# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities




WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



# 4

# Evolutionary Algorithms Based on the Automata Theory for the Multi-Objective Optimization of Combinatorial Problems

Elias D. Niño
*Universidad del Norte*
*Colombia*

## 1. Introduction

In this chapter we are going to study metaheuristics based on the Automata Theory for the Multi-objective Optimization of Combinatorial Problems. As well known, 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 & Ming-Yang, 2004). In this field it is possible to find a lot of problems denominated NP-Hard, that is mean that the problem does not have a solution in Polynomial Time. For instance, problems such as Multi-depot vehicle routing problem(Lim & Wang, 2005), delivery and pickup vehicle routing problem with time windows(Wang & Lang, 2008), multi-depot vehicle routing problem with weight-related costs(Fung et al., 2009), Railway Traveling Salesman Problem(Hu & Raidl, 2008), Heterogeneous, Multiple Depot, Multiple Traveling Salesman Problem(Oberlin et al., 2009) and Traveling Salesman with Multi-agent(Wang & Xu, 2009) are categorized as NP-Hard problems.

One of the most classical problems in the Combinatorial Optimization Field is the Traveling Salesman Problem (TSP), it has been analyzed for years(Sauer & 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:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \cdot X_{ij} \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} X_{ij} = 1, \forall i = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} X_{ij} = 1, \forall j = 1, \ldots, n \tag{3}$$

$$\sum_{i \in \kappa} \sum_{j \in \kappa} X_{ij} \leq |\kappa| - 1, \forall \kappa \subset \{1, \ldots, n\} \tag{4}$$

$$X_{ij} = 0, 1 \forall i, j \tag{5}$$

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 constrain of visiting each city only once. Lastly, Equation (4) set the subsets of solutions, avoiding cycles in the tour.

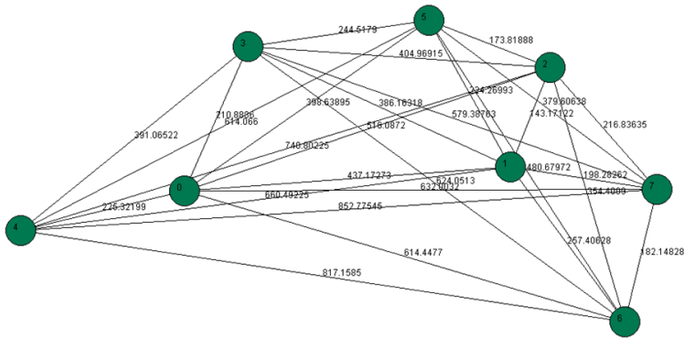



Fig. 1. TSP instance of ten cities

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 instance, problems such as Heterogeneous Machine Scheduling(Kim & 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 & 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 & 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 & 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 & Engelbrecht, 2007), MPEG-2 TS Multiplexers in CATV Networks(Jianghong et al., 2000), Contention Awareness(Shanmugapriya et al., 2009) and The Hard Scheduling Optimization(Niño, Ardila, Perez & Donoso, 2010) had been derived from TSP. Although several algorithms have been implemented to solve TSP, there is no one that optimal solves it. For this reason, this chapter discuss novel metaheuristics based on the Automata Theory to solve the Multi-objective Traveling Salesman Problem.

This chapter is structured as follows: Section 2 shows important definitions to understand the Multi-objective Combinatorial Optimization and the Metaheuristic Approximation. Section 3, 4 and 5 discuss Evolutionary Metaheuritics based on the Automata Theory for the Multi-objective Optimization of Combinatorial Problems. Finally, Section 6 and 7 discuss the Experimental Results of each proposed Algorithm using Multi-objective Metrics from the specialized literature.

## 2. Preliminaries

### 2.1 Multi-objective 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:

$$optimize \quad F(X) = \{f_1(X), f_2(X), \ldots, f_n(X)\} \tag{6}$$

Subject to:

$$H(X) = 0 \tag{7}$$

$$G(X) \leq 0 \tag{8}$$

$$X_l \leq X \leq X_u \tag{9}$$

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. For instance, table 1 shows three solutions for a particualr Mono-objective Problem. If we suppose that those are related to a maximization problem then the Optimal Solution (found) is the solution 1 otherwise (minimization) will be the solution 2. On the other hand, in table 2 can be seen three solutions for a particular Tri-objective Problem. Thus, if we suppose that all the components of the solutions are related with a minimization problem, solution 2 is a *dominated solution* due to all the components (0.8, 0.9 and 1.0) are the biggest values. On the other hand, solution 0 and 1 are *no-dominated solutions* due to in the first and second component (0.6 and 0.4) solution 0 is bigger than the relative components of the solution 1 but in the third component (0.5) solution 0 is lower than the same component in solution 1. Both examples show the

| $k$ | $F(X_k)$ |
|---|---|
| 0 | 10 |
| 1 | 20 |
| 2 | 5 |

Table 1. Solutions for a particular Mono-objective Problem

difference between Mono-objective and Multi-objective Optimization. While the first deal with finding the Optimal Solution, the last does with finding a set of Optimal Solutions. In Combinatorial Optimization, the set of Optimal Solution is called *Pareto Front*. It contains all the no-dominated solutions for a Multi-objective Problem. Figure 2 shows a Pareto Front for a particular Tri-objective Problem. Lastly, it is probably that some Multi-objective Problems

| $k$ | $F(X_k) = \{f_0(X_k), f_1(X_k), f_2(X_k)\}$ |
|---|---|
| 0 | $\{0.6, 0.4, 0.5\}$ |
| 1 | $\{0.2, 0.3, 0.8\}$ |
| 2 | $\{0.8, 0.9, 1.0\}$ |

Table 2. Solutions for a particular Tri-objective Problem

have an infinite Pareto Front, in those cases is necessary to determinate how many solutions are required, for instance, using a maximum number of solution permitted in the Pareto Front.

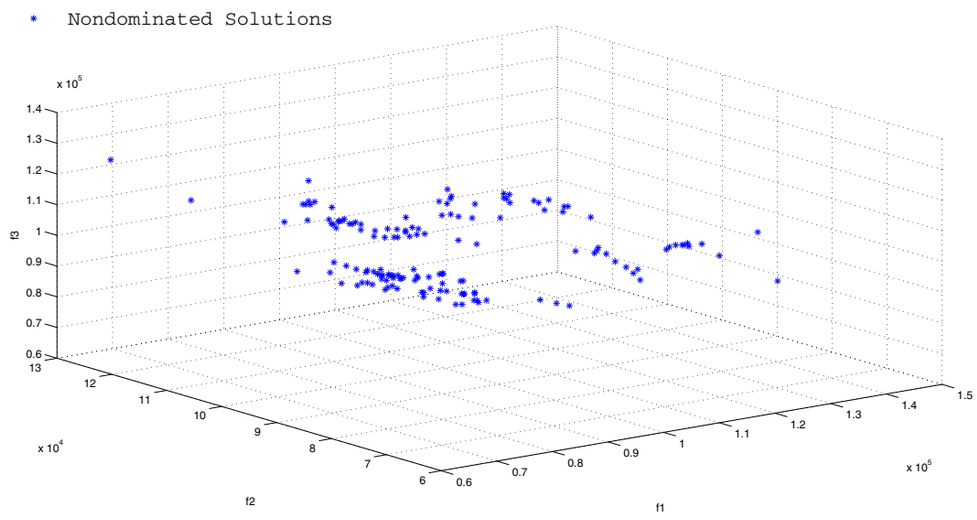



Fig. 2. Pareto Front for a particular Tri-objective Problem

### 2.2 Tabu search

Tabu Search(Glover & Laguna, 1997) is a basic local search strategy for the Optimization of Combinatorial Problems. It is defined as follows: Given $S$ as the Initial Solutions Set.

*Step 1. Selection.* Select $x \in S$

*Step 2. Perturbation.* Perturbs the solution $x$ for the purpose of knowing its Neighborhood ($N(x)$). Perturbing a solution means to modify the solution $x$ in order to obtain a new solution ($x_i^{'}$). The solutions found are called Neighbors, and those represent the Neighborhood. For instance, figure 3 shows three perturbations for a $x$ solutions and the new solutions $x_1^{'}$,$x_2^{'}$ and $x_3^{'}$ found. The perturbation can be done according to the representation of the solutions. Regularly, the representations of the solutions in Combinatorial Problems are based on Discrete Structures such as Vectors, Matrices, Queues and Lists. Lastly, good solutions are added to $S$.

*Setp 3. Check Stop Condition.* The stop condition can be delimited using rules such as number of execution without improvement or maximum number of iteration exceeded.

Recently, novels Tabu Search inspired Algorithms have been developed in order to solve Combinatorial Problems such as Permutation Flow Shop Scheduling(Ren et al., 2011), Displacement based on Support Vector(Fei et al., 2011), Examination Timetabling Problem(Malik et al., 2011), Partial Transmit Sequences for PAPR Reduction(Taspinar et al., 2011), Inverse Problems(An et al., 2011), Fuzzy PD Controllers(Talbi & Belarbi, 2011b), Instrusion Detection(Jian-guang et al., 2011), Tel-Home Care Problems(Lee et al., 2011), Ant Colony inspired Problems(Zhang-liang & Yue-guang, 2011), Steelmaking-Continuous Casting Production Scheduling(Zhao et al., 2011), Fuzzy Inference System(Talbi & Belarbi, 2011a) and Coordination of Dispatchable Distributed Generation and Voltage Control Devices(Ausavanop & Chaitusaney, 2011).

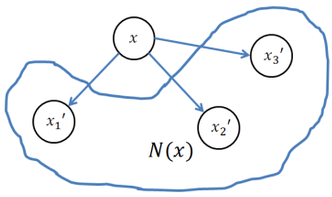



Fig. 3. The Neighborhood of a solution $x$ is known after being perturbed

### 2.3 Genetic algorithms

Genetic Algorithms are Algorithms based on the Theory of Natural Selection(Wijkman, 1996). Thus, Genetic Algorithms mimics the realBehavior Genetic Algorithms(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 4 shows the basics steps of a Genetic Algorithm. The most known Genetic

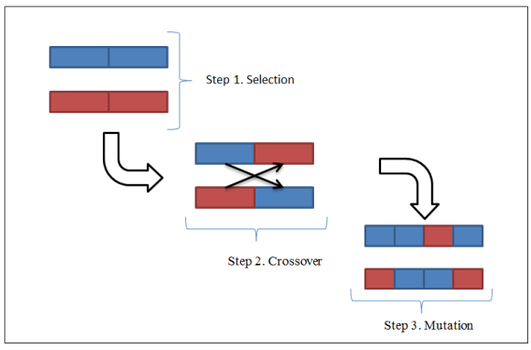



Fig. 4. Basics steps of a Genetic Algorithm

Algorithms from the literature(Dukkipati & 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; 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 fiel. For instance, problems such as Pattern-recognition based Machine Translation System(Sofianopoulos & Tambouratzis, 2011), Tuning of Fuzzy Logic controllers for a heating(Gacto et al., 2011), Real-coded Quantum Clones(Xiawen & 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 & Keedwell, 2011) and Vendor-managed Inventory(Azuma et al., 2011) have been solved using SPEA and NSGA-II.

### 2.4 Simulated Annealing algorithms

Simulated Annealing(Kirkpatrick et al., 1983) is a generic probabilistic metaheuristic based in the Annealing in Metallurgy. Similar to Tabu Search, Simulated Annealing explores the neighborhood of solutions being flexible with no-good solutions. That is mean, accepting bad solutions as well as good solution, but only in the first iterations. The acceptation of a bad solution is based on the Boltzmann Probabilistic Distribution:

$$P(x) = e^{\left(-\left(\frac{E}{T_i}\right)\right)} \tag{10}$$

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 (only accept good solutions).

Recentrly, similar to Genetic Algoritms and Tabu Search, many problems have been solved using Simulated Annealing metaheuristic. For instance, Neuro Fuzzy - SystemsCzabaski (2006), Contrast Functions for BSSGarriz et al. (2005), Cryptanalysis of Transposition CipherSong et al. (2008), Transmitter-Receiver Collaborative-Relay BeamformingZheng et al. (2011) and Two-Dimensional Strip Packing ProblemDereli & Sena Da (2007) have been solved through Simulated Annealing inspired algorithms.

### 2.5 Deterministic Finite Automata

Formally, a Deterministic Finite Automata is a Quint-tuple defined as follows:

$$A = (Q, \Sigma, \delta, q_0, F) \tag{11}$$

Set of transitions $\delta$. The set of transitions ($\delta$) describes the behavior of the automata. Let $a \in S$ and $q, r \in Q$, then the function is defined as follows:

$$\delta(q, a) = r \tag{12}$$

*Example 1.* Let $A = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2\}$, $S = \{0, 1\}$, $F = \{q_1\}$ and the set of transitions $\delta$ defined in table 3, the representation of $A$ using a state diagram can be derived as shown in figure 5. Notice that each state of DFA has transitions with all the elements of $\Sigma$.

| | 0 | 1 |
|---|---|---|
| $q_0$ | $q_2$ | $q_0$ |
| $q_1$ | $q_1$ | $q_1$ |
| $q_2$ | $q_2$ | $q_1$ |

Table 3. Set of transitions for the DFA of example 1

### 2.6 Metaheuristic Of Deterministic Swapping (MODS)

Metaheuristic Of Deterministic Swapping (MODS) (Niño et al., 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, Ardila, Donoso & Jabba, 2010). A MDFA is a Deterministic Finite Automata that allows the

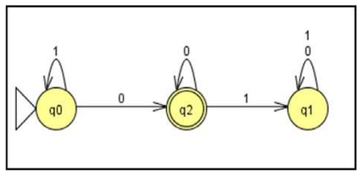



Fig. 5. Automata state diagram for the example 1.

representation of the feasible solution space of a Combinatorial Problem. Formally, a MDFA is defined as follows:

$$M = (Q, \Sigma, \delta, Q_0, F(X)) \tag{13}$$

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 (10), what is the optimal Pareto Front?

$$F(X) = \left\{ f_1(X) = \sum_{i=1}^{3} i \cdot X_i, f_2(X) = \sum_{i=1}^{3} \left(\frac{1}{i}\right) \cdot X_i \right\} \tag{14}$$

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 $X_q = (P_k, P_i, P_j)$ represents the solution for the state $q$: machine 1 executes the process $P_k$, machine 2 executes the process $P_i$ and machine 3 executes the process $P_j$ then the arrays solution for each state will be $X_q0 = (P_1, P_2, P_3)$, $X_q1 = (P_1, P_3, P_2)$, $X_q2 = (P_2, P_1, P_3)$, $X_q3 = (P_2, P_3, P_1)$, $X_q4 = (P_3, P_1, P_2)$ y $X_q5 = (P_3, P_2, P_1)$. Now, we have six states $q_0$,$q_1$,$q_2$,$q_3$,$q_4$ and $q_5$, 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 6. Once the set of states is defined, the Input Alphabet

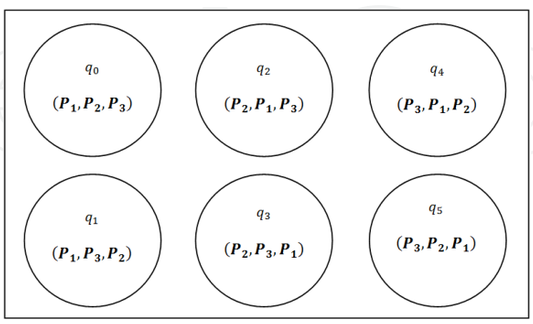



Fig. 6. Set of states for the MDFA of example 2

($\Sigma$) 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 the combination of $\Sigma$ and $\delta$. Obviously, doing this, we avoid

unfeasible solutions. Regarding the proposed problem, we propose the set $\Sigma$ as follows:

$$\Sigma = \{(P_1, P_2), (P_1, P_3), (P_2, P_3)\} \tag{15}$$

Hence, it is elemental that $\delta(q_0, (P_1, P_2)) = q_2$, $\delta(q_0, (P_1, P_3)) = q_5$, ... , $\delta(q_5, (P_2, P_3)) = q_3$. At this part, the transitions has been defined therefore the MDFA can be seen in figure 7.

Finally, the solution of each state is replaced in (10). The results can be seen in table 4 and the Optimal Pareto Front is shown in figure 8.

| State | Assignments | | | Times | | | $F(X)$ | |
|---|---|---|---|---|---|---|---|---|
| $q_i$ | $M_1$ | $M_2$ | $M_3$ | $M_1$ | $M_2$ | $M_3$ | $f_1(X)$ | $f_2(X)$ |
| $q_0$ | $P_1$ | $P_2$ | $P_3$ | 10 | 50 | 5 | 125 | 36.66 |
| $q_1$ | $P_1$ | $P_3$ | $P_2$ | 10 | 5 | 50 | 170 | 29.16 |
| $q_2$ | $P_2$ | $P_1$ | $P_3$ | 50 | 10 | 5 | 85 | 56.66 |
| $q_3$ | $P_2$ | $P_3$ | $P_1$ | 50 | 5 | 10 | 90 | 55.83 |
| $q_4$ | $P_3$ | $P_1$ | $P_2$ | 5 | 10 | 50 | 175 | 26.66 |
| $q_5$ | $P_3$ | $P_2$ | $P_1$ | 5 | 50 | 10 | 135 | 33.33 |

Table 4. Values of $F(X)$ for the states of example 2

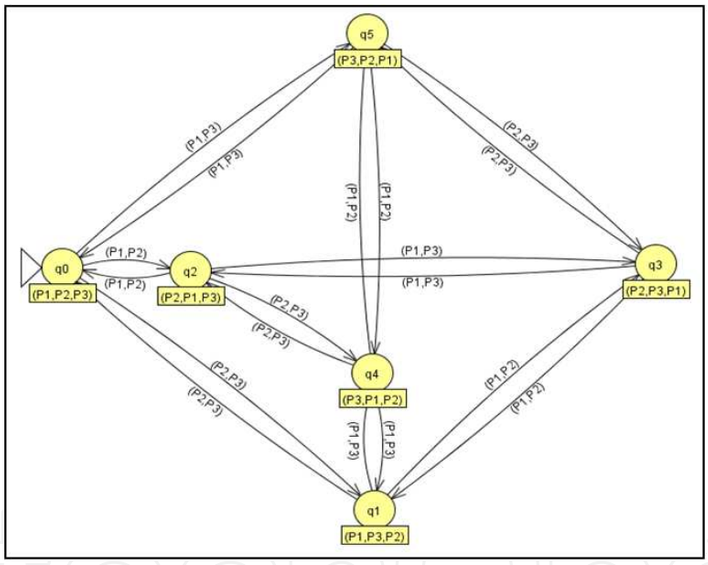



Fig. 7. MDFA for example 2, Parallel execution of processes

As can be seen in figure 7, the feasible solution space for this problem was described using a MDFA. Also, unfeasible solutions are not allowed because of 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.

MODS explores the feasible solution space represented through a MDFA using a search direction given by an elitist set of solutions ($Q_*$). The elitist solution are states that, when

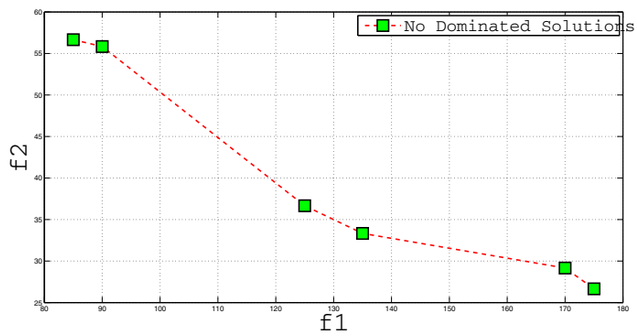



Fig. 8. Pareto Front for the MDFA of example 2, Parallel execution of processes

were visited, their solution dominated at least one solution of an element in $Q_\phi$. $Q_\phi$ contains all the states with non-dominated solutions. Due to this, it can be inferred that the elements of $Q_*$ are contained in $Q_\phi$, for this reason is true that:

$$Q_\phi = Q_\phi \cup Q_* \tag{16}$$

Lastly, the template algorithm of MODS is defined as follows:

*Step 1.* Create the initial set of solutions $Q_0$ using a heuristic relative to the problem to solve.

*Step 2.* Set $Q_\phi$ as $Q_0$ and $Q_*$ as $\phi$.

*Step 3.* Select a random state $q \in Q_\phi$ or $q \in Q_*$

*Step 4.* Explore the neighborhood of $q$ using $\delta$ and $\Sigma$. Add to $Q_\phi$ the solutions found that are not dominated by elements of $Q_f$. In addition, add to $Q_*$ those solutions found that dominated at least one element from $Q_\phi$.

*Step 5.* Check stop condition, go to 3.

## 3. Simulated Annealing Metaheuristic Of Deterministic Swapping (SAMODS)

Simulated Annealing & Metaheuristic Of Deterministic Swapping(Niño, 2012) (SAMODS) is a hybrid local search strategy based on the MODS theory and Simulated Annealing Algorithm for the Multiobjective Optimization of combinatorial problems. Its main propose consists in optimizing a combinatorial problem using a Search Direction and an Angle Improvement. SAMODS is based in the next Automata:

$$M = (Q, Q_0, P(q), F(X), A(n)) \tag{17}$$

Alike MODS, $Q_0$ is the set of initial solutions, $Q$ is the feasible solution space and $F(X)$ are the functions of the combinatorial problem. $P(q)$ and $A(n)$ are defined as follows:

$P(q)$ is the *Permutation Function*, formally it is defined as follows:

$$P(q) : Q \to Q \tag{18}$$

$P$ receives a solution $q \in Q$ and perturbs it returning a new solution $r_i \in Q$. 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 15.

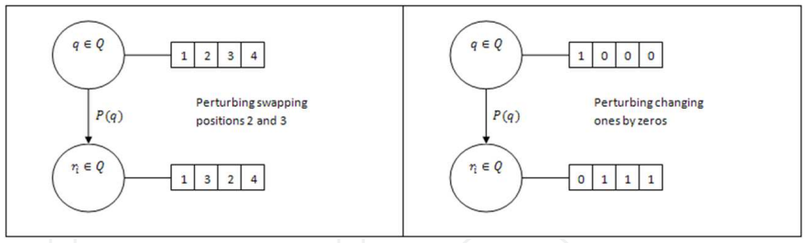



Fig. 9. Different representation and perturbation of solutions.

$A(n)$ is the *Weight Function*. Formally, it is defined as follow:

$$A(n) : \mathbb{N} \rightarrow \Re^n \tag{19}$$

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:

$$\sum_{i=1}^{n} \alpha_i = 1, 0 \leq \alpha_i \leq 1 \tag{20}$$

Where $\alpha_i$ is the weight assigned to function $i$. Table 5 shows some vectors randomly generated by $A(n)$.

| *Input Parameter* | *Function* | *Vector of Weights* |
|---|---|---|
| 2 | $A(2)$ | $\{0.6, 0.4\}$ |
| 3 | $A(3)$ | $\{0.2, 0.4, 0.4\}$ |
| 4 | $A(4)$ | $\{0.3, 0.8, 0.1, 0.0\}$ |

Table 5. Some weight vectors generated by A(n)

But, what is the importance of those weights? 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 for optimizing 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 16 in a Bi-objective combinatorial problem. Due to this, (6) is rewritten as follows:

$$F(X) = \sum_{i=1}^{n} \alpha_i \cdot f_i(X) \tag{21}$$

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 (20).

SAMODS main idea is simple: it takes advantage of the search directions given by MODS and it proposed an angle direction given by the function $A(n)$. Thus, there are two directions; the first helps in the convergence of the Pareto Front and the second helps the solutions to find neighborhoods where $F(X)$ is optimized. Due to this, SAMODS template is defined as follows:

*Step 1.* Setting sets. Set $Q_0$ as the set of Initial Solutions. Set $Q_\phi$ and $Q_*$ as $Q_0$.

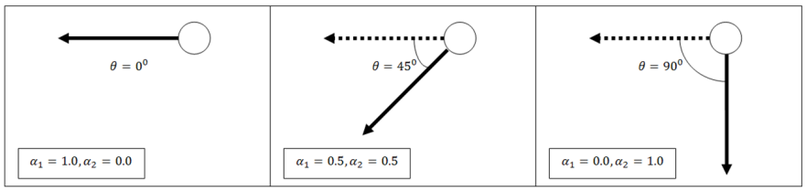



Fig. 10. Different angles given by different weights for a Bi-objective Problem.

*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 got to 8, else set $T_{i+1} = \rho \times T_i$, randomly select $s \in Q_\phi$, set $W = A(n) = \{w_1, w_2, \cdots, w_n\}$ and go to step 4.

*Step 4.* Perturbing Solutions. Set $s^{'} = P(s)$, add to $Q_\phi$ and $Q_*$ according to the next rules:

$$Q_\phi = Q_\phi \cup \left\{ s^{'} \right\} \Leftrightarrow (\nexists r \in Q_\phi)(r \quad dominated \quad to \quad s^{'}) \tag{22}$$

$$Q_* = Q_* \cup \left\{ s^{'} \right\} \Leftrightarrow (\exists r \in Q_*)(s^{'} \quad dominated \quad to \quad r) \tag{23}$$

If $Q_\phi$ has at least one element that dominated to $s^{'}$ go to step 5, otherwise go to step 7.

*Step 5.* Guess with dominated solutions. Randomly generated a number $n \in [0, 1]$. Set $z$ as follows:

$$z = e^{(-(\gamma / T_i))} \tag{24}$$

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

$$\gamma = \sum_{i=1}^{n} w_i \cdot f_i(s_X) - \sum_{i=1}^{n} w_i \cdot f_i(s^{'}{}_X) \tag{25}$$

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 $n < z$ then set $s$ as $s^{'}$ and go to step 4 else go to step 6.

*Step 6.* Change the search direction. Randomly select a solution $s \in Q_*$ and go to step 4.

*Step 7.* Removing dominated solutions. Remove the dominated solution for each set ($Q_*$ and $Q_\phi$). Go to step 3.

*Step 8.* Finishing. $Q_\phi$ has the non-dominated solutions.

As can be seen in figure 11, alike MODS, SAMODS removes the dominated solutions when the new solution found is not dominated. Besides, if the new solution found dominated at least one element from the solution set ($Q_\phi$) then it will be added to the elitisms set ($Q_*$) that works as a search direction for the Pareto Front. As far as here, SAMODS could sounds as a simple local search strategy but not, when a new solution found is dominated, SAMODS 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 12. Due to this, local optimums are avoided. When the

temperature is low, the bad solutions are avoided because $z$ value is low therefore SAMODS accepts only non-dominated solutions. However, by that time, $Q_\phi$ will be leaded on by $Q_*$.

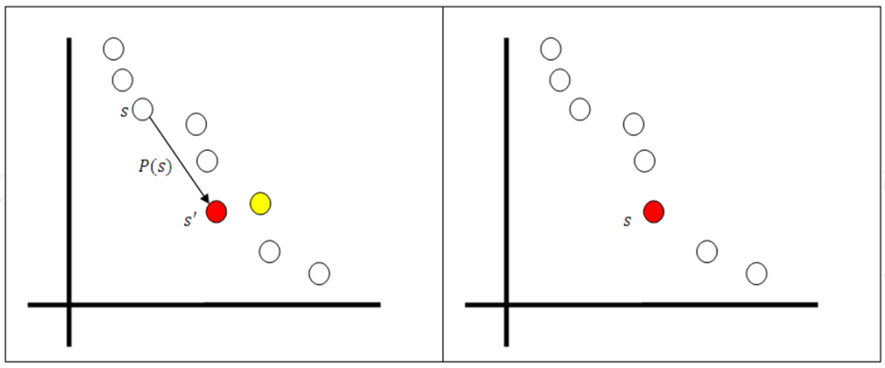



Fig. 11. Behavior of SAMODS when the new solution found is not dominated. Once a new solution found is non-dominated, it is added to the elitism set $Q_*$ and the dominated solutions from $Q_\phi$ are removed.

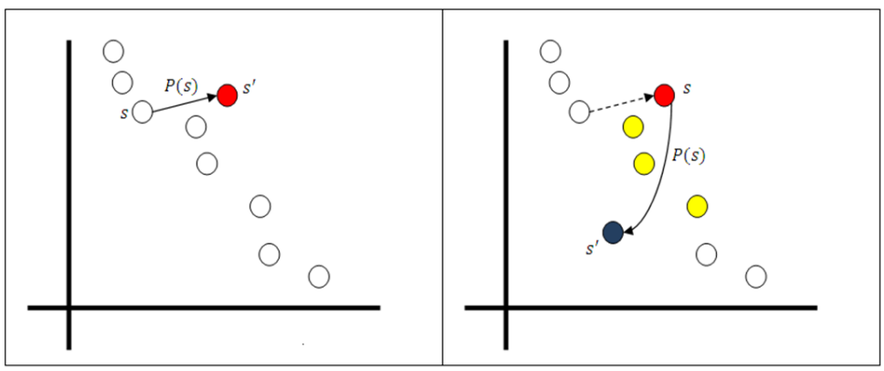



Fig. 12. Behavior of SAMODS when the new solution found is dominated. In this case, guessing gives a new solution non-dominated.

## 4. Genetic Simulated Annealing Metaheuristic Of Deterministic Swapping (SAGAMODS)

Simulated Annealing, Genetic Algorithm & Metaheuristic Of Deterministic Swapping(Niño, 2012) (SAGAMODS) is a hybrid search strategy based on the Automata Theory, Simulated Annealing and Genetics Algorithms. SAGAMODS is an extension of the SAMODS theory. It comes up as result of the next question: could SAMODS avoid quickly local optimums? Although, SAMODS avoids local optimums guessing, it can take a lot of time accepting dominated solutions for finding non-dominated. Thus, the answer to this question is based on the Evolutionary Theory. SAGAMODS proposes crossover step before SAMODS template is executed. Due to this, SAGAMODS supports to SAMODS for exploring distant regions of the solution space.

Formally, SAGAMODS is based on the next automata:

$$M = (Q, Q_S, C(q, r, k), F(X)) \tag{26}$$

Where $Q$ is the feasible solutions space, $Q_S$ is the initial solutions and $F(X)$ are the objectives of the problem. $C(q,r,k)$ is defined as follows:

Formally, *Cross Function K* is defined as follows:

$$C(q,r,k) : Q \to Q \tag{27}$$

Where $q,r \in Q$ and $k \in N$. $q$ and $r$ are named parents solutions and $k$ is the cross point. The main idea of this function is cross two solutions in the same point and returns a new solution. For instance, two solutions of 4 variables are cross in figure 13. Obviously, the crossover is made regarding the representation of the solutions. Lastly, SAGAMODS template is defined

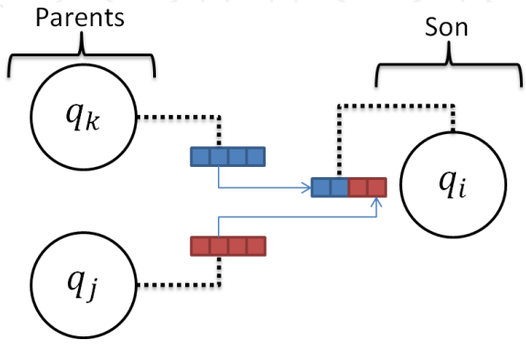



Fig. 13. Crossover between two solutions. Solutions of the states $q_k$ and $q_j$ are crossed in order to get state $q_i$

as follows:

*Step 1.* Setting parameters. Set $Q_S$ as the solution set, $x$ as the number of solutions to cross for each iteration.

*Step 2. Selection.* Set $Q_C$ (crossover set) as selection of $x$ solutions in $Q_S$, $Q_M$ (mutation set) as $\phi$ and $k$ as a random value.

*Step 3. Crossover.* For each $s_i, s_{i+1} \in Q_C / 1 \leq i < |Q_C|$:

$$Q_M = Q_M \cup \{C(s_i, s_{i+1}, k)\} \tag{28}$$

*Step 4. Mutation.* Set $Q_0$ as $Q_M$. Execute SAMODS as a local search strategy.

*Step 5.* Check stop conditions. Go to 2.

## 5. Evolutionary Metaheuristic Of Deterministic Swapping (EMODS)

Evolutionary Metaheuristic of Deterministic Swapping (EMODS), is a novel framework that allows the Multiobjective Optimization of Combinatorial Problems. Its framework is based on MODS template therefore its steps are the same: create Initial Solutions, Improve the Solutions (Optional) and Execute the Core Algorithm. Unlike SAMODS and SAGAMODS, EMODS avoids the slowly convergence of Simulated Annealing's method. EMODS explores different regions from the feasible solution space and search for non-dominated solution using Tabu Search.

The Core Algorithm is defined as follows:

*Step 1.* Set $\theta$ as the maximum number of iterations, $\beta$ as the maximum number of state selected in each iteration, $\rho$ as the maximum number of perturbations by state and $Q_\phi$ as $Q_0$

*Step 2. Selection.* Randomly select a state $q \in Q_\phi$ or $q \in Q_*$

*Step 3. Mutation - Tabu Search.* Set $N$ as the new solutions found as result of perturbing $q$. Add to $Q_\phi$ and $Q_*$ according to the next equations:

$$\left(Q_\phi = Q_\phi \cup \{q\}\right) \Longleftrightarrow \left(\nexists r \in Q_\phi / q \quad is \quad dominated \quad by \quad r\right) \tag{29}$$

$$\left(Q_* = Q_* \cup \{q\}\right) \Longleftrightarrow \left(\exists r \in Q_\phi / r \quad is \quad dominated \quad by \quad q\right) \tag{30}$$

Remove the states with dominated solutions for each set.

*Step 4. Crossover.* Randomly select states from $Q_\phi$ and $Q_*$. Generate a random point of cross.

*Step 5.* Check stop condition, go to 3.

Step 2 and 3 support the algorithm in removing dominated solutions from the set of solutions $Q_\phi$ as can be seen in figure 3. However, one of the most important steps in the EMODS algorithm is step 4. There, similar to SAGAMODS, the algorithm applies an Evolutionary Strategy based in the crossover step of Genetic Algorithms for avoiding Local Optimums. Due to the crossover is not always made in the same point (the k-value is randomly generated in each state analyzed) the variety of solutions found are diverse avoiding local optimums. An overview of EMODS behavior for a Tri-objective Combinatorial Optimization problem can be seen in figure 14

## 6. Experimental analysis

### 6.1 Experimental settings

The algorithms were tested using well-known instances from the Multi-objective Traveling Salesman Problem taken from TSPLIB(Heidelberg, n.d.). The instances worked are shown in table 6 and the input parameters for the algorithms are shown in table 7. The test of the algorithms was made using a Dual Core Computer with 2 Gb RAM. The optimal solutions were constructed based in the best non-dominated solutions of all algorithms in comparison for each instance worked.

### 6.2 Performance metrics

There are metrics that allow measuring the quality of a set of optimal solutions and the performance of an Algorithm (Corne & Knowles, 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.

$$GNDV = |PF_{know}| \tag{31}$$

A higher value for this metric is desired. *Rate of Generation of No-dominated Vectors (RGNDV)* This metric measures the proportion of the No Dominates Solutions (31) generated by an

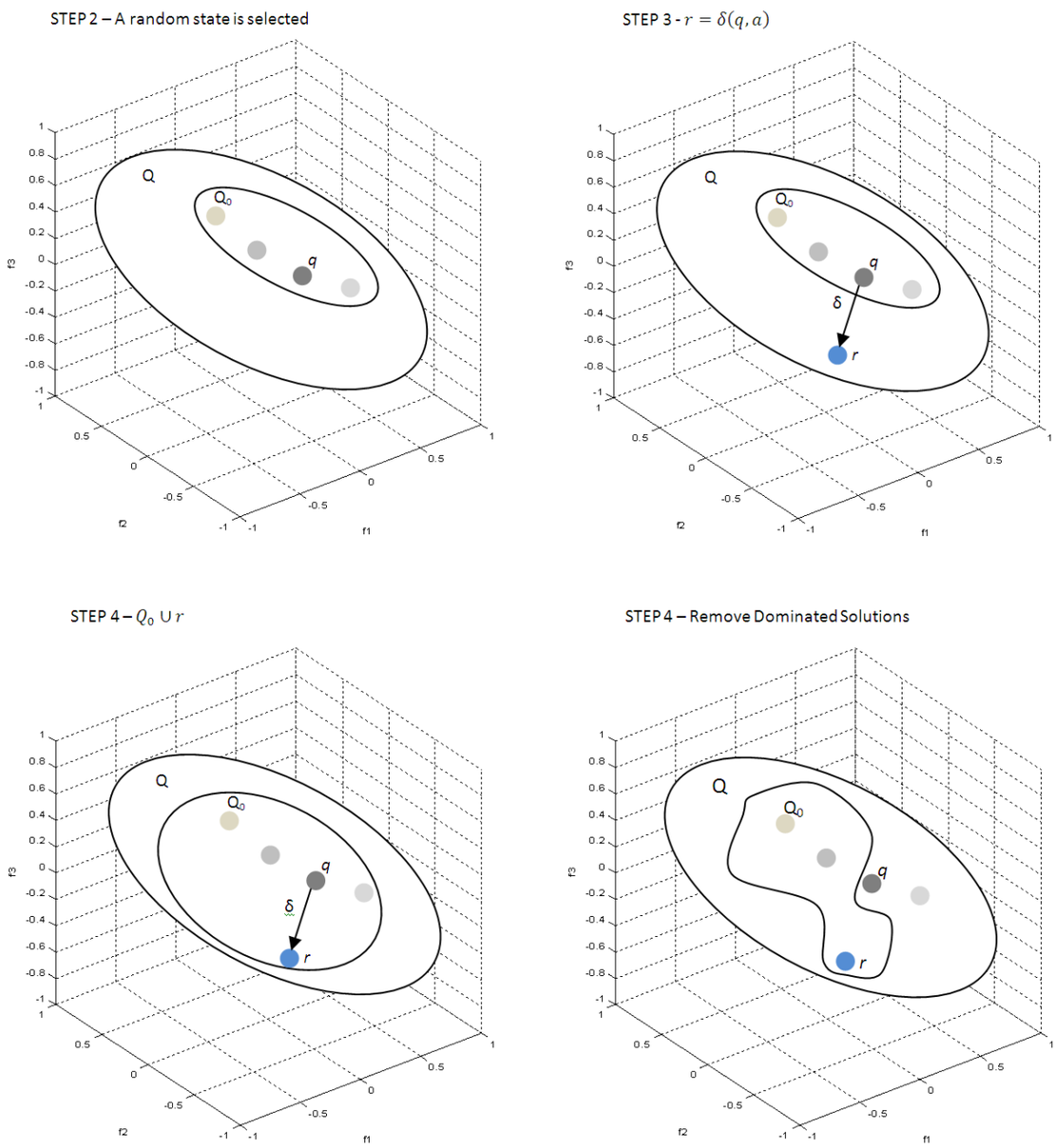



Fig. 14. An overview of EMODS behavior for a Tri-objective Problem.

algorithm and the Real Solutions.

$$RGNDV = \left(\frac{GNDV}{|PF_{true}|}\right) \cdot 100\% \tag{32}$$

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.

$$ReGNDV = |\{y | y \in PF_{know} \wedge y \in PF_{true}\}| \tag{33}$$

| Combinatorial Problem Instance | Number of Objectives |
|---|---|
| KROAB100 | 2 |
| KROAC100 | |
| KROAD100 | |
| KROAE100 | |
| KROBC100 | |
| KROBD100 | |
| KROBE100 | |
| KROCD100 | |
| KROCE100 | |
| KRODE100 | |
| KROABC100 | 3 |
| KROABD100 | |
| KROABE100 | |
| KROACD100 | |
| KROACE100 | |
| KROADE100 | |
| KROBCD100 | |
| KROBCE100 | |
| KROBDE100 | |
| KROCDE100 | |
| KROABCD100 | 4 |
| KROABCE100 | |
| KROABDE100 | |
| KROACDE100 | |
| KROBCDE100 | |
| KROABCDE100 | 5 |

Table 6. Instances worked for testing the proposed algorithms.

| Algorithm | Max. Iterations | Max. Perturbations | Initial Temperature | Cooler Value | Crossover Rate |
|---|---|---|---|---|---|
| MODS | 100 | 80 | NA | NA | NA |
| SAMODS | 100 | 80 | 1000 | 0.95 | NA |
| SAGAMODS | 100 | 80 | 1000 | 0.95 | 0.6 |
| EMODS | 100 | 80 | NA | NA | 0.6 |

Table 7. Parameters setting for each compared algorithm.

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.

$$GD = \left(\frac{1}{|PF_{know}|}\right) \cdot \left(\sum_{i=1}^{|PF_{know}|} d_i\right)^{(1/p)} \tag{34}$$

Where $d_i$ is the smallest Euclidean distance between the solution i of $FP_{know}$ and the solutions of $FP_{true}$. $p$ is the dimension of the combinatorial problem, it means the number of objective functions. *Inverse Generational Distance (IGD)* This is another distance measurement between $FP_{know}$ and $FP_{true}$:

$$IGD = \left(\frac{1}{|PF_{true}|}\right) \cdot \left(\sum_{i=1}^{|PF_{know}|} d_i\right) \tag{35}$$

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 $PF_{know}$

$$S = \left(\frac{1}{|PF_{know}| - 1}\right)^2 \cdot \left(\sum_{i=1}^{|PF_{know}|} \left(\overline{d} - d_i\right)^2\right)^{(1/p)} \tag{36}$$

Where $d_i$ is the smallest Euclidean distance between the solution $i$ of $PF_{know}$ and the rest of solutions of $PF_{know}$. $\overline{d}$ 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 (ε)* It estimates the error rate respect to the precision of the Real Algorithms Solutions (33) as follows:

$$\varepsilon = \left(\left|\frac{PF_{true}}{ReGNDV}\right|\right) \cdot 100\% \tag{37}$$

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.

### 6.3 Experimental results

The tests made with Bi-objectives, Tri-objectives, Quad-objectives and Quin-objectives TSP instances are shown in tables 8, 9, 10 and 11 respectively. The average of the measurement is shown in table 12. Furthermore, a graphical comparison for bi-objectives and tri objectives instances worked is shown in figures 15 and 16 respectively.

### 6.4 Analysis

It can be concluded, that, in the case of two and three objectives, metrics such as $S$, $IGD$, $GD$ and $\varepsilon$ determine the best algorithm. In this case, the measurement of the metrics is similar for SAMODS and SAGAMODS. On the other hand, MODS has the most poor-quality measurement for the metrics used and EMODS has the best quality measurement for the same metrics.

Lastly, why are the results of the metrics similar for quint-instances? In this case, all the solutions for each solution set are in the optimal set. The answer to this question is based in the angle improvement. MODS as a local search strategy explore a part of the feasible solution using its search direction ($Q_*$). However, SAMODS and SAGAMODS, in addition, use a search direction given by the change of the search angle. While SAMODS was looking in a

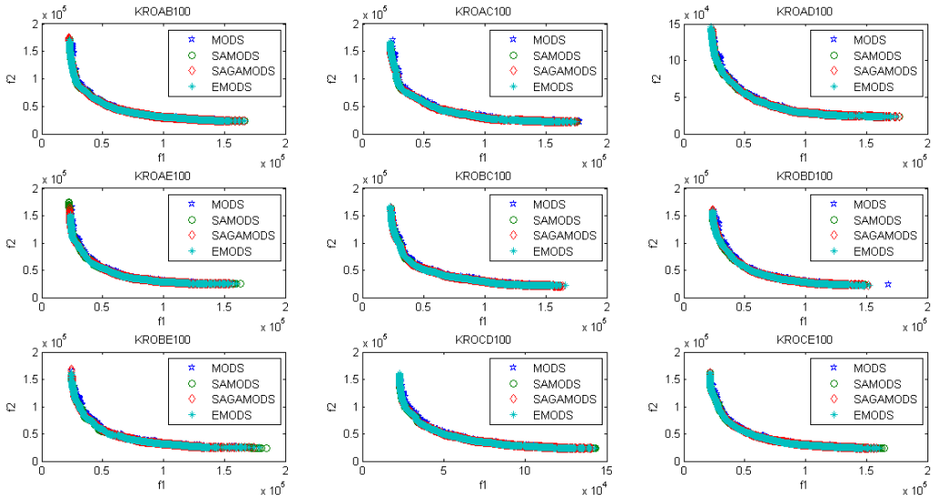



Fig. 15. Graphical comparison between MODS, SAMODS, SAGAMODS and EMODS for Bi-objective TSP instances.

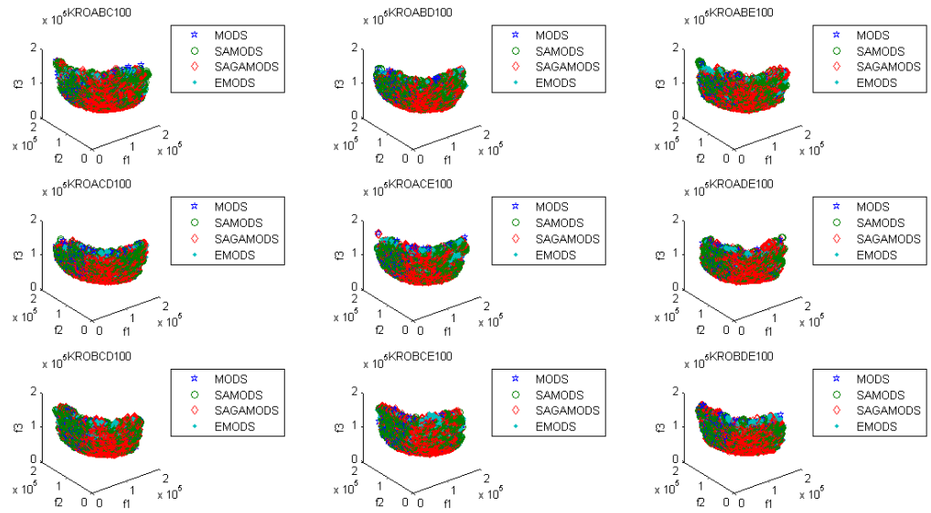



Fig. 16. Graphical comparison between MODS, SAMODS, SAGAMODS and EMODS for Tri-objective TSP instances.

part of the feasible solution space, SAGAMODS was doing the same in other. The same reason applies to EMODS. It can be possible because of the large size of the feasible solution space ($\Re^5$). The possibility of exploring the same part of the solution space for different algorithms is low.

| *Instance* | *Algorithm* | *GNDV* | *RGNDV* | *ReGNDV* | $\left(\frac{ReGNDV}{GNDV}\right)$ % | *S* | *GD* | *IGD* | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| AB | MODS | 289 | 0.0189 | 0 | 0% | 0.0193 | 21.2731 | 2473.4576 | 100% |
| | SAMODS | 7787 | 0.5096 | 1247 | 16.01% | 0.001 | 0.2404 | 229.2593 | 91.84% |
| | SAGAMODS | 8479 | 0.5549 | 2974 | 35.07% | 0.0007 | 0.1837 | 158.8229 | 80.54% |
| | EMODS | 26125 | 1.7096 | 11060 | 42.33% | 0.0002 | 0.0412 | 75.8814 | 27.62% |
| AC | MODS | 217 | 0.0155 | 0 | 0% | 0.034 | 28.575 | 2751.3232 | 100% |
| | SAMODS | 6885 | 0.4927 | 2303 | 33.45% | 0.0008 | 0.2297 | 179.0354 | 83.52% |
| | SAGAMODS | 7023 | 0.5025 | 2431 | 34.61% | 0.0008 | 0.2628 | 243.7536 | 82.6% |
| | EMODS | 20990 | 1.502 | 9241 | 44.03% | 0.0002 | 0.0617 | 119.8825 | 33.87% |
| AD | MODS | 281 | 0.0187 | 0 | 0% | 0.0139 | 20.4429 | 2198.883 | 100% |
| | SAMODS | 6383 | 0.4253 | 1464 | 22.94% | 0.0029 | 0.3188 | 275.9153 | 90.24% |
| | SAGAMODS | 6289 | 0.4191 | 764 | 12.15% | 0.0016 | 0.2835 | 211.7935 | 94.91% |
| | EMODS | 17195 | 1.1458 | 12779 | 74.32% | 0.0005 | 0.0521 | 53.5314 | 14.85% |
| AE | MODS | 283 | 0.0189 | 0 | 0% | 0.0533 | 21.3238 | 2433.63 | 100% |
| | SAMODS | 5693 | 0.3804 | 1433 | 25.17% | 0.0016 | 0.4308 | 402.0483 | 90.42% |
| | SAGAMODS | 6440 | 0.4304 | 1515 | 23.52% | 0.0013 | 0.3906 | 422.7859 | 89.88% |
| | EMODS | 20695 | 1.383 | 12016 | 58.06% | 0.0002 | 0.066 | 124.7197 | 19.7% |
| BC | MODS | 298 | 0.0212 | 0 | 0% | 0.0158 | 19.537 | 2411.8365 | 100% |
| | SAMODS | 6858 | 0.488 | 789 | 11.5% | 0.0024 | 0.3597 | 433.0882 | 94.39% |
| | SAGAMODS | 6919 | 0.4923 | 2201 | 31.81% | 0.0015 | 0.2378 | 192.5601 | 84.34% |
| | EMODS | 21902 | 1.5584 | 11064 | 50.52% | 0.0003 | 0.0582 | 115.5673 | 21.28% |
| BD | MODS | 241 | 0.0198 | 0 | 0% | 0.0239 | 21.7441 | 2251.0972 | 100% |
| | SAMODS | 6844 | 0.561 | 2054 | 30.01% | 0.0021 | 0.2542 | 248.0971 | 83.16% |
| | SAGAMODS | 5934 | 0.4864 | 1971 | 33.22% | 0.0018 | 0.2818 | 229.2093 | 83.84% |
| | EMODS | 19420 | 1.5919 | 8174 | 42.09% | 0.0003 | 0.0432 | 57.6434 | 32.99% |
| BE | MODS | 280 | 0.0259 | 0 | 0% | 0.0309 | 19.0193 | 2622.5243 | 100% |
| | SAMODS | 6260 | 0.5789 | 952 | 15.21% | 0.001 | 0.2601 | 245.1433 | 91.2% |
| | SAGAMODS | 5802 | 0.5365 | 1622 | 27.96% | 0.0025 | 0.3912 | 476.4848 | 85% |
| | EMODS | 17362 | 1.6055 | 8240 | 47.46% | 0.0004 | 0.0631 | 111.0209 | 23.8% |
| CD | MODS | 286 | 0.022 | 0 | 0% | 0.0184 | 18.0035 | 2040.9722 | 100% |
| | SAMODS | 6171 | 0.4751 | 1912 | 30.98% | 0.0007 | 0.2588 | 196.3394 | 85.28% |
| | SAGAMODS | 6301 | 0.4851 | 994 | 15.78% | 0.0014 | 0.2852 | 248.5855 | 92.35% |
| | EMODS | 18628 | 1.434 | 10084 | 54.13% | 0.0002 | 0.0426 | 48.3785 | 22.37% |
| CE | MODS | 224 | 0.0187 | 0 | 0% | 0.0285 | 23.1312 | 2245.6542 | 100% |
| | SAMODS | 5881 | 0.4919 | 946 | 16.09% | 0.0017 | 0.2894 | 242.2535 | 92.09% |
| | SAGAMODS | 4613 | 0.3859 | 939 | 20.36% | 0.0028 | 0.481 | 411.854 | 92.15% |
| | EMODS | 15211 | 1.2724 | 10070 | 66.2% | 0.0003 | 0.0339 | 22.2645 | 15.77% |
| DE | MODS | 228 | 0.0147 | 0 | 0% | 0.0477 | 23.6222 | 1864.9602 | 100% |
| | SAMODS | 6110 | 0.3928 | 1157 | 18.94% | 0.0022 | 0.2942 | 207.7947 | 92.56% |
| | SAGAMODS | 7745 | 0.4979 | 407 | 5.26% | 0.0012 | 0.2644 | 269.6204 | 97.38% |
| | EMODS | 20058 | 1.2896 | 13990 | 69.75% | 0.0005 | 0.0304 | 23.8829 | 10.06% |

Table 8. Measuring algorithms performance for Bi-objectives instances of Traveling Salesman Problem with Multi-objective optimization metrics.

| *Instance* | *Algorithm* | *GNDV* | *RGNDV* | *ReGNDV* | $\left(\frac{ReGNDV}{GNDV}\right)\%$ | *S* | *GD* | *IGD* | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| ABC | MODS | 2115 | 0.0307 | 83 | 3.92% | 0.1567 | 0.2819 | 3075.4309 | 99.88% |
| | SAMODS | 12768 | 0.1853 | 227 | 1.78% | 0.0722 | 0.0421 | 2256.4593 | 99.67% |
| | SAGAMODS | 12523 | 0.1818 | 328 | 2.62% | 0.073 | 0.0427 | 2220.5614 | 99.52% |
| | EMODS | 70474 | 1.023 | 68254 | 96.85% | 0.0477 | 0.001 | 5.6388 | 0.93% |
| ABD | MODS | 1951 | 0.0292 | 74 | 3.79% | 0.1524 | 0.305 | 3153.9212 | 99.89% |
| | SAMODS | 12094 | 0.1811 | 317 | 2.62% | 0.0746 | 0.0441 | 2270.3475 | 99.53% |
| | SAGAMODS | 12132 | 0.1817 | 250 | 2.06% | 0.0726 | 0.0441 | 2286.7374 | 99.63% |
| | EMODS | 68001 | 1.0184 | 66133 | 97.25% | 0.0471 | 0.0011 | 6.3212 | 0.96% |
| ABE | MODS | 1931 | 0.0281 | 63 | 3.26% | 0.1496 | 0.315 | 3278.1554 | 99.91% |
| | SAMODS | 12461 | 0.1815 | 373 | 2.99% | 0.0743 | 0.0438 | 2371.7641 | 99.46% |
| | SAGAMODS | 12391 | 0.1805 | 370 | 2.99% | 0.0745 | 0.0436 | 2304.3277 | 99.46% |
| | EMODS | 70411 | 1.0257 | 67839 | 96.35% | 0.0474 | 0.0012 | 8.0639 | 1.17% |
| ACD | MODS | 2031 | 0.0305 | 66 | 3.25% | 0.1425 | 0.2945 | 3213.7378 | 99.9% |
| | SAMODS | 12004 | 0.1802 | 241 | 2.01% | 0.0734 | 0.0444 | 2277.0343 | 99.64% |
| | SAGAMODS | 12123 | 0.182 | 206 | 1.7% | 0.0735 | 0.0442 | 2310.3683 | 99.69% |
| | EMODS | 67451 | 1.0127 | 66090 | 97.98% | 0.0468 | 0.001 | 4.5012 | 0.77% |
| ACE | MODS | 1950 | 0.0306 | 57 | 2.92% | 0.1628 | 0.3024 | 3215.8357 | 99.91% |
| | SAMODS | 11382 | 0.1785 | 263 | 2.31% | 0.074 | 0.0461 | 2271.6542 | 99.59% |
| | SAGAMODS | 11476 | 0.18 | 303 | 2.64% | 0.0734 | 0.0456 | 2241.9933 | 99.52% |
| | EMODS | 64804 | 1.0162 | 63145 | 97.44% | 0.048 | 0.0012 | 7.3103 | 0.98% |
| ADE | MODS | 1824 | 0.0274 | 67 | 3.67% | 0.1487 | 0.3289 | 3248.597 | 99.9% |
| | SAMODS | 12149 | 0.1827 | 179 | 1.47% | 0.0733 | 0.0442 | 2336.2798 | 99.73% |
| | SAGAMODS | 11773 | 0.1771 | 258 | 2.19% | 0.0771 | 0.0457 | 2346.6414 | 99.61% |
| | EMODS | 67767 | 1.0193 | 65981 | 97.36% | 0.0468 | 0.0011 | 5.7824 | 0.76% |
| BCD | MODS | 2065 | 0.03 | 43 | 2.08% | 0.1451 | 0.2927 | 3206.9305 | 99.94% |
| | SAMODS | 13129 | 0.1908 | 260 | 1.98% | 0.0712 | 0.0417 | 2387.8219 | 99.62% |
| | SAGAMODS | 12889 | 0.1873 | 253 | 1.96% | 0.0786 | 0.042 | 2308.1811 | 99.63% |
| | EMODS | 70035 | 1.0176 | 68270 | 97.48% | 0.0452 | 0.001 | 5.6235 | 0.81% |
| BCE | MODS | 2009 | 0.0286 | 58 | 2.89% | 0.1505 | 0.3065 | 3327.787 | 99.92% |
| | SAMODS | 12992 | 0.1852 | 229 | 1.76% | 0.0701 | 0.0428 | 2448.3577 | 99.67% |
| | SAGAMODS | 12582 | 0.1794 | 201 | 1.6% | 0.0736 | 0.0445 | 2503.0421 | 99.71% |
| | EMODS | 71176 | 1.0147 | 69654 | 97.86% | 0.0464 | 0.0011 | 7.8122 | 0.7% |
| BDE | MODS | 2039 | 0.0316 | 45 | 2.21% | 0.1532 | 0.2914 | 3252.7813 | 99.93% |
| | SAMODS | 12379 | 0.192 | 205 | 1.66% | 0.0728 | 0.0434 | 2401.8804 | 99.68% |
| | SAGAMODS | 12427 | 0.1928 | 195 | 1.57% | 0.0742 | 0.0431 | 2377.0621 | 99.7% |
| | EMODS | 65509 | 1.0163 | 64015 | 97.72% | 0.0476 | 0.0011 | 5.6322 | 0.69% |
| CDE | MODS | 2010 | 0.0278 | 83 | 4.13% | 0.1463 | 0.3022 | 3094.2824 | 99.89% |
| | SAMODS | 13084 | 0.1807 | 399 | 3.05% | 0.0712 | 0.0414 | 2193.6586 | 99.45% |
| | SAGAMODS | 13009 | 0.1796 | 347 | 2.67% | 0.0719 | 0.0418 | 2224.4738 | 99.52% |
| | EMODS | 74063 | 1.0227 | 71589 | 96.66% | 0.0453 | 0.0011 | 7.2279 | 1.14% |

Table 9. Measuring algorithms performance for Tri-objective instances of TSP with Multi-objective optimization metrics.

| *Instance* | *Algorithm* | *GNDV* | *RGNDV* | *ReGNDV* | $\left(\frac{ReGNDV}{GNDV}\right)$ % | *S* | *GD* | *IGD* | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| ABCD | MODS | 5333 | 0.0925 | 3303 | 61.94% | 0.3497 | 0.0256 | 6030.5288 | 94.27% |
| | SAMODS | 28523 | 0.4947 | 12178 | 42.7% | 0.231 | 0.0042 | 3454.1238 | 78.88% |
| | SAGAMODS | 36802 | 0.6382 | 14967 | 40.67% | 0.2203 | 0.0031 | 3092.5232 | 74.04% |
| | EMODS | 201934 | 3.502 | 27214 | 13.48% | 0.1754 | 0.0005 | 1991.148 | 52.8% |
| ABCE | MODS | 5533 | 0.0973 | 3439 | 62.15% | 0.3452 | 0.0244 | 5861.8605 | 93.95% |
| | SAMODS | 27684 | 0.4868 | 11471 | 41.44% | 0.2331 | 0.0043 | 3444.6454 | 79.83% |
| | SAGAMODS | 35766 | 0.6289 | 14552 | 40.69% | 0.2232 | 0.0032 | 3118.6397 | 74.41% |
| | EMODS | 204596 | 3.5976 | 27408 | 13.4% | 0.1754 | 0.0005 | 1885.4464 | 51.81% |
| ABDE | MODS | 5259 | 0.0942 | 3142 | 59.75% | 0.3487 | 0.0256 | 5864.5036 | 94.37% |
| | SAMODS | 27180 | 0.4869 | 11247 | 41.38% | 0.232 | 0.0043 | 3398.9429 | 79.85% |
| | SAGAMODS | 34930 | 0.6257 | 14472 | 41.43% | 0.2236 | 0.0033 | 2986.5319 | 74.08% |
| | EMODS | 195756 | 3.5067 | 26963 | 13.77% | 0.1775 | 0.0005 | 1916.7141 | 51.7% |
| ACDE | MODS | 5466 | 0.094 | 3400 | 62.2% | 0.3405 | 0.0246 | 5617.5202 | 94.15% |
| | SAMODS | 26757 | 0.4602 | 11336 | 42.37% | 0.235 | 0.0044 | 3394.8396 | 80.5% |
| | SAGAMODS | 34492 | 0.5932 | 14638 | 42.44% | 0.2265 | 0.0033 | 2965.4482 | 74.83% |
| | EMODS | 196800 | 3.3845 | 28774 | 14.62% | 0.1764 | 0.0005 | 1793.4489 | 50.52% |
| BCDE | MODS | 5233 | 0.0879 | 3082 | 58.9% | 0.3499 | 0.0259 | 5677.9988 | 94.83% |
| | SAMODS | 28054 | 0.471 | 11739 | 41.84% | 0.2315 | 0.0042 | 3296.196 | 80.29% |
| | SAGAMODS | 36258 | 0.6087 | 15145 | 41.77% | 0.2218 | 0.0032 | 2902.8041 | 74.57% |
| | EMODS | 203017 | 3.4083 | 29599 | 14.58% | 0.1752 | 0.0005 | 1873.1748 | 50.31% |

Table 10. Measuring algorithms performance for Quad-objectives instances of TSP with Multi-objective optimization metrics.

| *Instance* | *Algorithm* | *GNDV* | *RGNDV* | *ReGNDV* | $\left(\frac{ReGNDV}{GNDV}\right)$ % | *S* | *GD* | *IGD* | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| ABCDE | MODS | 7517 | 0.0159 | 7517 | 100% | 0.5728 | 0.0125 | 15705.6864 | 98.41% |
| | SAMODS | 26140 | 0.0554 | 26140 | 100% | 0.4101 | 0.0033 | 10801.6382 | 94.46% |
| | SAGAMODS | 26611 | 0.0564 | 26611 | 100% | 0.4097 | 0.0033 | 10544.8901 | 94.36% |
| | EMODS | 411822 | 0.8723 | 411822 | 100% | 0.3136 | 0.0001 | 950.4252 | 12.77% |

Table 11. Measuring algorithms performance for Quint-objectives instances of TSP with Multi-objective optimization metrics.

| *Objectives* | *Algorithm* | *GNDV* | *RGNDV* | *ReGNDV* | $\left(\frac{ReGNDV}{GNDV}\right)$ % | *S* | *GD* | *IGD* | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | MODS | 262.7 | 0.0194 | 0 | 0% | 0.0286 | 21.6672 | 2329.4338 | 100% |
| | SAMODS | 6487.2 | 0.4796 | 1425.7 | 22.03% | 0.0016 | 0.2936 | 265.8974 | 89.47% |
| | SAGAMODS | 6554.5 | 0.4791 | 1581.8 | 23.97% | 0.0015 | 0.3062 | 286.547 | 88.3% |
| | EMODS | 19758.6 | 1.4492 | 10671.8 | 54.89% | 0.0003 | 0.0492 | 75.2773 | 22.23% |
| 3 | MODS | 1992.5 | 0.0295 | 63.9 | 3.21% | 0.1508 | 0.302 | 3206.7459 | 99.91% |
| | SAMODS | 12444.2 | 0.1838 | 269.3 | 2.16% | 0.0727 | 0.0434 | 2321.5258 | 99.6% |
| | SAGAMODS | 12332.5 | 0.1822 | 271.1 | 2.2% | 0.0743 | 0.0437 | 2312.3389 | 99.6% |
| | EMODS | 68969.1 | 1.0187 | 67097 | 97.3% | 0.0468 | 0.0011 | 6.3914 | 0.89% |
| 4 | MODS | 5364.8 | 0.0932 | 3273.2 | 60.99% | 0.3468 | 0.0252 | 5810.4824 | 94.31% |
| | SAMODS | 27639.6 | 0.4799 | 11594.2 | 41.94% | 0.2325 | 0.0043 | 3397.7495 | 79.87% |
| | SAGAMODS | 35649.6 | 0.619 | 14754.8 | 41.4% | 0.2231 | 0.0032 | 3013.1894 | 74.39% |
| | EMODS | 200420.6 | 3.4798 | 27991.6 | 13.97% | 0.176 | 0.0005 | 1891.9864 | 51.43% |
| 5 | MODS | 7517 | 0.0159 | 7517 | 100% | 0.5728 | 0.0125 | 15705.6864 | 98.41% |
| | SAMODS | 26140 | 0.0554 | 26140 | 100% | 0.4101 | 0.0033 | 10801.6382 | 94.46% |
| | SAGAMODS | 26611 | 0.0564 | 26611 | 100% | 0.4097 | 0.0033 | 10544.8901 | 94.36% |
| | EMODS | 411822 | 0.8723 | 411822 | 100% | 0.3136 | 0.0001 | 950.4252 | 12.77% |

Table 12. Measuring algorithms performance for Multi-objectives instances of TSP with Multi-objective optimization metrics.

## 7. Conclusion

SAMODS, SAGAMODS and EMODS are algorithms based on the Automata Theory for the Multi-objective Optimization of Combinatorial Problems. All of them are derived from the MODS metaheuristic, which is inspired in the Theory of Deterministic Finite Swapping. SAMODS is a Simulated Annealing inspired Algorithm. It uses a search direction in order to optimize a set of solution (Pareto Front) through a linear combination of the objective functions. On the other hand, SAGAMODS, in addition to the advantages of SAMODS, is an Evolutionary inspired Algorithm. It implements a crossover step for exploring far regions of a solution space. Due to this, SAGAMODS tries to avoid local optimums owing to it takes a general look of the solution space. Lastly, in order to avoid slow convergence, EMODS is proposed. Unlike SAMODS and SAGAMODS, EMODS does not explore the neighborhood of a solution using Simulated Annealing, this step is done using Tabu Search. Thus, EMODS gets optimal solution faster than SAGAMODS and SAMODS. Lastly, the algorithms were tested using well known instances from TSPLIB and metrics from the specialized literature. The results shows that for instances of two, three and four objectives, the proposed algorithm has the best performance as the metrics values corroborate. For the last instance worked, quint-objective, the behavior of MODS, SAMODS and SAGAMODS tend to be the same, them have similar error rate but, EMODS has a the best performance. In all the cases, EMODS shows the best performance. However, for the last test, all the algorithms have different solutions sets of non-dominated solutions, and those form the optimal solution set.

## 8. Acknowledgment

First of all, I want to thank to God for being with me in my entire life, He made this possible. Secondly, I want to thank to my parents Elias Niño and Arely Ruiz and my sister Carmen Niño for their enormous love and support. Finally, and not less important, to thank to my beautiful wife Maria Padron and our baby for being my inspiration.

## 9. References

An, S., Yang, S., Ho, S., Li, T. & Fu, W. (2011). A modified tabu search method applied to inverse problems, *Magnetics, IEEE Transactions on* 47(5): 1234 –1237.

Araujo, D., Bastos-Filho, C., Barboza, E., Chaves, D. & Martins-Filho, J. (2011). A performance comparison of multi-objective optimization evolutionary algorithms for all-optical networks design, *Computational Intelligence in Multicriteria Decision-Making (MDCM), 2011 IEEE Symposium on*, pp. 89 –96.

Ausavanop, O. & Chaitusaney, S. (2011). Coordination of dispatchable distributed generation and voltage control devices for improving voltage profile by tabu search, *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on*, pp. 869 –872.

Azuma, R. M., Coelho, G. P. & Von Zuben, F. J. (2011). Evolutionary multi-objective optimization for the vendor-managed inventory routing problem, *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1457 –1464.

Chen, K.-Y., Liu, A. & Lee, C.-H. (2003). A multiprocessor real-time process scheduling method, *Multimedia Software Engineering, 2003. Proceedings. Fifth International Symposium on*, pp. 29 – 36.

Chipara, O., Lu, C. & Roman, G.-C. (2007). Real-time query scheduling for wireless sensor networks, *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pp. 389 –399.

Chua, T., Wang, F., Cai, T. & Yin, X. (2006). A heuristics-based advanced planning and scheduling system with bottleneck scheduling algorithm, *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, pp. 240 –247.

Chunyue, Y., Meirong, X. & Ruiguo, Z. (2009). Single-machine scheduling problem in plate hot rolling production, *Control and Decision Conference, 2009. CCDC '09. Chinese*, pp. 2500 –2503.

Corne, D. & Knowles, J. (2003). Some multiobjective optimizers are better than others, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, Vol. 4, pp. 2506 – 2512 Vol.4.

Czabaski, R. (2006). Deterministic annealing integrated with insensitive learning in neuro-fuzzy systems, *in* L. Rutkowski, R. Tadeusiewicz, L. Zadeh & J. Zurada (eds), *Artificial Intelligence and Soft Computing ICAISC 2006*, Vol. 4029 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 220–229.

de Pablo, D. (2009). On scheduling models: An overview, *Computers Industrial Engineering, 2009. CIE 2009. International Conference on*, pp. 153 –158.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii, *Evolutionary Computation, IEEE Transactions on* 6(2): 182 –197.

Dereli, T. & Sena Da, G. (2007). A hybrid simulated-annealing algorithm for two-dimensional strip packing problem, *in* B. Beliczynski, A. Dzielinski, M. Iwanowski & B. Ribeiro (eds), *Adaptive and Natural Computing Algorithms*, Vol. 4431 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 508–516.

Dukkipati, A. & Narasimha Murty, M. (2002). Selection by parts: 'selection in two episodes' in evolutionary algorithms, *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, Vol. 1, pp. 657 –662.

Fei, X., Ke, W., Jidong, S., Zheng, X. & Guilan, L. (2011). Back analysis of displacement based on support vector machine and continuous tabu search, *Electric Technology and Civil Engineering (ICETCE), 2011 International Conference on*, pp. 2016 –2019.

Fisher, R. (1930). *The genetical theory of natural selection*, Clarendon Press, Oxford.

Fung, R., Tang, J. & Zhang, J. (2009). A multi-depot vehicle routing problem with weight-related costs, *Computers Industrial Engineering, 2009. CIE 2009. International Conference on*, pp. 1028 –1033.

Gacto, M., Alcala, R. & Herrera, F. (2011). Evolutionary multi-objective algorithm to effectively improve the performance of the classic tuning of fuzzy logic controllers for a heating, ventilating and air conditioning system, *Genetic and Evolutionary Fuzzy Systems (GEFS), 2011 IEEE 5th International Workshop on*, pp. 73 –80.

Garriz, J., Puntonet, C., Morales, J. & delaRosa, J. (2005). Simulated annealing based-ga using injective contrast functions for bss, *in* V. Sunderam, G. van Albada, P. Sloot & J. Dongarra (eds), *Computational Science ICCS 2005*, Vol. 3514 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 505–600.

Glover, F. & Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Norwell, MA, USA.

Grobler, J. & Engelbrecht, A. (2007). A scheduling-specific modeling approach for real world scheduling, *Industrial Engineering and Engineering Management, 2007 IEEE International Conference on*, pp. 85 –89.

Hamidzadeh, B. & Atif, Y. (1996). Dynamic scheduling of real-time aperiodic tasks on multiprocessor architectures, *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on ,*, Vol. 1, pp. 469 –478 vol.1.

Heidelberg, U. O. (n.d.). Tsplib - office research group discrete optimization - university of heidelberg, URL: *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/*.

Hu, B. & Raidl, G. (2008). Solving the railway traveling salesman problem via a transformation into the classical traveling salesman problem, *Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on*, pp. 73 –77.

Huang, Y., Brocco, A., Bessis, N., Kuonen, P. & Hirsbrunner, B. (2010). Community-aware scheduling protocol for grids, *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 334 –341.

Ishibuchi, H., Akedo, N., Ohyanagi, H. & Nojima, Y. (2011). Behavior of emo algorithms on many-objective optimization problems with correlated objectives, *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1465 –1472.

Jian-guang, W., Ran, T. & Zhi-Yong, L. (2011). An improving tabu search algorithm for intrusion detection, *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on*, Vol. 1, pp. 435 –439.

Jianghong, D., Zhongyang, X., Hao, C. & Hui, D. (2000). Scheduling algorithm for mpeg-2 ts multiplexers in catv networks, *Broadcasting, IEEE Transactions on* 46(4): 249 –255.

Kim, G. H. & Lee, C. (1998). Genetic reinforcement learning approach to the heterogeneous machine scheduling problem, *Robotics and Automation, IEEE Transactions on* 14(6): 879 –893.

Kim, I. & Lipasti, M. (2003). Macro-op scheduling: relaxing scheduling loop constraints, *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 277 – 288.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by Simulated Annealing, *Science, Number 4598, 13 May 1983* 220, 4598: 671–680.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4175*

Lee, H.-C., Keh, H.-C., Huang, N.-C. & Chang, W.-H. (2011). An application of google map and tabu-search algorithm for traveling salesman problem on tel-home care, *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*, pp. 4764 –4767.

Lim, A. & Wang, F. (2005). Multi-depot vehicle routing problem: a one-stage approach, *Automation Science and Engineering, IEEE Transactions on* 2(4): 397–402.

Liu, J., Hamdi, M. & Hu, Q. (2003). Distributed parallel scheduling algorithms for high speed virtual output queuing switches, *Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on*, p. 27.

Malik, A. M. A., Othman, A. K., Ayob, M. & Hamdan, A. R. (2011). Hybrid integrated two-stage multi-neighbourhood tabu search-emcq technique for examination timetabling problem, *Data Mining and Optimization (DMO), 2011 3rd Conference on*, pp. 232 –236.

Manimaran, G., Shashidhar, M., Manikutty, A. & Murthy, C. (1997). Integrated scheduling of tasks and messages in distributed real-time systems, *Parallel and Distributed Real-Time Systems, 1997. Proceedings of the Joint Workshop on*, pp. 64 –71.

McClymont, K. & Keedwell, E. (2011). Benchmark multi-objective optimisation test problems with mixed encodings, *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 2131 –2138.

McGarry, M., Reisslein, M., Colbourn, C., Maier, M., Aurzada, F. & Scheutzow, M. (2008). Just-in-time scheduling for multichannel epons, *Lightwave Technology, Journal of* 26(10): 1204 –1216.

Mingsheng, S., Shixin, S. & Qingxian, W. (2003). An efficient parallel scheduling algorithm of dependent task graphs, *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pp. 595 – 598.

Niño, E. D. (2012). Samods and sagamods: Novel algorithms based on the automata theory for the multi-objective optimization of combinatorial problems, *International Journal of Artificial Intelligence - Special issue of IJAI on Metaheuristics in Artificial Intelligence* Pending: Pending.

Niño, E. D., Ardila, C., Donoso, Y. & Jabba, D. (2010). A novel algorithm based on deterministic finite automaton for solving the mono-objective symmetric traveling salesman problem, *International Journal of Artificial Intelligence* 5(A10): 101 – 108.

Niño, E. D., Ardila, C., Donoso, Y., Jabba, D. & Barrios, A. (2011). Mods: A novel metaheuristic of deterministic swapping for the multi Ú objective optimization of combinatorials problems, *Computer Technology and Application* 2(4): 280 – 292.

Niño, E. D., Ardila, C., Perez, A. & Donoso, Y. (2010). A genetic algorithm for multiobjective hard scheduling optimization, *International Journal of Computers Communications & Control* 5(5): 825–836.

Niehaus, D., Ramamritham, K., Stankovic, J., Wallace, G., Weems, C., Burleson, W. & Ko, J. (1993). The spring scheduling co-processor: Design, use, and performance, *Real-Time Systems Symposium, 1993., Proceedings.*, pp. 106 –111.

Oberlin, P., Rathinam, S. & Darbha, S. (2009). A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem, *American Control Conference, 2009. ACC '09.*, pp. 1292 –1297.

Ren, W.-J., Duan, J.-H., rong Zhang, F., yan Han, H. & Zhang, M. (2011). Hybrid tabu search algorithm for bi-criteria no-idle permutation flow shop scheduling problem, *Control and Decision Conference (CCDC), 2011 Chinese*, pp. 1699 –1702.

Sauer, J. & Coelho, L. (2008). Discrete differential evolution with local search to solve the traveling salesman problem: Fundamentals and case studies, *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*, pp. 1 –6.

Shah, S., Mahmood, A. & Oxley, A. (2009). Hybrid scheduling and dual queue scheduling, *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 539 –543.

Shanmugapriya, R., Padmavathi, S. & Shalinie, S. (2009). Contention awareness in task scheduling using tabu search, *Advance Computing Conference, 2009. IACC 2009. IEEE International*, pp. 272 –277.

Sofianopoulos, S. & Tambouratzis, G. (2011). Studying the spea2 algorithm for optimising a pattern-recognition based machine translation system, *Computational Intelligence in Multicriteria Decision-Making (MDCM), 2011 IEEE Symposium on*, pp. 97 –104.

Song, J., Yang, F., Wang, M. & Zhang, H. (2008). Cryptanalysis of transposition cipher using simulated annealing genetic algorithm, *in* L. Kang, Z. Cai, X. Yan & Y. Liu (eds), *Advances in Computation and Intelligence*, Vol. 5370 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 795–802.

Song, S., Hwang, K. & Kwok, Y.-K. (2006). Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling, *Computers, IEEE Transactions on* 55(6): 703 –719.

Talbi, N. & Belarbi, K. (2011a). Evolving fuzzy inference system by tabu search algorithm and its application to control, *Multimedia Computing and Systems (ICMCS), 2011 International Conference on*, pp. 1 –6.

Talbi, N. & Belarbi, K. (2011b). A self organized fuzzy pd controller using tabu search, *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pp. 460 –464.

Taspinar, N., Kalinli, A. & Yildirim, M. (2011). Partial transmit sequences for papr reduction using parallel tabu search algorithm in ofdm systems, *Communications Letters, IEEE* PP(99): 1 –3.

Wang, S.-Q. & Xu, Z.-Y. (2009). Ant colony algorithm approach for solving traveling salesman with multi-agent, *Information Engineering, 2009. ICIE '09. WASE International Conference on*, Vol. 1, pp. 381 –384.

Wang, Y. & Lang, M. (2008). Study on the model and tabu search algorithm for delivery and pickup vehicle routing problem with time windows, *Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. IEEE International Conference on*, Vol. 1, pp. 1464 –1469.

Wei, Y., Gu, K., Liu, H. & Li, D. (2007). Contract net based scheduling approach using interactive bidding for dynamic job shop scheduling, *Integration Technology, 2007. ICIT '07. IEEE International Conference on*, pp. 281 –286.

Wijkman, P. (1996). Evolutionary computation and the principle of natural selection, *Intelligent Information Systems, 1996., Australian and New Zealand Conference on*, pp. 292 –297.

Wismans, L., Van Berkum, E. & Bliemer, M. (2011). Comparison of evolutionary multi objective algorithms for the dynamic network design problem, *Networking, Sensing and Control (ICNSC), 2011 IEEE International Conference on*, pp. 275 –280.

Wu, D. & Negi, R. (2003). Downlink scheduling in a cellular network for quality of service assurance, *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, Vol. 3, pp. 1391 – 1395 Vol.3.

Xiawen, Y. & Yu, S. (2011). A real-coded quantum clone multi-objective evolutionary algorithm, *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, pp. 4683 –4687.

Yingzi, W., Xinli, J., Pingbo, H. & Kanfeng, G. (2009). Pattern driven dynamic scheduling approach using reinforcement learning, *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on*, pp. 514 –519.

Yong-Fa, Q. & Ming-Yang, Z. (2004). Research on a new multiobjective combinatorial optimization algorithm, *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pp. 187 –191.

You-xin, M., Jie, Z. & Zhuo, C. (2009). An overview of ant colony optimization algorithm and its application on production scheduling, *Innovation Management, 2009. ICIM '09. International Conference on*, pp. 135 –138.

Yu, G., Chai, T. & Luo, X. (2011). Multiobjective production planning optimization using hybrid evolutionary algorithms for mineral processing, *Evolutionary Computation, IEEE Transactions on* 15(4): 487 –514.

Yu, L., Ohsato, A., Kawakami, T. & Sekiguchi, T. (1999). Corba-based design and development of distributed scheduling systems: an application to flexible flow shop scheduling systems, *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, Vol. 4, pp. 522 –527 vol.4.

Zhang-liang, W. & Yue-guang, L. (2011). An ant colony algorithm with tabu search and its application, *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, Vol. 2, pp. 412 –416.

Zhao, Y., Jia, F., Wang, G. & Wang, L. (2011). A hybrid tabu search for steelmaking-continuous casting production scheduling problem, *Advanced Control of Industrial Processes (ADCONIP), 2011 International Symposium on*, pp. 535 –540.

Zheng, D., Liu, J., Chen, L., Liu, Y. & Guo, W. (2011). Transmitter-receiver collaborative-relay beamforming by simulated annealing, *in* Y. Tan, Y. Shi, Y. Chai & G. Wang (eds), *Advances in Swarm Intelligence*, Vol. 6729 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 411–418.

Zhu, D., Mosse, D. & Melhem, R. (2003). Multiple-resource periodic scheduling problem: how much fairness is necessary?, *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pp. 142 – 151.

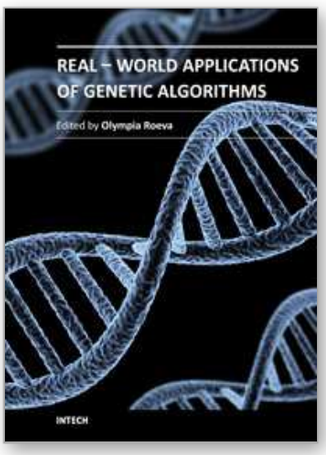
Zitzler, E., Laumanns, M. & Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm, *Technical Report 103*, Computer Engineering and Networks

Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E., Laumanns, M. & Thiele, L. (2002). Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, *in* K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papaliliou & T. Fogarty (eds), *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN2001 Conference,Athens, Greece, September 19-21, 2001*, International Center for Numerical Methos in Engineering (CIMNE), Barcelona, Spain, pp. 95–100.

IntechOpen

IntechOpen




**Real-World Applications of Genetic Algorithms**
Edited by Dr. Olympia Roeva

ISBN 978-953-51-0146-8
Hard cover, 376 pages
**Publisher** InTech
**Published online** 07, March, 2012
**Published in print edition** March, 2012

The book addresses some of the most recent issues, with the theoretical and methodological aspects, of evolutionary multi-objective optimization problems and the various design challenges using different hybrid intelligent approaches. Multi-objective optimization has been available for about two decades, and its application in real-world problems is continuously increasing. Furthermore, many applications function more effectively using a hybrid systems approach. The book presents hybrid techniques based on Artificial Neural Network, Fuzzy Sets, Automata Theory, other metaheuristic or classical algorithms, etc. The book examines various examples of algorithms in different real-world application domains as graph growing problem, speech synthesis, traveling salesman problem, scheduling problems, antenna design, genes design, modeling of chemical and biochemical processes etc.

**How to reference**
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Elias D. Niño (2012). Evolutionary Algorithms Based on the Automata Theory for the Multi-Objective Optimization of Combinatorial Problems, Real-World Applications of Genetic Algorithms, Dr. Olympia Roeva (Ed.), ISBN: 978-953-51-0146-8, InTech, Available from: http://www.intechopen.com/books/real-world-applications-of-genetic-algorithms/evolutionary-algorithms-based-on-the-automata-theory-for-the-multi-objective-optimization-of-combina

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2012 The Author(s). Licensee IntechOpen. This is an open access article distributed under the terms of the Creative Commons Attribution 3.0 License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

IntechOpen

IntechOpen