J., Coello Coello, C.A., García-Nieto, J., Luna, F., and Alba, E.: A study of multi-objective metaheuristics when solving parameter scalable problems. *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 4 (2010) 618-635.
11. Wang, Y. and Yang, Y.: Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, Vol. 179, No. 12 (2009) 1944-1959.
12. Marakas, G.M.: *Decision Support Systems in the 21th Century*. Prentice Hall, New Jersey (1999).
13. Roy, B.: Reading in Multiple Criteria Decision Aid, chapter The Outranking Approach and the Foundations of ELECTRE methods, Springer-Verlag (1990) 155-183.
14. Roy, B. and Slowinski, R.: Handling effects of reinforced preference and counter-veto in credibility of outranking. *European Journal of Operational Research*, Vol. 188, No. 1 (2008) 185-190.
15. Brans, J. and Mareschal, B.: Multiple Criteria Decision Analysis: State of the Art Surveys, chapter PROMETHEE Methods, pages 163-190. Springer-Verlag, New York (2005).
16. Peñuela, C. and Granada, M.: Optimización multiobjetivo usando un algoritmo genético y un operador elitista basado en un ordenamiento no dominado (NSGA-II). *Scientia Et Technica*, Vol. 8, No. 35 (2007) 175-180.
17. Osyczka, A.: Multicriteria optimization for engineering design. En *Design Optimization*, Academic Press (1985) 155-183.
18. Cowling, P., and Soubeiga, E.: Neighborhood structures for personnel scheduling: A summit meeting scheduling problem.
19. Soubeiga, E.: Development and Application of Hyperheuristics to Personnel Scheduling. PhD thesis, The University of Nottingham (2003).
20. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A Classification of Hyper-Heuristic Approachesby, In *Handbook of Meta-heuristics 2nd Edition* (eds. M.Gendreau and J-Y.Potvin), Springer (2010) 449-468.
21. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Technicalreport103, Computer Engineering and Networks Laboratory(TIK),Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001).
22. Doerner, K., Gutjahr, W.J., Hartl, R., Strauss, C., and Stummer, C.: Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals OR*, Vol. 131 (2004)79-99.
23. Reiter, P.: Metaheuristic Algorithms for Solving Multi-objective/Stochastic Scheduling and Routing Problems. Tesis Doctoral, University of Wien (2010).
24. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester-New York-Weinheim-Brisbane-Singapore-Toronto (2001):