Nino Ruiz, Elias D.; Nieto Parra, Henry; Chinchilla Camargo, Anangelica Isabel
EVOLUTIONARY ALGORITHM BASED ON SIMULATED ANNEALING FOR THE MULTI-OBJECTIVE
OPTIMIZATION OF COMBINATORIAL PROBLEMS
International Journal of Combinatorial Optimization Problems and Informatics
Morelos, México

Available in: http://www.redalyc.org/articulo.oa?id=265229633005
EVOLUTIONARY ALGORITHM BASED ON SIMULATED ANNEALING FOR THE MULTI-OBJECTIVE OPTIMIZATION OF COMBINATORIAL PROBLEMS

EMSA: Hybrid Metaheuristic based on Genetic Algorithms, Simulated Annealing and Deterministic Swapping

Elias D. Nino Ruiz\textsuperscript{1,2}, Henry Nieto Parra\textsuperscript{3} and Anangelica Isabel Chinchilla Camargo\textsuperscript{3}

\textsuperscript{1}Department of Computer Sciences, Virginia Tech, Blacksburg, United States of America
\textsuperscript{2}Department of Computer Sciences, Universidad del Norte, Barranquilla, Colombia
\textsuperscript{3}Department of Industrial Engineering, Universidad del Norte, Barranquilla, Colombia
enino@vt.edu, enino@uninorte.edu.co, \{hnieto, achinchilla\}@uninorte.edu.co

Keywords: Combinatorial Optimization, Genetic Algorithms, Simulated Annealing, Multi-objective Optimization

Abstract: This paper states a novel hybrid-metaheuristic based on the Theory of Deterministic Swapping, Theory of Evolution and Simulated Annealing Meta-heuristic for the multi-objective optimization of combinatorial problems. The proposed algorithm is named EMSA. It is an improvement of MODS algorithm. Unlike MODS, EMSA works using a search direction given through the assignation of weights to each function of the combinatorial problem to optimize. Also, in order to avoid local optimums, EMSA uses crossover strategy of Genetic Algorithm. Lastly, EMSA is tested using well know instances of the Bi-Objective Traveling Salesman Problem (TSP) from TSPLIB. Its results were compared with MODS Metaheuristic (its precessor). The comparison was made using metrics from the specialized literature such as Spacing, Generational Distance, Inverse Generational Distance and Non-Dominated Generation Vectors. In every case, the EMSA results on the metrics were always better and in some of those cases, the superiority was 100%.

1 INTRODUCTION

Combinatorial optimization is a branch of optimization. Its domain is optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution (Yong-Fa and Ming-Yang, 2004). In this field it is possible to find a lot of problems denominated NP Hard such as Multi-depot vehicle routing problem (A. and Wang, 2005), delivery and pickup vehicle routing problem with time windows (Wang and Lang, 2008), multi-depot vehicle routing problem with weight-related costs (Fung et al., 2009), Railway Traveling Salesman Problem (Hu and Raidl, 2008), Heterogeneous, Multiple Depot, Multiple Traveling Salesman Problem (Oberlin et al., 2009) and Traveling Salesman with Multi-agent (Wang and Xu, 2009).

One of the most classical problems in the Combinatorial Optimization Field is the Traveling Salesman Problem (TSP), it has been analyzed for years(Sauer and Coelho, 2008) either in a Mono or Multi-objective way. It is defined as follows: “Given a set of cities and a departure city, visit each city only once and go back to the departure city with the minimum cost”. Basically, that is mean, visiting each city once, to find an optimal tour in a set of cities, an instance of TSP problem can be seen in figure 1. Formally, TSP is defined as follows:

\begin{equation}
\min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \cdot X_{ij}
\end{equation}

Subject to:

\begin{equation}
\sum_{j=1}^{n} X_{ij} = 1, \forall i = 1, \ldots, n
\end{equation}

\begin{equation}
\sum_{j=1}^{n} X_{ij} = 1, \forall j = 1, \ldots, n
\end{equation}

\begin{equation}
\sum_{i \in \kappa} \sum_{j \in \kappa} X_{ij} \leq |\kappa| - 1, \forall \kappa \subset \{1, \ldots, n\}
\end{equation}

\begin{equation}
X_{ij} = 0, 1, \forall i, j
\end{equation}

Where $C_{ij}$ is the cost of the path $X_{ij}$ and $\kappa$ is any nonempty proper subset of the cities $1, \ldots, m$. (1) is
the objective function. The goal is the optimization of the overall cost of the tour. (2), (3) and (5) fulfills the constraint of visiting each city only once. Lastly, Equation (4) set the subsets of solutions, avoiding cycles in the tour.

Figure 1: TSP instance of ten cities

TSP has an important impact on different sciences and fields, for instance in Operations Research and Theoretical Computer Science. Most problems related to those fields, are based in the TSP definition. For example, problems such as Heterogeneous Machine Scheduling(Kim and Lee, 1998), Hybrid Scheduling and Dual Queue Scheduling(Shah et al., 2009), Project Management(de Pablo, 2009), Scheduling for Multichannel EPONs(McGarry et al., 2008), Single Machine Scheduling(Chunyue et al., 2009), Distributed Scheduling Systems(Yu et al., 1999), Relaxing Scheduling Loop Constraints(Kim and Lipasti, 2003), Distributed Parallel Scheduling(Liu et al., 2003), Scheduling for Grids(Huang et al., 2010), Parallel Scheduling for Dependent Task Graphs(Mingsheng et al., 2003), Dynamic Scheduling on Multiprocessor Architectures(Hamidzadeh and Atif, 1996), Advanced Planning and Scheduling System(Chua et al., 2006), Tasks and Messages in Distributed Real-Time Systems(Manimaran et al., 1997), Production Scheduling(You-xin et al., 2009), Cellular Network for Quality of Service Assurance(Wu and Negi, 2003), Net Based Scheduling(Wei et al., 2007), Spring Scheduling Co-processor(Niehaus et al., 1993), Multiple-resource Periodic Scheduling(Zhu et al., 2003), Real-Time Query Scheduling for Wireless Sensor Networks(Chipara et al., 2007), Multimedia Computing and Real-time Constraints(Chen et al., 2003), Pattern Driven Dynamic Scheduling(Yingzi et al., 2009), Security-assured Grid Job Scheduling(Song et al., 2006), Cost Reduction and Customer Satisfaction(Grobler and Engelbrecht, 2007), MPEG-2 TS Multiplexers in CATV Networks(Jianghong et al., 2000), Contention Awareness(Shannugapriya et al., 2009) and The Hard Scheduling Optimization(Niño et al., 2010b) had been derived from TSP. Although several algorithms have been implemented to solve TSP, there is no one that optimal solves it.

This paper is structured as follows: Section 2 shows important definitions to understand the multi-objective combinatorial optimization field and the metaheuristic approximation. Section 3 and 4 discuss an hybrid evolutionary metaheuristic to optimize multi-objective combinatorial problems. Finally, Section 5 and 6 discuss the Experimental Results of the proposed Algorithm using Multi-objective Metrics from the specialized literature.

2 PRELIMINARIES

2.1 Multiobjective Optimization

The Multi-objective optimization consists in two or more objectives functions to optimize and a set of constraints. Mathematically, the Multi-objective Optimization model is defined as follows:

\[
\text{optimize } F(X) = \{f_1(X), f_2(X), \ldots, f_n(X)\} \quad (6)
\]

Subject to:

\[
H(X) = 0 \quad (7)
\]

\[
G(X) \leq 0 \quad (8)
\]

\[
X_l \leq X \leq X_u \quad (9)
\]

Where \(F(X)\) is the set of objective functions, \(H(X)\) and \(G(X)\) are the constraints of the problem. Lastly, \(X_l\) and \(X_u\) are the bounds for the set of variables \(X\).

Unlike to Mono-objective Optimization, Multi-objective Optimization deal with searching a set of Optimal Solutions instead of a Optimal Solution.

2.2 Genetic Algorithms

Genetic Algorithms are Algorithms based on the Theory of Natural Selection(Wijkman, 1996). Thus, Genetic Algorithms mimics the real behavior of real evolutionary systems(Fisher, 1930) through three basic steps: Given a set of Initial Solutions \(S\)

**Step 1. Selection.** Select solutions from a population. In pairs, select two solutions \(x, y \in S\)

**Step 2. Crossover.** Cross the selected solutions avoiding local optimums.

**Step 3. Mutation.** Perturbs the new solutions found for increasing the population. The perturbation can be done according to the representation of the solution. In this step, good solutions are added to \(S\).

Figure 2 shows the basics steps of a Genetic Algorithm.
The most known Genetic Algorithms from the literature(Dukkipati and Narasimha Murty, 2002) are the Non-Dominated Sorting Genetic Algorithm(Deb et al., 2002) (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2(Zitzler et al., 2001; Zitzler et al., 2002) (SPEA 2). NSGA-II uses a no-dominated sort for sorting the solutions in different Pareto Sets. Consequently, it demands a lot of time, but it allows a global verification of the solutions for avoiding the Local Optimums. On the other hand, SPEA 2 is an improvement of SPEA. The difference with the first version is that SPEA 2 works using strength for every solution according to the number of solutions that it dominates. Consequently, at the end of the iterations, SPEA 2 has the non dominated solutions stronger avoiding Local Optimums. SPEA 2 and NSGA-II have been implemented to solve a lot of problems in the Multiobjective and Combinatorial Optimization field. For instance, problems such as Pattern-recognition based Machine Translation System(Sofianopoulos and Tambouratzis, 2011), Tuning of Fuzzy Logic controllers for a heating(Gacto et al., 2011), Real-coded Quantum Clones(Xiawen and Yu, 2011), Optimization Problems with Correlated Objectives(Ishibuchi et al., 2011), Production Planning(Yu et al., 2011), Optical and Dynamic Networks Designs(Araujo et al., 2011; Wismans et al., 2011), Benchmark multi-objective optimization(McClymont and Keedwell, 2011) and Vendor-managed Inventory(Azuma et al., 2011) have been solved using SPEA and NSGA-II.

2.3 Simulated Annealing inspired Algorithms

Simulated Annealing(Kirkpatrick et al., 1983) is a generic probabilistic metaheuristic based in the Annealing in Metallurgy. It explores the neighborhood of solutions being flexible with ill solutions. That is mean, accepting bad solutions as well as good solution, but only in the first iterations, when the temperature is high. The acceptation of a bad solution is based on the Boltzmann Probabilistic Distribution:

$$P(x) = e^{-\frac{E(x)}{T}}$$

(10)

Where $E$ is the change of the Energy and $T_i$ is the temperature in the moment $i$. In the first level of the temperature, bad solutions are accepted as well, anyways, when the temperature go down, Simulated Annealing behaves similar to Tabu Search(Glover and Laguna, 1997).

Recently, many problems have been solved using Simulated Annealing metaheuristic. For instance, Neuro Fuzzy - Systems(Czabaski, 2006), Contrast Functions for BSS(Garriz et al., 2005), Cryptanalysis of Transposition Cipher(Song et al., 2008), Transmitter-Receiver Collaborative-Relay Beam-forming(Zheng et al., 2011) and Two-Dimensional Strip Packing Problem(Dereli and Sena Da, 2007) have been solved through Simulated Annealing inspired algorithms.

2.4 Metaheuristic of Deterministic Swapping

Metaheuristic Of Deterministic Swapping (MODS) (Niño, 2011) is a local search strategy that explores the Feasible Solution Space of a Combinatorial Problem supported in a data structure named Multi Objective Deterministic Finite Automata (MDFA) (Niño et al., 2010a). A MDFA is a Deterministic Finite Automata that allows the representation of the feasible solution space of a Combinatorial Problem. Formally, a MDFA is defined as follows:

$$M = (Q, \Sigma, \delta, Q_0, F(X))$$

(11)

Where $Q$ represents all the set of states of the automata (feasible solution space), $\Sigma$ is the input alphabet that is used for $\delta$ (transition function) to explore the feasible solution space of a combinatorial problem, $Q_0$ contains the initial set of states (initial solutions) and $F(X)$ are the objectives to optimize.

Example 1. MDFA for a Scheduling Parallel Machine Problem:

A Company has three machines. It is necessary to schedule three processes in parallel $P_1, P_2$ and $P_3$. Each process has a duration of 5, 10 y 50 minutes respectively. If the processes can be executed in any of the machines, how many manners the machines can be assigned to the processes? Given the Bi-objective function in (12), what is the optimal Pareto Front?

$$F(X) = \left\{ f_1(X) = \sum_{i=1}^{3} i \cdot X_i, f_2(X) = \sum_{i=1}^{3} \left( \frac{1}{7} \right) \cdot X_i \right\}$$

(12)
First of all, we need to build the MDFA. For doing this, we must define the states of the MDFA setting the structure of the solution for each state. Therefore, if we state that \( X_0 = \{(P_1, P_2, P_3)\} \) represents the solution for the state \( q_0 \): machine 1 executes the process \( P_1 \), machine 2 executes the process \( P_2 \) and machine 3 executes the process \( P_3 \) then the arrays solution for each state will be \( X_{P_0} = (P_1, P_2, P_3) \), \( X_{q_1} = (P_1, P_3, P_2) \), \( X_{q_2} = (P_2, P_1, P_3) \), \( X_{q_3} = (P_2, P_3, P_1) \), \( X_{q_4} = (P_3, P_1, P_2) \) y \( X_{q_5} = (P_3, P_2, P_1) \). Now, we have six states \( q_0, q_1, q_2, q_3, q_4, q_5 \), those represent the feasible solution space of the Scheduling problem proposed. The set of states for the MDFA of this problem can be seen in figure 3.

![Figure 3: Set of states for the MDFA of example 1](image)

Once the set of states is defined, the Input Alphabet (\( \Sigma \)) and the Transition Function (\( \delta \)) be done. It is very important to take into account, first, the bond of both allows to perturb the solutions in all the possible manners, in other words, we can change of state using both allows to perturb the solutions in all the possible manners, like this are intractable for a large number of variables (machines) was done. For this reason, it was easy to draw the entire MDFA. However, problems like this are intractable for a large number of variables, in other words, when the number of variables grow the feasible solution space grows exponentially. In this manner, it is not a good idea to draw the entire feasible solution space and pick the best solutions. Thus, what should we do in order to solve any combinatorial problem, without taking into account its size, using a MDFA? Looking an answer to this question, MODS was proposed.

MODS explores the feasible solution space represented through a MDFA using a search direction given by an elitist set of solutions (\( Q \_e \)). The elitist solution are states that, when were visited, their solution dominated at least one solution of an element in \( Q \_e \). \( Q \_e \) contains all the states with non-dominated solutions. Due to this, it can be inferred that the elements of \( Q \_e \) are contained in \( Q \_e \), for this reason is true that:

![Figure 4: MDFA for example 1, Parallel execution of processes.](image)

Finally, the solution of each state is replaced in (12). The results can be seen in table 1 and the Optimal Pareto Front is shown in figure 5.

![Figure 5: Pareto Front for the MDFA of example 1, Parallel execution of processes.](image)

As can be seen in figure 4, the feasible solution space for this problem was described using a MDFA. Also, unfeasible solutions are not allowed owing to the definition of \( \Sigma \). Nevertheless, the general problem was not solved, only a particular case of three variables (machines) was done. For this reason, it was easy to draw the entire MDFA. However, problems like this are intractable for a large number of variables, in other words, when the number of variables grow the feasible solution space grows exponentially. In this manner, it is not a good idea to draw the entire feasible solution space and pick the best solutions. Thus, what should we do in order to solve any combinatorial problem, without taking into account its size, using a MDFA? Looking an answer to this question, MODS was proposed.

<table>
<thead>
<tr>
<th>State</th>
<th>Assignments</th>
<th>Genes</th>
<th>Times</th>
<th>( F(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( M_1 ) ( M_2 ) ( M_3 )</td>
<td>( M_1 ) ( M_2 ) ( M_3 )</td>
<td>( M_1 ) ( M_2 ) ( M_3 )</td>
<td>( F(X) )</td>
</tr>
<tr>
<td>( q_0 )</td>
<td>( P_1 ) ( P_2 ) ( P_3 )</td>
<td>10 ( 5 ) ( 3 )</td>
<td>125</td>
<td>36.66</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( P_1 ) ( P_3 ) ( P_2 )</td>
<td>10 ( 5 ) ( 3 )</td>
<td>130</td>
<td>29.16</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( P_2 ) ( P_1 ) ( P_3 )</td>
<td>50 ( 10 ) ( 5 )</td>
<td>85</td>
<td>56.66</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( P_2 ) ( P_3 ) ( P_1 )</td>
<td>50 ( 10 ) ( 5 )</td>
<td>90</td>
<td>55.83</td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( P_3 ) ( P_1 ) ( P_2 )</td>
<td>5 ( 10 ) ( 5 )</td>
<td>155</td>
<td>26.66</td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( P_3 ) ( P_2 ) ( P_1 )</td>
<td>5 ( 10 ) ( 5 )</td>
<td>135</td>
<td>33.33</td>
</tr>
</tbody>
</table>

Table 1: Values of \( F(X) \) for the states of example 1.
Step 1. Create the initial set of solutions \( S_0 \) using a heuristic relative to the problem to solve.

Step 2. Set \( S \) as \( S_0 \) and \( S^* \) as \( \emptyset \).

Step 3. Select a random solution \( s \in S \) or \( s \in S^* \).

Step 4. Perturb \( s \) in order to know its neighborhood. For those solutions that are non-dominated by elements of \( S \), join to this set. In addition, add to \( S^* \) those solutions found that dominated at least one element from \( S \).

Step 5. Check stop condition, go to 3.

\( S \) is the solution set of non-dominated solutions and \( S^* \) is the set of Elitism Solutions. As can be seen, \( S^* \) works as a search direction for \( S \).

3 AN HYBRID METAHEURISTIC FOR THE MULTI-OBJECTIVE OPTIMIZATION OF COMBINATORIAL PROBLEMS

Evolutionary Metaheuristic of Simulated Annealing (EMSA) is an improvement of MODS metaheuristic. EMSA is defined as a hybrid metaheuristic based on Genetic Algorithms, Simulated Annealing and Deterministic Swapping (Niño, 2012) for the multi-objective optimization of combinatorial problems. Its main propose consists in optimizing a combinatorial problem using a Search Direction and an Angle Improvement. EMSA is based in the next Automata:

\[ M_{EMSA} = (S, S_0, P(s), A(n), C(q, r, k), F(X)) \]  

(14)

Alike MODS, \( S_0 \) is the set of initial solutions, \( S \) is the feasible solution space (unknown) and \( F(X) \) are the functions of the combinatorial problem. \( P(s), C(q, r, k) \), and \( A(n) \) are defined as follows:

\( P(s) \) is the Permutation Function, formally it is defined as follows:

\[ P(s) : S \rightarrow S \]  

(15)

\( P \) receives a solution \( s \in S \) and perturbs it returning a new solution \( s' \in S \). The perturbation can be done based on the representation of the solutions. An example of some perturbations based on the representation of the solution can be seen in figure 6.

\[ A(n) \] is the Weight Function. Formally, it is defined as follow:

\[ A(n) : \mathbb{N} \rightarrow \mathbb{R}^n \]  

(16)

Where \( n \) is the number of objectives of the problem.

Function \( A \) receives a natural number as parameter and it returns a vector with the weights. The weight values are randomly generated with an uniform distribution. Those represent the weight to assign to each function of the combinatorial problem. The weight values returned by the function fulfill the next constrain:

\[ \sum_{i=1}^{n} \alpha_i = 1, 0 \leq \alpha_i \leq 1 \]  

(17)

Where \( \alpha_i \) is the weight assigned to function \( i \). Table 2 shows some vectors randomly generated by \( A(n) \).

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Function</th>
<th>Vector of Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( A(2) )</td>
<td>{0.6, 0.4}</td>
</tr>
<tr>
<td>3</td>
<td>( A(3) )</td>
<td>{0.2, 0.4, 0.4}</td>
</tr>
<tr>
<td>4</td>
<td>( A(4) )</td>
<td>{0.3, 0.8, 0.1, 0.0}</td>
</tr>
</tbody>
</table>

The weights, in an implicit manner, allow setting the angle direction to the solutions. The angle direction is the course being followed by the solutions in order to optimize \( F(X) \). Hence, when the weights values are changed, the angle of optimization is changed and a new search direction is obtained. For instance, different search directions for different weight values are shown in figure 7 in a bi-objective combinatorial problem. Due to this, (6) is rewritten as follows:

\[ F(X) = \sum_{i=1}^{n} \alpha_i \cdot f_i(X) \]  

(18)

Where \( n \) is the number of objectives of the problem and \( \alpha_i \) is the weight assigned to the function \( i \). The weights fulfills the constrain established in (17).

Cross Function \( K \) is defined as follows:

\[ C(q, r, k) : S \rightarrow S \]  

(19)

Figure 7: Different angles given by different weights for a bi-objective Problem.
Where \( q, r \in S \) and \( k \in \mathbb{N} \). \( q \) and \( r \) are named parents solutions and \( k \in \mathbb{N} \) is the cross point. The propose of this function is cross two solutions in the same point and returns a new solution.

Lastly, EMSA template is defined as follows:

Step 1. Setting sets. Set \( S_0 \) as the set of Initial Solutions. Set \( S_0 \) and \( S_r \), as \( S_0 \).

Step 2. Settings parameters. Set \( T \) as the initial temperature, \( n \) as the number of objectives of the problem and \( \rho \) as the cooler factor.

Step 3. Setting Angle. If \( T \) is equal to 0 then go to 9, else set \( T_{i+1} = \rho \times T_i \), and go to step 4.

Step 4. Selection. Randomly select \( s \in S_0 \cup s \in S_r \), set \( W = A(n) = \{ w_1, w_2, \cdots , w_n \} \) and go to step 5.

Step 5. Mutation. Set \( s = P(s) \), add to \( S_0 \) and \( S_r \), according to the next rules:

\[
S_0 = S_0 \cup \{ s \} \iff (\exists r \in S_0)(r \text{ dominated to } s') \\
S_r = S_r \cup \{ s' \} \iff (\exists r \in S_r)(s' \text{ dominated to } r)
\]  

Where \( T_i \) is the temperature value in moment \( i \) and \( \gamma \) is defined as follows:

\[
\gamma = \sum_{j=1}^{n} w_j \cdot f_j(s^X) - \sum_{i=1}^{n} w_i \cdot f_i(s'X)
\]  

Where \( s^X \) is the vector \( X \) of solution \( s \), \( s'X \) is the vector \( X \) of solution \( s' \), \( w_i \) is the weight assigned to the function \( i \) and \( n \) is the number of objectives of the problem.

If \( n < \gamma \) then set \( s \) as \( s' \) and go to step 4 else go to step 7.

Step 7. Crossover. Set \( s' \) as:

\[
s' = C(s, s', k)
\]

Where \( 1 \leq k \leq ||s^X|| \). Go to step 5.

Step 8. Removing dominated solutions. Remove the dominated solution for each set \( (S_0, \text{ and } S_r) \). Go to step 3.

Step 9. Finishing. \( S_0 \) has the non-dominated solutions.

As can be seen in figure 8, immediately, EMSA removes the dominated solutions when the new solution found is not dominated. Furthermore, if the new found solution dominated at least one element from the solution set \( (S_0) \) then it will be added to the elitisms set \( (S_r) \) that works as a search direction for the Pareto Front owing to it contains the best non-dominated solutions found. As far as here, EMSA could sounds as a simple local search strategy but not, when a new solution found is dominated, EMSA tries to improve it using guessing. Guessing is done accepting dominated solution as good solutions. Alike Simulated Annealing inspired algorithms, the dominated solutions are accepted under the Boltzmann Distribution Probability assigning weights to the objectives of the problem. It is probably that perturbing a dominated solution, a non-dominated solution can be found as can be seen in figure 8 and 9. Due to this, local optimums are avoided. When the temperature is low, the bad solutions are avoided because \( \gamma \) value is low therefore EMSA accepts only no dominated solutions. Finally, when a bad solution is rejected because of \( \gamma \) value, EMSA uses crossover in order to explore new regions of the feasible solution space trying to avoid local optimums.

4 COMPLEXITY OF THE METAHEURISTIC PROPOSED

For the calculation of EMSA Complexity, we are going to consider assignations, arithmetic operations and other simple instructions are \( O(k) \), where \( k \) is a constant. Besides, the external cycle (step 3), the cre-
ation of a search direction (step 4), the crossover (step 7), looking for dominated solutions (step 5) and remove the dominated solution of a set (step 8), all of them have a value of $O(n)$ where $n$ is a large value. Thus, the equation that represents EMSA run times is defined as follows:

$$T_{EMSA}(n) = \sum_{i=1}^{n} 4 \cdot n \cdot O(k)$$  \hspace{1cm} (25)

Thus,

$$T_{EMSA}(n) = 4 \cdot n^2 + n \cdot O(k)$$ \hspace{1cm} (26)

Applying the well-known Big-O notation in (26) we find the upper bound of EMSA:

$$T_{EMSA}(n) = O(n^2)$$ \hspace{1cm} (27)

The lower bound is calculated using (26). Hence:

$$T_{EMSA}(n) = 4 \cdot n^2 + n \cdot O(k) \geq 4 \cdot n + n \cdot O(k) = (4 + O(k)) \cdot n = (c \cdot n)$$ \hspace{1cm} (28)

Thus, applying the well-known $\Omega$ notation in (28) we find the lower bound of EMSA:

$$T_{EMSA}(n) = \Omega(c \cdot n)$$ \hspace{1cm} (29)

As can be seen in (Niño, 2011), EMSA and MODS has the same complexity.

5 EXPERIMENTAL RESULTS

5.1 Performance Metrics

There are metrics that allow measuring the quality of a set of optimal solutions and the performance of an Algorithm (D. and J., 2003). Most of them use two Pareto Fronts. The first one is $PF_{true}$ and it refers to the real optimal solutions of a combinatorial problem. The second is $PF_{know}$ and it represents the optimal solutions found by an algorithm.

**Generation of Non-dominated Vectors (GNDV)** It measures the number of No Dominates Solutions generated by an algorithm.

$$GNDV = ||PF_{know}||$$ \hspace{1cm} (30)

A higher value for this metric is desired.

**Rate of Generation of No-dominated Vectors (RGNDV)** This metric measures the proportion of the Non-dominated Solutions (30) generated by an algorithm and the Real Solutions.

$$RGNDV = \left( \frac{GNDV}{||PF_{true}||} \right) \cdot 100\%$$ \hspace{1cm} (31)

A value closer to 100% for this metric is desired.

**Real Generation of Non-dominated Vectors (ReGNDV)** This metric measures the number of Real Solutions found by an algorithm.

$$ReGNDV = ||\{y \in PF_{know} \land y \in PF_{true}\}||$$ \hspace{1cm} (32)

A value closer to $|PF_{true}|$ for this metric is desired.

**Generational Distance (GD)** This metric measures the distance between $PF_{know}$ and $PF_{true}$. It allows to determinate the error rate in terms of the distance of a set of solutions relative to the real solutions.

$$GD = \left( \frac{1}{||PF_{know}||} \cdot \left( \sum_{i=1}^{n} d_i \right)^{(1/p)} \right)$$ \hspace{1cm} (33)

Where $d_i$ is the smallest Euclidean distance between the solution $i$ of $PF_{know}$ and the solutions of $PF_{true}$. $p$ is the dimension of the combinatorial problem, it means the number of objective functions.

**Inverse Generational Distance (IGD)** This is another distance measurement between $PF_{know}$ and $PF_{true}$:

$$IGD = \left( \frac{1}{||PF_{true}||} \cdot \left( \sum_{i=1}^{n} d_i \right)^{(1/p)} \right)$$ \hspace{1cm} (34)

Where $d_i$ is the smallest Euclidean distance between the solution $i$ of $PF_{know}$ and the solutions of $PF_{true}$.

**Spacing (S)** It measures the range variance of neighboring solutions in $PF_{know}$.

$$S = \left( \frac{1}{||PF_{know}|| - 1} \cdot \left( \sum_{i=1}^{n} (\bar{d} - d_i)^2 \right)^{(1/p)} \right)$$ \hspace{1cm} (35)

Where $d_i$ is the smallest Euclidean distance between the solution $i$ of $PF_{know}$ and the rest of solutions of $PF_{know}$. $\bar{d}$ is the mean of all $d_i$. $p$ is the dimension of the combinatorial problem.

A value closer to 0 for this metric is desired. A value of 0 means that all the solutions are equidistant.

**Error Rate ($\varepsilon$)** It estimates the error rate respect to the precision of the Real Algorithms Solutions (27) as follows:

$$\varepsilon = \left( \frac{||PF_{true}||}{ReGNDV} \right) \cdot 100\%$$ \hspace{1cm} (36)

A value of 0% in this metric means that the values of the Real Pareto Front are constructed from the values of the Algorithm Pareto Front.

Lastly, notice that every metric by itself does not have sense. It is necessary to support in the other metrics for a real judge about the quality of the solutions. For instance, if a Pareto Front has a higher value in GNDV but a lower value in ReGNDV then the solutions has a poor-quality.
5.2 Experimental Settings

The proposed algorithm was tested using well-known instances from the Bi-objective Traveling Salesman Problem taken from TSPLIB (Heidelberg). The input parameters for the algorithms are shown in Table 3. The test of the algorithms was made using a Dual Core Computer with 2 Gb RAM.

Each algorithm was running ten times and the best non-dominated solutions were taken for making a real comparison. The optimal solutions were constructed based on the best non-dominated solutions of MODS and EMSA for each instance worked.

Table 3: Parameters setting for each compared algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Iterations</th>
<th>Perturbations</th>
<th>Initial Temperature</th>
<th>Cooler Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODS</td>
<td>100</td>
<td>80</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>EMSA</td>
<td>100</td>
<td>NA</td>
<td>1000</td>
<td>0.339</td>
</tr>
</tbody>
</table>

The test made with bi-objectives instances is shown in Table 4. In addition, a graphical comparison is shown in Figure 10.

6 CONCLUSIONS

EMSA is a improvement of MODS metaheuristic, both are based on the Automata Theory (Deterministic Swapping) but EMSA do it in an implicit manner. EMSA is usefull for solving problems such as TSP and its applications. Besides, the proposed algorithm shows a better computational performance than MODS in TSP problem with the same complexity. Furthermore, EMSA uses a search direction in order to optimize the solutions. The search direction is given by a linear combination of the functions to optimize. With respect to local optimums, EMSA try to avoid it in two times, using Guessing and Crossover. Guessing is made under the Boltzmann Probability Distribution accepting bad solutions as well as good solutions. On the other hand, crossover is done with a simple, crossover function. Lastly, metrics from the specialized literature were used for comparing EMSA and MODS solutions. The results shows that EMSA has a better computational performance than MODS and in some cases the superiority was 100% out of 100%.

REFERENCES


Figure 10: Graphical comparison between MODS and EMSA for bi-objective instances.

Table 4: Measuring algorithms performance for bi-objective instances of TSP with multi-objective optimization metrics.

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>ALGORITHM</th>
<th>GNDV</th>
<th>RGNDV</th>
<th>ReGNDV</th>
<th>(GNDV/ReGNDV)</th>
<th>%</th>
<th>S</th>
<th>GD</th>
<th>IGD</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>KROAB100</td>
<td>MODS</td>
<td>289</td>
<td>0.044</td>
<td>0 0</td>
<td>0%</td>
<td>0.019</td>
<td>14.015</td>
<td>2300.006</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAB100</td>
<td>EMSA</td>
<td>8479</td>
<td>1</td>
<td>8479</td>
<td>100%</td>
<td>0.001</td>
<td>17.173</td>
<td></td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAC100</td>
<td>MODS</td>
<td>217</td>
<td>0.031</td>
<td>0 0</td>
<td>0%</td>
<td>0.034</td>
<td>19.12</td>
<td>2451.132</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAC100</td>
<td>EMSA</td>
<td>7023</td>
<td>1</td>
<td>7023</td>
<td>100%</td>
<td>0.001</td>
<td>17.997</td>
<td>1801.158</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAD100</td>
<td>MODS</td>
<td>6239</td>
<td>1</td>
<td>6239</td>
<td>100%</td>
<td>0.052</td>
<td>11.848</td>
<td>6.026</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAD100</td>
<td>EMSA</td>
<td>484</td>
<td>0.548</td>
<td>0 0</td>
<td>0%</td>
<td>0.033</td>
<td>1.204</td>
<td>248.344</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAE100</td>
<td>MODS</td>
<td>6440</td>
<td>1</td>
<td>6440</td>
<td>100%</td>
<td>0.001</td>
<td>0.028</td>
<td>5.019</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROAE100</td>
<td>EMSA</td>
<td>5919</td>
<td>1</td>
<td>5919</td>
<td>100%</td>
<td>0.001</td>
<td>0.034</td>
<td>7.992</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBC100</td>
<td>MODS</td>
<td>6919</td>
<td>1</td>
<td>6919</td>
<td>100%</td>
<td>0.001</td>
<td>0.034</td>
<td>7.992</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBC100</td>
<td>EMSA</td>
<td>5034</td>
<td>1</td>
<td>5034</td>
<td>100%</td>
<td>0.001</td>
<td>0.030</td>
<td>8.16</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBD100</td>
<td>MODS</td>
<td>5934</td>
<td>1</td>
<td>5934</td>
<td>100%</td>
<td>0.001</td>
<td>0.030</td>
<td>8.16</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBD100</td>
<td>EMSA</td>
<td>491</td>
<td>0.548</td>
<td>0 0</td>
<td>0%</td>
<td>0.033</td>
<td>1.204</td>
<td>248.344</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBE100</td>
<td>MODS</td>
<td>5802</td>
<td>1</td>
<td>5802</td>
<td>100%</td>
<td>0.001</td>
<td>0.030</td>
<td>8.16</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROBE100</td>
<td>EMSA</td>
<td>491</td>
<td>0.548</td>
<td>0 0</td>
<td>0%</td>
<td>0.033</td>
<td>1.204</td>
<td>248.344</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROCD100</td>
<td>MODS</td>
<td>6801</td>
<td>1</td>
<td>6801</td>
<td>100%</td>
<td>0.001</td>
<td>0.029</td>
<td>7.229</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROCD100</td>
<td>EMSA</td>
<td>7745</td>
<td>1</td>
<td>7745</td>
<td>100%</td>
<td>0.001</td>
<td>0.026</td>
<td>5.227</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROCE100</td>
<td>MODS</td>
<td>4613</td>
<td>1</td>
<td>4613</td>
<td>100%</td>
<td>0.001</td>
<td>0.029</td>
<td>7.229</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KROCE100</td>
<td>EMSA</td>
<td>4613</td>
<td>1</td>
<td>4613</td>
<td>100%</td>
<td>0.001</td>
<td>0.029</td>
<td>7.229</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KRODE100</td>
<td>MODS</td>
<td>4613</td>
<td>1</td>
<td>4613</td>
<td>100%</td>
<td>0.001</td>
<td>0.029</td>
<td>7.229</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>KRODE100</td>
<td>EMSA</td>
<td>4613</td>
<td>1</td>
<td>4613</td>
<td>100%</td>
<td>0.001</td>
<td>0.029</td>
<td>7.229</td>
<td>100%</td>
<td>0%</td>
</tr>
</tbody>
</table>