Expert Systems With Applications 42 (2015) 8956–8970

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Hybrid evolutionary algorithms for the Multiobjective Traveling Salesman Problem

Iraklis-Dimitrios Psychas, Eleni Delimpasi, Yannis Marinakis*

Technical University of Crete, School of Production Engineering and Management, University Campus, 73100 Chania, Crete, Greece

ARTICLE INFO

Keywords:
Multiobjective Differential Evolution
NSGA II
Multiobjective Traveling Salesman Problem
VNS

ABSTRACT

In the Multiobjective Traveling Salesman Problem (moTSP) simultaneous optimization of more than one objective functions is required. In this paper, three hybrid evolutionary algorithms with common characteristics are proposed and analyzed for the solution of the Multiobjective Traveling Salesman Problem. The two hybrid evolutionary algorithms are based on Differential Evolution algorithm and the third one is a hybridized version of the NSGA II. One of the challenges of the proposed algorithms is the efficient application of an algorithm, the Differential Evolution algorithm, which is suitable for continuous optimization problems, in a combinatorial optimization problem. Thus, we test two different versions of the algorithm, the one with the use of an external archive (denoted as MODE) and the other using the crowding distance (denoted as NSDE). Also, another novelty of the proposed algorithms is the use of three different mutation operators in each of the two versions of the Differential Evolution algorithm leading to six different algorithms (MODE1, MODE2 and MODE3 for the first version and NSDE1, NSDE2 and NSDE3 for the second version). We use in each algorithm a Variable Neighborhood Search (VNS) procedure in each solution separately in order to increase the exploitation abilities of the algorithms. In order to give the quality of the algorithms, experiments are conducted using classic Euclidean Traveling Salesman Problem benchmark instances taken from the TSP library. Also, we use a number of different evaluation measures in order to conclude which of the three algorithms is the most suitable for the solution of the selected problem. In general, the proposed algorithms can easily be applied in all multiobjective routing problems by changing the objective function and the constraints of the problem and they have the ability to use more than two objective functions (in the paper we test the algorithm with up to five different objective functions). The hybridized use of the global search algorithm, the Differential Evolution, with the Variable Neighborhood Search increases the exploration and the exploitation abilities of the algorithms giving very effective evolutionary multiobjective optimization algorithms.

1. Introduction

In real world applications, optimization problems with more than one objectives are very common. Thus, in these problems (Multiobjective Optimization Problems), usually, there is no single solution. The solution of Multiobjective Optimization problems with the use of evolutionary algorithms is a field that has been extensively studied during the last years. For a complete survey of the field of the evolutionary multiobjective optimization please see Abraham, Jain, and Goldberg (2005) and Coello Coello, Van Veldhuizen, and Lamont (2007).

The *Traveling Salesman Problem* (*TSP*) is the problem of finding the shortest tour through all the cities that a salesman has to visit. The TSP is probably the most famous and extensively studied problem in the field of Combinatorial Optimization (Gutin & Punnen, 2002; Lawer, Lenstra, Rinnoy Kan, & Shmoys, 1985). The TSP belongs to the class of NP-hard optimization problems (Johnson & Papadimitriou, 1985). A number of variants of the classic Traveling Salesman Problem have been proposed like the traveling salesman with time windows, multi-traveling salesman problem, the probabilistic traveling salesman problem, the prize collecting traveling salesman problem, etc. The *Multiobjective Traveling Salesman Problem (moTSP)* is the variant of the classic traveling salesman problem where simultaneous optimization of distances, costs, times, or other relevant objectives are required (Samanlioglu, Ferrell, & Kurz, 2008). In this research, the symmetric case is considered where the distance, time, or cost between cities are known and symmetric. In practical moTSP applications, there might be several competitive objective functions such as cost factors related to distance, expenses, travel time, degree of risk, energy consumption, and other relevant considerations for the tour (Samanlioglu et al., 2008).

* Corresponding author. Tel.: +30 28210 37288; fax: +30 28210 69410.
E-mail addresses: ipsychas102@gmail.com (I.-D. Psychas), eldelibasi@yahoo.gr (E. Delimpasi), marinakis@ergasya.tuc.gr (Y. Marinakis).

http://dx.doi.org/10.1016/j.eswa.2015.07.051
0957-4174/

The Multiobjective Traveling Salesman Problem is an NP-hard problem. The instances with a large number of customers cannot be solved in optimality within reasonable time. For this reason a large number of approximation techniques have been proposed for its solution. These techniques are classified into three main categories: the classical heuristics, the single solution based metaheuristics and the population based metaheuristics. More precisely, algorithms based on two phase local search (Paquete & Stutzle, 2003), two phase Pareto local search (Lust & Jaszkiewicz, 2010; Lust & Teghem, 2010a), stochastic local search (Paquete & Stutzle, 2009), tabu search (Hansen, 2000), genetic algorithms (Elaoud, 2010; Jaszkiewicz, 2002; Samanlioglu et al., 2008), evolutionary algorithms (Kumar & Singh, 2007), co-evolution strategies (Yang, Kang, & Kang, 2008), memetic algorithms (Jaszkiewicz & Zielniewicz, 2009) and ant colony optimization (Angus, 2007; Garcia-Martinez, Cordon, & Herrera, 2007) have been proposed in order to give efficient algorithms for the solution of the Multiobjective Traveling Salesman Problem. A complete survey about the Multiobjective Traveling Salesman Problem can be found in Lust and Teghem (2010b).

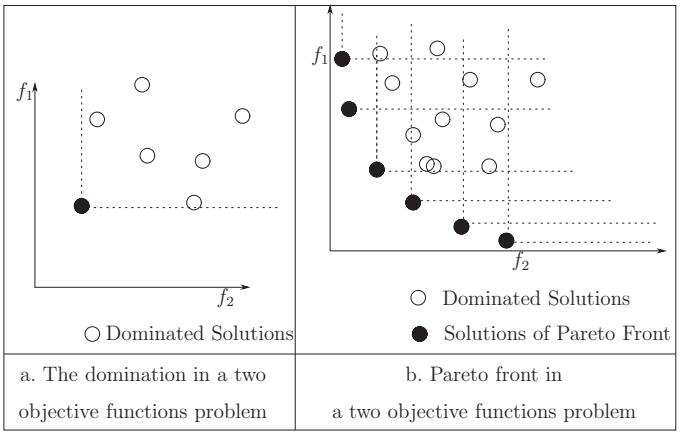
A number of publications for the solution of the multiobjective TSP and its variants have been studied in the last two years (2014–2015). The multiobjective TSP under stochastic environment was proposed by He, Wang, Pan, and Wang (2014). Changdar, Mahapatra, and Pal (2014) proposed a multiobjective genetic algorithm in order to solve a multiobjective solid TSP considering that the first objective is the cost and the second is the time. Both of them are fuzzy in nature. Li (2014) proposed to solve a 2-objectives and 3-objectives dynamic multiobjective TSP using a parallel search system. Florios and Mavrotas (2014) managed to produce the exact Pareto front for two objective functions using the AUGMECON2 method for the multiobjective TSP and for the multiobjective SCP (Set Covering Problem). Two variants of the multi-objective chemical reaction optimization (MOCRO) algorithm was used by Bouzoubia, Layeb, and Chikhi (2014) in order to produce solutions for the multiobjective TSP. This algorithm uses a non-dominated sorting procedure similar with the one used in the NSGA II algorithm. The solution of the multiobjective multiple traveling salesmen problem using membrane algorithms was proposed by He (2014). On the other hand, Bolanos, Echeverry, and Escobar (2015) solved the same problem using a NSGA II algorithm. Luo, Liu, Hao, and Liu (2014) proposed a new variant of the NSGA II algorithm (INSGA-II-MOTSP) for the solution of the multiobjective TSP. This algorithm used a layer strategy in order to avoid the production of unnecessary Pareto fronts. Another multiobjective TSP, the biobjective multiple traveling salesman problem with profits (BOMTSPP) was proposed by Labadie, Melechovsky, and Prins (2014). In this problem there are two objectives. The minimization of the total route cost and the optimization of the profits that are collected from the costumers. The problem was solved using a Path Relinking algorithm and a NSGA II algorithm. Wang, Guo, Zheng, and Wang (2015b) proposed a new variant of the multiobjective TSP, the uncertain multiobjective TSP with uncertain variables on the arcs. The solutions in this problem were given by a combination of a variant of the Artificial Bee Colony (ABC) algorithm with an uncertain approach. A new hybrid NSGA II algorithm was proposed in (Wang, Sanin, & Szczerbicki, 2015a) for the solution of the multiobjective TSP. A review of the recently proposed ant colony optimization algorithms for Multiobjective Traveling Salesman Problems using two to four objective functions is given in Ariyasingha and Fernando (2015).

In this paper, three new multiobjective evolutionary algorithms are presented and compared between them. The two of them are based on Differential Evolution algorithm with new mutation operators and the third is a hybridized version of NSGA II. The main characteristics of the proposed multiobjective algorithms are the following.

(1) Usually the Multiobjective Differential Evolution algorithms are used to solve continuous optimization problems. The main contribution of this paper is the development of two multiobjective optimization algorithm based on the Differential Evolution (DE) principles which will be applied directly in combinatorial optimization problems, like the Multiobjective Traveling Salesman Problem.
(2) Three new equations for the mutation operator for each one of the Differential Evolution versions are used. We propose these equations as we would like to answer to the main issues raised in a Multiobjective Differential Evolution algorithm, namely, which individuals will be selected for the production of the trial vector, which individuals will play the role of the parents and how the solutions of the Pareto front will affect the trial vectors. These three equations combine solutions from the Pareto front and from the population with different ways in order to find the most efficient mutation operator for the selected problem.
(3) A hybridized version of NSGA II, suitable for combinatorial optimization problems, is presented.
(4) Two different versions of the proposed algorithm, one with the use of the crowding distance (just like in the hybrid NSGA II) and one without are presented.
(5) The combination of all algorithms with a very powerful metaheuristic algorithm, the Variable Neighborhood Search (VNS) (Hansen & Mladenovic, 2001), is performed.

In our research, three variants of a Nondominated Shorting Deferential Evolution algorithm are proposed and are compared with three variants of a Multiobjective Differential Evolution algorithm and a NSGA II algorithm. In the literature, there is no other research, at least to our knowledge, that combines the Differential Evolution algorithm with the NSGA II sorting principles in order to create a Nondominated Shorting Differential Evolution algorithm for the solution of the multiobjective TSP. Furthermore, in our research we propose three new variants of the equation that is used for the calculation of the trial vector that combine individuals from the Pareto front and, also, from the current population of individuals. Also, usually the initial population for an evolutionary algorithm is produced randomly. On the other hand, considering the work of Kumar and Singh (2007) the initial population is separated in as many sub-populations as the objective functions are. Then, for each sub-population a set of optimized solutions for each objective function is used for the initial population. In the main phase of the algorithm, those solutions are evolved and the non-dominated solutions are the Pareto front's solutions of each iteration. In the proposed algorithms, an advanced variant of the Kumar and Singh method is proposed. The solutions of each sub-population are produced by using four different local search methods in order to give more exploitation and exploration abilities in the initial population. Furthermore, the evolutionary algorithms are not suitable for combinatorial optimization. In our research, we use a method in order to transform each element of the solution (path representation of the tour) into a floating point in the interval (0,1] and vice versa by using a method that is proposed by Marinakis, Iordanidou, and Marinaki (2013) in order to create evolutionary algorithms suitable for combinatorial optimization problems. Also, all the proposed algorithms are hybrid and are combined with a proposed Variable Neighborhood Search algorithm in order to improve the solutions. This method is more effective than a simple local search method and gives more exploration abilities to the solutions. In comparison to recent researches (Ariyasingha & Fernando, 2015; Florios & Mavrotas, 2014) that use the kro instances for the solution of the MOTSP for the combination of two to four objective functions, in our work, the proposed algorithms give results for the multiobjective TSP that combine from two to five objective functions.

It is very important to mention the need for the development of a very efficient algorithm for this problem. Ideally, an Expert and Intelligent System will have to replace the decision maker (the human



a. The domination in a two objective functions problem

b. Pareto front in a two objective functions problem

Fig. 1. Pareto front.

expert) when the human expert can-not solve the problem by hand or with a simpler method. The problem studied in this paper belongs in this category, as it is a combinatorial NP-hard problem and the solution of an instance with more than 10 nodes by hand is very time consuming. It is, also, very difficult to solve a Multiobjective Traveling Salesman Problem using an exact algorithm. Thus, we have to find an efficient algorithm that solves this kind of problems. There are a number of evolutionary multiobjective optimization algorithms that solve variants of the Multiobjective Traveling Salesman Problem. The novelties of the proposed algorithm compared to the other algorithms were presented previously. However, from the point of view of an Expert System we proposed a system that it has small interference with the user (the user needs only to put the parameters of the algorithm), it produces a set of equivalent solutions (the nondominated set of solutions or the Pareto front) and it finds the solutions in short computational time. The advantages of the algorithm were described earlier, however, there are, also, some limitations in the application of this kind of algorithms for combinatorial optimization problems. The main problem is that these algorithms have been designed for the solution of continuous optimization problems and not for combinatorial optimization problems. Thus, the solutions need initially to be transformed from discrete to continuous space, then, to apply the main phase of the Differential Evolution algorithm and, afterwards, to transform back the solutions from continuous to discrete space. This procedure sometimes causes the problem of losing a part of a good solution (meaning a sequence of nodes that gave a very good solution) but as the algorithm gave us the opportunity to store good and nondominated solutions in an external archive, the solutions are kept for the next generations. In general, the only problem that can cause this drawback is a slower convergence of the algorithm.

The structure of the paper is as follows. In Section 2 some evolutionary multiobjective optimization algorithms, mainly, based on Differential Evolution algorithm are described. In Section 3, an analytical description of the three hybrid evolutionary algorithms are presented while in Section 4 the computational results are presented and, finally, concluding remarks and the future research are given in Section 5.

2. Evolutionary multiobjective optimization algorithms

When optimizing two or more competitive objective functions together, the arising problem satisfies the conflicting objective functions or a compromise between them. This problem is called multiobjective optimization problem (Coello Coello et al., 2007). In Fig. 1a the domination concept in a two objective functions problem is presented while in Fig. 1b the Pareto front is given for a problem of two objective functions. For a complete review of the evolutionary multiobjective optimization algorithms please see Abraham et al. (2005), Coello Coello et al. (2007), Deb and Chichester (2001) and Reyes-Sierra and Coello Coello (2006).

The most important evolutionary multiobjective algorithms based on a genetic algorithm are the Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1984), the Multiple Objective Genetic Algorithm (MOGA) (Fonseca & Fleming, 1993), the Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas & Deb, 1994), the Non-dominated Sorting Genetic Algorithm II (NSGA II) (Deb, Agrawal, Pratab, & Meyarivan, 2000) and the Niched Pareto Genetic Algorithm (NPGA) (Horn & Nafpliotis, 1993; Horn, Nafpliotis, & Goldberg, 1994).

In recent years, a number of papers have been presented for the solution of multiobjective optimization problems using Differential Evolution. In order to apply the Differential Evolution algorithm in multiobjective optimization problems, the basic algorithm should be modified in such a way that it calculates the set of nondominated solutions. The most important issues of a Multiobjective Differential Evolution algorithm are how to select the leader of the population, how to retain the nondominated solutions in all iterations and not only between two consecutive iterations and, finally, how to maintain diversity in the population (Chakraborty, 2008; Mezura-Montes, Reyes-Sierra, & Coello Coello, 2008). In the following, the most known applications of Differential Evolution in multiobjective optimization problems are presented.

Chang, Xu, and Quek (1999) applied a Differential Evolution algorithm to fine tune the fuzzy automatic train operation for a typical mass transit system using the *DE*/*Rand*/1 version of Differential Evolution with an external archive. The most interesting issue of this algorithm concerns the selection mechanism where it is modified in order to enforce that the members of the new generation are nondominated not only regarding their objective values but also regarding a set of distance metric values (one assigned to each objective) which ensures the new solutions are at certain minimum distance from the previously found nondominated solutions.

Abbass (2002), Abbass and Sarker (2002) and Abbass, Sarker, and Newton (2001) developed the Pareto Differential Evolution (PDE). The initial population is initialized using a Gaussian distribution retaining only the nondominated solutions. Three parents are randomly selected (one as the main parent and also trial solution) and an offspring is generated from them that is placed in the population only if it dominates the main parent. This process continues until the population is completed but if the number of nondominated solutions exceeds a certain threshold, a distance metric is adopted to remove parents which are too close from each other. Abbass et al. (2001) proposed a new version of the PDE called Self-adaptive Pareto Differential Evolution (SPDE). In this algorithm, the crossover and mutation rates are self-adapted. Abbass (2001) proposed, also, an algorithm called Memetic Pareto Artificial Neural Networks (MPANN). In this algorithm, the PDE uses a Back-Propagation algorithm to speed up the convergence.

Kukkonen and Lampinen (2004) proposed a version of Differential Evolution called Generalized Differential Evolution (GDE) to solve multiobjective optimization problems. In this algorithm, the original DE is modified by introducing Pareto Dominance as a selection criterion between the old population member and the trial vector. A second version of this algorithm is the GDE2 where a crowding distance measure was used to select the best solution when the old population vector and the trial vector are feasible and nondominated with respect to each other, in such a way that the vector located in the less crowded region will be part of the population of the next generation. The algorithm ϵ-MyDE was proposed by Santana-Quintero and Coello Coello (2005). This algorithm does not allow two solutions in the Pareto front with difference less than a threshold ϵ_i in the objective function i.

Xue (2004) and Xue, Sanderson, and Graves (2003) proposed the Multiobjective Differential Evolution (MODE) where if the trial

member is dominated, then, it is replaced with one Pareto solution and if the trial vector is nondominated, then, it is inserted in the population. Iorio and Li (2004) developed the Nondominated Sorting Differential Evolution (NSDE). This algorithm is a simple modification of the NSGA-II as in NSDE the genetic operators are replaced with the operators of the Differential Evolution. Robic and Filipic (2005) proposed the Differential Evolution for Multiobjective Optimization (DEMO) which is based on the concept of Pareto Dominance. In this algorithm, when the trial vector dominates the parent vector, then, the trial vector replaces the parent, otherwise the trial vector is discarded. In order to truncate the population, DEMO applies a nondominated sorting mechanism.

Parsopoulos, Tasoulis, Pavlidis, Plagianakos, and Vrahatis (2004) proposed the Vector Evaluated Differential Evolution (VEDE) which is inspired by the Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1984). A number of M subpopulations are considered in a ring topology. Each population is evaluated using one of the objective functions of the problem, and there is an exchange of information among the populations through the migration of the best individuals. The new vector is introduced in the population if it dominates the main parent. The Pareto front is stored in an external archive.

3. Hybrid evolutionary algorithms for the multiobjective TSP

3.1. General description of the algorithms

The proposed algorithms for the solution of the Multiobjective Traveling Salesman Problem combine an evolutionary algorithm (either in the two versions, a Differential Evolution algorithm or in the third version, a Genetic algorithm (hybrid NSGA II)) with a Variable Neighborhood Search algorithm. In the following, initially, a Differential Evolution algorithm for a single objective problem is described. Afterwards, the proposed algorithms are described in detail. Initially, the one version of Differential Evolution is presented which uses an external archive (MODE1, MODE2 and MODE3 versions). The differences between MODE1, MODE2 and MODE3 versions are in the mutation equations. Afterwards, the hybrid NSGA II algorithm is presented. Finally, the other version of the Differential Evolution is presented (NSDE1, NSDE2 and NSDE3). The reason that this version is presented last is because it uses instead of the external archive, the crowding distance and the rank as they are used in the hybrid NSGA II algorithm. The differences between NSDE1, NSDE2 and NSDE3 versions are in the mutation equations. All algorithms use for the improvement of the individuals a Variable Neighborhood Search algorithm. The algorithm stops when a maximum number of iterations have been reached.

3.2. Classic Differential Evolution

Differential Evolution (DE) is a stochastic, population-based algorithm that was proposed by Storn (1996) 1999) and Storn and Price (1997). Recent books for the DE can be found in Feoktistov (2006) and Price, Storn, and Lampinen (2005). DE focuses in the distance and the direction information of the other solutions. In the Differential Evolution algorithms (Engelbrecht, 2007), initially, a mutation is applied to generate a trial vector and, afterwards, a crossover operator is used to produce one offspring. The mutation step sizes are not sampled from an a priori known probability distribution function as in other evolutionary algorithms but they are influenced by differences between individuals of the current population.

The single objective Differential Evolution works as follows: Initially, the mutation operator produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential (Engelbrecht, 2007; Feoktistov, 2006; Price et al., 2005). This trial vector will, then, be used by the crossover operator to produce offspring. For each parent, $x_i(t)$, the trial vector, $u_i(t)$, is generated as follows: a target vector, $x_{i_1}(t)$, is selected from the population, such that $i \neq i_1$. Then, two individuals, x_{i_2} and x_{i_3}, are selected randomly from the population such that $i \neq i_1 \neq i_2 \neq i_3$. Using these individuals, the trial vector is calculated by perturbing the target vector as follows:

$$u_i(t) = x_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \quad (1)$$

where $\beta \in (0, \infty)$ is the scale factor. The upper bound of β is usually the value 1 because as it has been proved if the $\beta > 1$ there is no improvement in the solutions (Engelbrecht, 2007; Price et al., 2005) and the most usually utilized value is $\beta = 0.5$.

The target vector x_{i_1} in Eq. (1) is a random member of the population. The vectors x_{i_2} and x_{i_3} are selected usually at random. Except of this classic mutation operator, there is a number of different mutation operators that have been proposed (Engelbrecht, 2007; Price et al., 2005). In this paper, a number of different new mutation operators are used suitable for the Multiobjective Traveling Salesman Problem as we would like to take advantage of all the solutions either they belong in the Pareto front or not.

After the completion of the mutation phase of the algorithm a crossover operator is, usually, applied (binomial crossover (Engelbrecht, 2007) or uniform crossover (Price et al., 2005)). In this crossover operator, the points are selected randomly from the trial vector and from the parent. Initially, a crossover operator number (Cr) is selected (Price et al., 2005) that controls the fraction of parameters that are selected from the trial vector. The Cr value is compared with the output of a random number generator, $rand_i(0, 1)$. If the random number is less or equal to the Cr, the corresponding value is inherited from the trial vector, otherwise, it is selected from the parent:

$$x'_i(t) = \begin{cases} u_i(t), & \text{if } rand_i(0,1) \leq Cr \\ x_i(t), & \text{otherwise.} \end{cases} \quad (2)$$

After the crossover operator, the fitness function of the offspring $x'_i(t)$ is calculated and if it is better than the fitness function of the parent, then, the trial vector is selected for the next generation, otherwise, the parent survives for at least one more generation (Engelbrecht, 2007).

3.3. Initial population

Usually in an evolutionary algorithm, the initial population is calculated at random but as we would like to give more exploration and exploitation abilities in the initial population, we use the following strategy. At first, the initial population is separated in as many subpopulations as the objective functions are. If P is the number of the initial solutions of the initial population, then, each sub-population must have $W = P/K$ solutions where K is the number of the objective functions. The first member of each sub-population is calculated by using the Nearest Neighborhood method (Lawer et al., 1985). The swap method (Lawer et al., 1985) is used for the calculation of 2nd to $W/3$ individuals (where W is the sub-population number), while the 2-opt method (Lin, 1965) is used in order to produce the $W/3 + 1$ to $2W/3$ individuals. Finally, the last individuals are produced at random. These strategies were selected as we would like to have a number of solutions near to the solution of the nearest neighborhood method, thus, increasing the exploitation abilities of the initial population of the algorithm around of probably good initial solutions.

3.4. Fitness function

Concerning the fitness function, it should be noted that in Mulitobjective TSP, the fitness of each individual is related to the route length of each circle and since the problem that we deal with is a minimization problem, if a feasible solution has a high objective function value, then, it is characterized as an unpromising solution candidate.

More analytically, the fitness function is calculated as follows: given a set $\{1, \cdots, N\}$ of cities and K costs $c_k(i,j)$ (with $k = 1, \ldots, K$ and K is the number of objective functions) between each pair of cities $\{i, j\}$ with $(i \neq j)$, the Multiobjective Traveling Salesman Problem consists of finding a solution, an order π of the cities, so as to minimize the following costs:

$$min\, z_k(\pi) = \sum_{i=1}^{N-1} c_k(\pi(i), \pi(i+1)) + c_k(\pi(N), \pi(1)) \quad (3)$$

These K quantities, z_k, correspond to the values taken by the various objectives for a tour realized by a traveling salesman who visits each city exactly once and, then, returns to the starting city. We are interested here only in the symmetric Euclidean Multiobjective Traveling Salesman Problem.

3.5. Path representation

One of the key issues in designing a successful Multiobjective Evolutionary algorithm for the Multiobjective Traveling Salesman Problem is to find a suitable mapping between the Traveling Salesman Problem solutions and individuals in the evolutionary algorithm. Each individual is recorded via the path representation of the tour, that is, via the specific sequence of the nodes. It is represented by a d-dimensional vector in problem space and its performance is evaluated on the predefined fitness functions (see Section 3.4).

In the problem studied in this paper, one issue that we have to deal with is the fact that, as all the solutions are represented with the path representation of the tour, they are not in a suitable form for the evolutionary algorithms. For example, if we have an individual with five nodes a possible path representation is the following:

1 3 5 2 4

As the calculation of the trial vector of each individual is performed, the above mentioned representation should be transformed appropriately. Each element of the solution is transformed into a floating point in the interval (0,1], the trial vectors of all individuals are calculated and, then, a conversion back into the integer domain is performed using relative position indexing (Lichtblau, 2002). Thus, initially, each element of the solution is divided by the vector's largest element, and for the previous example the trial vector becomes:

0.2 0.6 1 0.4 0.8

After the calculation of the trial vectors, the elements of the vectors are transformed back into the integer domain by assigning the smallest floating value to the smallest integer (0.17 to 1 in the following example), the next floating value to the next integer (0.28 to 2 in the following example) and, so on, until the largest floating value is assigned to the largest integer (0.58 to 5 in the following example). Thus, if after the calculation of the trial vector, the solution has become:

0.37 0.42 0.17 0.58 0.28

then, the backward transformation will give the following vector:

3 4 1 5 2.

3.6. MODE

The Multiobjective Differential Evolution algorithms have to find the leader of each of the individuals. A number of different ways have been proposed (Chakraborty, 2008). The following strategy is used for confronting this issue: from the initial population of individuals an external archive is created using the nondominated solutions as it happens in a number of Multiobjective Differential Evolution implementations (Chakraborty, 2008). In this archive, a number of classifications are performed based on each objective function. The feasible population is, then, evaluated by each objective function separately and the corresponding values are stored, while a global best solution arises for each one of the K objective functions. The initialization of the algorithm ends with the creation of the initial Pareto front.

The iterative procedure starts with the mutation operator which produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential. Some important issues are which individuals will be selected for the production of the trial vector, which individuals will play the role of the parents and how the solutions of the Pareto front will affect the trial vectors. In this paper, three new mutation operators are applied that differ from the classic ones for the Differential Evolution (Engelbrecht, 2007; Feoktistov, 2006; Price et al., 2005) as the solutions should be from both the solutions of the Pareto front and from the other solutions not belonging to the Pareto front in order to give to the algorithm more exploration abilities. Thus, the three different equations proposed, which are denoted as MODE1, MODE2 and MODE3, are, respectively, the following :

$$u_i(t) = Pareto_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \quad (4)$$

$$u_i(t) = x_{i_1}(t) + \beta(Pareto_{i_2}(t) - Pareto_{i_3}(t)) \quad (5)$$

$$u_i(t) = Pareto_{i_1}(t) + \beta(Pareto_{i_2}(t) - Pareto_{i_3}(t)) \quad (6)$$

where the solutions x_i are random solutions from the population while the solutions $Pareto_i$ are random solutions from the Pareto front. Thus, in the first version (MODE1), the target vector is selected randomly from the Pareto front while the other two vectors are selected from the population. In the second version (MODE2), the target vector is selected from the population and the other two vectors are selected randomly from the Pareto front. Finally, in the third version (MODE3), all vectors are selected randomly from the Pareto front.

The reason that we use these three different operators is that, as it is mentioned previously, the most important part of Differential Evolution algorithm is the suitable calculation of the leader individuals from the external archive. If we had applied the classic mutation operators of a DE algorithm, then, the external archive could not have been used in this phase. In this phase, three different versions of the external archive in combination with solutions of the population are selected in order to see which of them will give better Pareto front in our problem.

In the first version, the random solution of the Pareto front is used as the target vector and the other two solutions which produce the weighted difference will be selected randomly from the population. As it was mentioned in the description of the Differential Evolution algorithm, the target vector can be selected with a number of different ways. One of them is the selection of the best member. This is a very challenging issue in a Multiobjective Optimization problem, because as we have a number of different equivalent vectors (solutions), it is very difficult to find which one is the best or the leader. Thus, in this version, the leader is selected randomly from the Pareto front.

In the second version, we use a variant where the target solution is selected randomly from the population and not from the Pareto front. The reason is that we would like to see the performance of the algorithm if the target solution was taken from the population. The other two solutions were selected from the Pareto front as we would like to, also, take advantage of the good solutions from the Pareto front.

In the final version, only solutions from the Pareto front are used as we would like to develop a version that only the best solutions will be included in the mutation phase. As in the Pareto front there are solutions in different places of the solution space, the selection of the three solutions in this phase from the external archive will increase the exploration abilities of the algorithm. For the other two versions

(MODE1 and MODE2) the exploration ability is increased as the one or two (depending the version) solutions are in the Pareto front and the other two or one (depending the version) solutions are from the population which give to the algorithm the possibility of searching for a better solution in promising regions (where the solution of the Pareto front is) and in possible non-promising regions (where the solution of the population may be).

After the calculation of the trial vector a crossover is, usually, used. In the proposed algorithm, a crossover operator is not used. C_r is set equal to 1 as it was observed after a large number of tests using different values for C_r that the results obtained, for the specific problem studied in this paper, were the best when the value of C_r was set equal to 1. Finally, in order to improve the solutions, a Variable Neighborhood Search (VNS) algorithm (see Section 3.7) is applied in each trial vector for a certain number of iterations ($ls_{number} = 100$). Afterwards, if the solution of the trial vector in iteration t dominates the solution of the parent, then, the solution of the parent is replaced by the trial vector. On the other hand, if the solution of the parent dominates the trial vector, then, the solution of the parent remains the same. Finally, if these two solutions are not dominated between them, then, the solution of the parent is not replaced. This is performed as it is desirable to give to the individual more exploration abilities. It should be noted that the nondominated solutions are not deleted from the Pareto front and, thus, the good solutions will not disappeared from the population.

In the next iterations, in order to insert an individual in the archive, there are two possibilities. The one possibility is that the individual is nondominated with respect to the contents of the archive and the other is that it dominates any individual in the archive but in this case all the dominated individuals have to be deleted from the archive. In order to keep computing time under control, the size of the archive has to be restricted in order its size not to be increased very quickly. Thus, a parameter A is used. When the number of individuals in the archive becomes larger than A, then, a classification of the individuals based on each objective function is performed and in the archive the A/K best individuals based on each classification remain. A pseudocode of the algorithm is the following:

Initialization
Selection of the number of individuals
Generation of the initial population
Evaluation of the population for each objective function
Selection of the mutation operator
Initialization of the Pareto front
Main phase
Do while the maximum number of generations has not been reached:

Do while the maximum number of parents has not been reached:

Selection of the parent vector
Creation of the trial vector by applying the mutation operator
Evaluation of the individuals for each objective function
Application of VNS in each trial vector (offspring)
if the offspring dominates the parent **then**

Replace the parent with the offspring for the next generation

endif

enddo
Update of the Pareto front

Enddo
Return Pareto front.

3.7. Variable Neighborhood Search

The Variable Neighborhood Search (VNS) algorithm (Hansen & Mladenovic, 2001) is applied in each individual. In this research, the VNS algorithm is used with the following way. Initially, the 2-opt local search algorithm (Lawer et al., 1985) is applied for each individual for a certain number of iterations (vns_{max}). Afterwards, if 2-opt is trapped in a local optimum for a number of iterations, the 3-opt algorithm (Lawer et al., 1985) is applied and, finally, when the 3-opt is trapped in a local optimum, a swap algorithm (Lawer et al., 1985) is applied.

3.8. Hybrid NSGA II

In this paper a hybridized version of NSGA II algorithm is presented. NSGA II is an improved version of NSGA (non-dominated sorting genetic algorithm) and was originally proposed by Deb, Pratap, Agarwal, and Meyarivan (2002). In this paper, the initial population of NSGA II is calculated using the same way with the one used in MODE algorithm. In order to use the same type of MODE's solutions, we transform the solutions produced by NSGA II into a floating point in the interval (0,1] by the same way as in MODE. In each iteration and for each solution we calculate the *rank* and the *crowding distance* (Deb et al., 2002). In each iteration, firstly, we sort the solutions using the *rank* and, afterwards, using the *crowding distance*. Then, we have to select two parents. For each parent solution, we randomly select two solutions and the parent solution is the one with the best *rank*. If the *rank* is the same, the parent solution is the one with the best *crowding distance*. In the next step, a crossover procedure between two parents is performed in order to produce two offspring. The equations for the two offspring are:

$$x'_l(t) = (1-g) * x_m(t) + g * x_n(t) \tag{7}$$

$$x'_f(t) = g * x_m(t) + (1-g) * x_n(t) \tag{8}$$

where g is a random number in (0,1), m, n are the indices denoting the two parents ($m, n = 1, \cdots, P$) and l, f are the indices denoting the two offspring ($l, f = 1, \cdots, P$) . We repeat the two previous steps for $P/2$ iterations until x'_i has P solutions (offspring). In the next step, the parents (x_i) and offspring (x'_i) vectors are combined in a new one (x''_i) and, then, the members of the x''_i are sorted using the *rank* and the *crowding distance* as in the previous step. After the calculation of the rank and the crowding distance, in order to improve the solutions and to have competitive solutions with MODE algorithm, a Variable Neighborhood Search (VNS) algorithm (see Section 3.7) is applied in each solution for a specific number of iterations. From the shorted solutions, a number of individuals equal to the initial population are selected as parents for the next iteration. A pseudocode of the algorithm is the following:

Initialization
Selection of the number of individuals
Generation of the initial population
Evaluation of the population for each objective function
Selection of the mutation operator
Initialization of the Pareto front
Main phase
Do while the maximum number of generations has not been reached:

Calculate the *rank* and the *crowding distance*
For every two parents

Produce two offspring using crossover operator
Evaluation of the offspring for each objective function

endfor
Calculate the *rank* and the *crowding distance* of all parents and offspring
Application of VNS on each individual
Evaluation of the individuals for each objective function
Short parents and offspring according to *rank* and *crowding distance*

Select number of individuals equal to the initial population
Update of the Pareto front

Enddo.
Return Pareto front

3.9. NSDE

Except of the MODE, a new variant of the Differential Evolution algorithm for Multiobjective optimization problems is developed that is called NSDE and is a combination of the three previously mentioned algorithms, the MODE, the VNS and the NSGA II. The novelty of the NSDE is based on the fact that it uses the new mutation operators of MODE adapted as it will be described in the following and the fact that the crossover operator, the crowding distance and the rank are calculated as in NSGA II. Also, in order to give competitive results with the other two algorithms, the NSDE algorithm uses the VNS procedure.

Then, the procedure continues with the mutation operator which produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential. In the equations for the mutation operator instead of using a trial vector from the population and from the Pareto front we use solutions from the Pareto front and from a new set of solutions that is created from the crossover operator of the NSGA II. This is performed as we would like to give more exploration abilities to the algorithm and to use in this phase of the algorithm elements from both previously described algorithms (MODE and hybrid NSGA II). Thus, the new mutation equations are:

$$u_i(t) = x_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \tag{9}$$

$$u_i(t) = Pareto_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \tag{10}$$

$$u_i(t) = x_{i_1}(t) + \beta(Pareto_{i_2}(t) - Pareto_{i_3}(t)) \tag{11}$$

where the solutions x_i are random solutions from the population produced using the crossover operator as in the modified NSGA II while the solutions $Pareto_i$ are random solutions from the Pareto front. A pseudocode of the algorithm is the following:

Initialization
Selection of the number of individuals
Generation of the initial population
Evaluation of the population for each objective function
Selection of the mutation operator
Initialization of the Pareto front
Calculate the *rank* and the *crowding distance*
Main phase
Do while the maximum number of generations has not been reached:

For every two parents
Produce two offspring using crossover operator
endfor
Do while the maximum number of parents has not been reached:
Produce two offspring using a crossover operator
Selection of the parent vector
Creation of the trial vector by applying the mutation operator
Evaluation of the individuals for each objective function
enddo
Application of VNS on each individual
Evaluation of the individuals for each objective function
Calculate the *rank* and the *crowding distance* of all parents and offspring
Short parents and offspring according to *rank* and *crowding distance*
Select number of individuals equal to the initial population
Update of the Pareto front

Enddo
Return Pareto front.

3.10. Evaluation measures

The evaluation of a multiobjective optimization problem is a very interesting and complicated procedure as there are many different measures that have been proposed for different problems. In the selected problem, we have one more difficulty. This is the fact that as the problem is NP-hard we do not know the optimum Pareto front and, thus, it is very difficult to prove if the set of the non-dominated solutions found by the proposed algorithm belongs to the optimum Pareto front or if we have just calculated a very good and efficient Pareto front. In general, the main goals of a set of non-dominated solutions are (Sarker & Coello Coello, 2002; Zitzler, Deb, & Thiele, 2000):

(1) the minimization of the distance in relation with the optimum Pareto front (if it is known);
(2) the finding of a uniform distribution of the solutions in the Pareto font (spread and distribution);
(3) the expanding of the diagram in greater extend in all axes;
(4) the finding of as much as possible solutions of the Pareto front.

In this paper, as the optimum Pareto front is not known, four different measures are used.

- In order to evaluate how well the proposed algorithm has distributed individuals over the non-dominated region, the observations made by Zitzler et al. (2000) are used. The authors mentioned that there are three objectives that an efficient multiobjective algorithm should have. First, the distance of the resulting non-dominated set to the Pareto-optimal front should be minimized. Second, a good (in most cases uniform) distribution of the solutions found is desirable. The assessment of this criterion is based on a certain distance metric. Finally, the extent of the obtained non-dominated front should be maximized. In the proposed algorithm, we use the maximum extent in each dimension to estimate the range to which the front spreads out. This is described by the following equation (Zitzler et al., 2000):

$$M_k = \sqrt{\sum_{i=1}^{R} max\{\| p' - q' \|\}} \tag{12}$$

  where R is the number of objectives and p', q' are the values of the objective functions of two solutions that belong to the Pareto front.
- The solutions l of the Pareto front.
- The spread or distribution of solutions (Sarker & Coello Coello, 2002). For the calculation of the spread the following equations are used:

$$Spacing = \sqrt{\frac{1}{|L-1|}\sum_{i=1}^{|L|}(d_i - \bar{d})^2} \tag{13}$$

$$d_i = min\sum_{k=1}^{K}|z_k(l_i) - z_k(l_j)|, \quad l_j \in L \text{ and } l_j \neq l_i \tag{14}$$

  where K is the number of objective functions, L is the number of solutions of the front, z_k is the value of the k objective function, d_i is the minimum distance of solution i of its nearest solution and $\bar{d}$ the average value of all distances.
- Coverage (Zitzler et al., 2000): for a pair (A,B) of approximation sets the fraction of solutions in B that are weakly dominated by one or more solutions in A. The coverage measure is calculated by the following equation:

$$C(A,B) = \frac{|\{b \in B; \exists a \in A : a \leq b\}|}{|B|}. \tag{15}$$

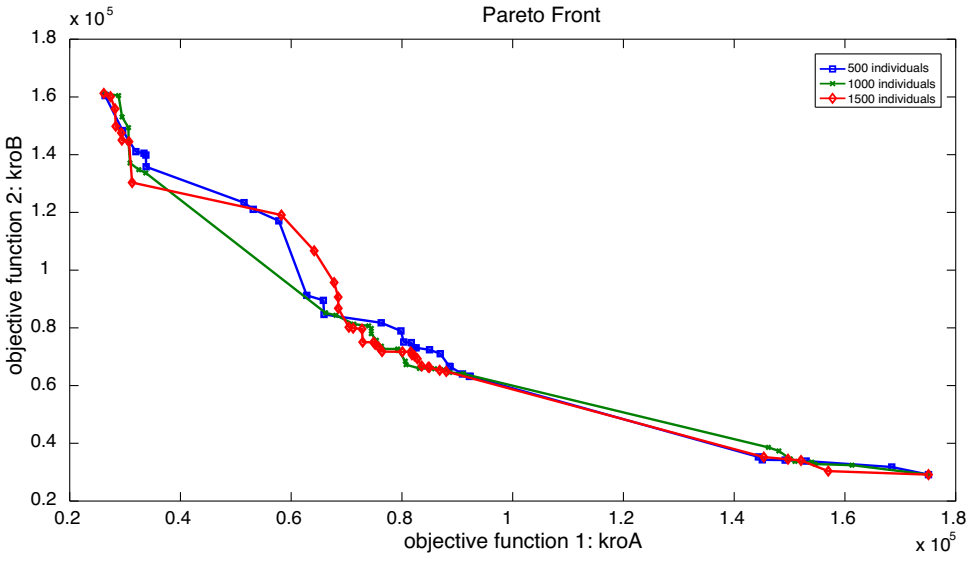


Fig. 2. Pareto front with different number of individuals.

4. Computational results

The whole algorithmic approaches was implemented in Visual C++. As it is mentioned previously in the multiobjective (K-objective) TSP, K different objective functions are defined. In practical applications, the objective functions may, for example, correspond to cost, length or travel time between each pair of towns. In order to test our algorithms we have to create a number of instances, but as the only information that we have for a Traveling Salesman Problem is the coordination of the nodes, it is very difficult to find a different objective function besides the minimization of the distance. Thus, we could say that an instance of the K-objective TSP has several graphs associated, each of them having a different cost $c_k(i, j)$ for the same edge ij. The only way to achieve this is to say that an instance with two objective functions is combined of two different single objective Traveling Salesman Problem instances having the same number of nodes. Similarly, if we would like to solve an instance with three objective functions we have to combine three different instances having the same number of nodes. The data for the computational experiments are taken from the TSPLIB and involve instances with Euclidean distances, five instances with 100 cities (kroA100, kroB100, kroC100, kroD100, and kroE100) and two instances with 200 cities (kroA200, kroB200). We created 2-, 3-, 4-, and 5-objective problems by combining these seven instances. For example, to create a 3-objective problem, we used kroA100, kroB100 and kroC100 (kroABC100) as the first, second, and third objectives, respectively. In the case where we have 200 nodes we created more instances using the kro instances with 100 nodes. For example, the instance kroAB-CD was created using the data of kroAB for the one objective function and the data of kroCD for the other.

A number of different alternative values for the parameters of the algorithm were tested and the ones selected are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. Thus, the selected parameters for MODE and NSDE are given in the following:

- number of individuals: 1000;
- number of generations: 200;
- $\beta = 0.5$;
- $Cr = 1$;

For example, in order to select the number of individuals, the algorithm was executed using 20–1500 individuals for a number of instances. We prefer to use a large number of individuals in order to increase, if possible, the number of solutions of the Pareto front. Thus, a number of individuals less than 500, although it reduces the computational time of the algorithm, it did not give a very good Pareto front as the number of non-dominated solutions did not increase with a very good rate and in the final Pareto front the solutions were too few. In Fig. 2, an example with the three alternatives that gave the most promising Pareto fronts is presented. It is an example of kroAB with three different numbers of individuals, 500, 1000 and 1500, respectively. As we can see the results using 1000 and 1500 individuals are a little better than the results of 500 individuals. The results of using 1000 and 1500 individuals are almost equivalent. However, the algorithm using 1000 individuals converges in less computational time than when it uses 1500 individuals and as in the other instances the results were similar, we selected the number of individuals equal to 1000. As we would like to have fair comparisons between MODE and NSDE versions and the hybrid NSGA II and to have the same function evaluations between the methods, the number of individuals and the number of generations were selected to be the same in hybrid NSGA II with the ones selected for MODE and NSDE versions.

After the selection of the final parameters, the three alternative versions of the Multiobjective Differential Evolution algorithm (MODE1, MODE2 and MODE3), of the Non-dominated Sorting Differential Evolution (NSDE1, NSDE2, NSDE3) and of the hybrid Non-dominated Sorting Genetic Algorithm II (NSGA II) were tested for all combinations of kro and for two to five objective functions. In the following tables, the comparisons performed based on the four evaluated measures presented previously and the Pareto front are given. More precisely, we use the number of solutions (L) in the non-dominated set, the maximum extend in each dimension (M_k), the minimization of the spread of solutions (*Spacing*) and the *Coverage*. In Tables 1–5, the results of the first three measures for the seven methods and all combinations are presented, while in Table 6, the results of the Coverage measure are, also, presented. In Figs. 3–5, three Pareto fronts are presented. Finally, in Table 7, the average computational time of each of the combination for the seven methods is presented.

In general, it is preferred to find as many as possible non-dominated solutions, the expansion of the Pareto front to be as large as possible which shows that better solutions have been found in every dimension and the spacing of solutions to be as smaller as possible which means that the non-dominated solutions are close between them. In all five tables, the spacing seems to have large

Table 1
Results for two objective functions.

	NSDE1			NSDE2			NSDE3			NSGA II		
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*
A–B	35	529.77	2853.80	30	530.08	4408.30	33	529.77	3413.10	41	525.48	3993.20
A–C	38	546.92	3236.20	43	550.08	2589.70	33	549.19	4909.70	37	545.25	3144.10
A–D	42	503.25	3686.60	31	506.05	5840.50	33	503.25	4681.70	36	495.70	3983.80
A–E	30	520.61	4025.80	24	520.61	4618.90	37	520.61	2691.80	32	525.73	4743.80
B–C	37	526.47	5289.10	34	516.81	2095.00	37	520.07	2345.70	25	523.50	6099.20
B–D	23	522.27	7397.00	27	521.32	3869.00	26	522.27	4650.20	35	517.22	4421.10
B–E	21	544.43	4672.60	36	545.78	3809.50	26	532.40	3921.30	37	540.58	5046.20
C–D	41	510.62	3828.20	36	510.62	4387.30	34	510.62	2495.20	22	506.43	6575.60
C–E	39	533.73	3235.70	43	533.73	2543.90	37	525.83	3404.20	33	527.79	3559.80
D–E	30	536.65	3350.00	31	536.65	3308.90	32	536.63	3729.10	32	537.31	4108.10
	MODE1			MODE2			MODE3					
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*			
A–B	13	530.77	3673.90	31	530.77	7136.80	17	530.77	6409.10			
A–C	23	550.52	3742.20	15	550.52	12040.00	19	550.52	3571.50			
A–D	20	506.62	7223.20	27	506.62	3596.10	28	506.62	3097.10			
A–E	20	521.37	3251.70	22	524.57	8085.50	20	524.57	3471.20			
B–C	30	533.42	2517.00	30	533.42	2674.50	31	533.42	3267.20			
B–D	22	522.27	3142.80	26	525.59	2241.70	24	525.59	10059.00			
B–E	23	544.43	3136.10	39	546.43	2088.20	38	546.43	3885.70			
C–D	17	510.62	4729.70	21	510.62	7256.60	22	510.62	5634.10			
C–E	26	537.68	3018.10	31	537.68	4079.10	33	537.68	3059.50			
D–E	20	536.65	4575.50	27	536.65	3376.80	19	536.65	10869.00			

Table 2
Results for three objective functions.

	NSDE1			NSDE2			NSDE3			NSGA II		
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*
A–B–C	81	665.85	6281.70	81	670.22	10494.00	82	669.15	5897.40	75	665.47	6361.90
A–B–D	92	661.12	6863.30	56	660.36	4782.60	62	660.26	7862.60	69	650.54	5971.60
A–B–E	77	673.22	7540.60	83	673.23	6259.00	98	671.20	5097.00	78	667.29	7727.60
A–C–D	81	655.55	5019.10	69	657.34	6510.10	76	655.55	5992.10	68	643.95	6625.70
A–C–E	75	673.20	5596.40	73	678.22	6457.40	84	678.22	5950.60	77	668.39	6756.60
A–D–E	78	664.38	6921.40	73	663.62	4618.70	81	664.47	5896.20	66	659.65	5413.80
B–C–D	89	651.47	7412.90	77	648.14	6748.30	86	649.92	5164.20	75	651.52	4574.30
B–C–E	69	673.35	9476.70	77	673.35	6130.20	84	673.65	4828.80	65	665.18	5769.40
B–D–E	78	676.51	7185.20	63	679.38	7933.70	64	679.20	12572.00	59	669.69	7031.90
C–D–E	68	667.33	7192.90	80	667.33	4625.40	70	667.33	9351.70	60	665.24	6098.50
	MODE1			MODE2			MODE3					
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*			
A–B–C	67	681.30	9353.80	69	681.30	6222.40	67	681.30	7027.60			
A–B–D	55	667.79	8467.50	59	667.79	5558.70	65	667.79	5484.10			
A–B–E	85	679.87	6117.00	80	679.87	8817.20	76	679.87	18134.00			
A–C–D	61	660.83	6879.60	56	660.83	7351.30	59	660.83	8695.90			
A–C–E	53	677.89	10234.00	51	677.89	8733.70	52	677.89	8657.60			
A–D–E	65	665.64	5445.00	60	665.64	6927.80	69	665.64	4850.90			
B–C–D	72	664.50	5104.80	79	664.50	4286.20	74	664.50	4247.50			
B–C–E	52	673.32	6313.40	57	673.32	9001.50	54	673.32	10135.00			
B–D–E	65	682.26	4814.40	66	682.26	6574.90	68	682.26	7105.20			
C–D–E	61	672.84	5856.80	61	672.84	6394.90	63	672.84	4197.10			

values; however, this is due to the fact that the values of the objective functions found in a Multiobjective Traveling Salesman Problem are in the interval $(0.2 \times 10^5, 2 \times 10^5)$. In Table 1, the results of the seven methods using two objective functions are presented. Regarding the number of Pareto solutions, NSDE1, NSDE3 and NSGA II perform better than the others in 3 instances (the instances produced by combinations of instances of single objective TSP as described earlier) each. NSDE2 performs better in 2 instances and MODE2 in 1 instance. Considering the M_k measure, MODE1, MODE2 and MODE3 prevail from all the others as they perform better in 7 out of 10 instances. Moreover, MODE2 and MODE3 perform better than all the other methods (including MODE1) in 1 out of 3 remaining instances. In the last two instances, NSGA II gives better results. Finally, for the spacing measure, NSDE2 gives better results in 4 instances, MODE2 and NSDE3 in 2 instances and MODE3 and NSDE1 in one instance, respectively.

In Tables 2–4, the results with three objective, four objective and five objective functions are presented, respectively. The algorithms were tested in 10 instances with 3 objective functions, 5 instances with 4 objective functions and 1 instance with five objective functions. The results in the three measures are analogous with the results of the instances with 2 objective functions. More precisely, concerning the number of Pareto solutions when we have 3 objective functions, the NSDE3 performs better in 5 instances, the NSDE1 in 4 and the NSDE2 in 1 while when we have 4 objective functions, the NSDE2 performs better in 2 and MODE3, NSDE3 and NSGA II perform

Table 3
Results for four objective functions.

	NSDE1			NSDE2			NSDE3			NSGA II		
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*
A–B–C–D	129	775.15	7954.00	132	779.42	7178.10	154	780.22	5967.20	114	764.52	7147.80
A–B–C–E	122	791.06	7514.00	130	791.06	7896.60	125	791.06	10391.00	132	776.31	7285.30
A–B–D–E	152	780.58	7181.60	123	784.36	9475.40	140	783.07	8255.10	123	776.92	6951.10
A–C–D–E	118	780.95	7849.10	144	780.02	7445.00	132	780.71	7846.30	125	775.36	9124.60
B–C–D–E	124	795.47	8615.70	144	795.19	10134.00	137	795.19	7396.30	104	772.84	8304.80
	MODE1			MODE2			MODE3					
kro	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*			
A–B–C–D	107	784.66	9065.00	111	784.66	8031.40	115	784.66	6856.40			
A–B–C–E	123	793.24	7140.40	117	793.24	8684.00	117	793.24	8764.60			
A–B–D–E	147	794.63	6474.40	139	794.63	7926.80	157	794.63	6722.30			
A–C–D–E	121	792.89	5869.90	123	792.89	6537.40	117	792.89	9458.00			
B–C–D–E	128	799.02	7302.60	137	799.02	7456.90	138	799.02	6533.10			

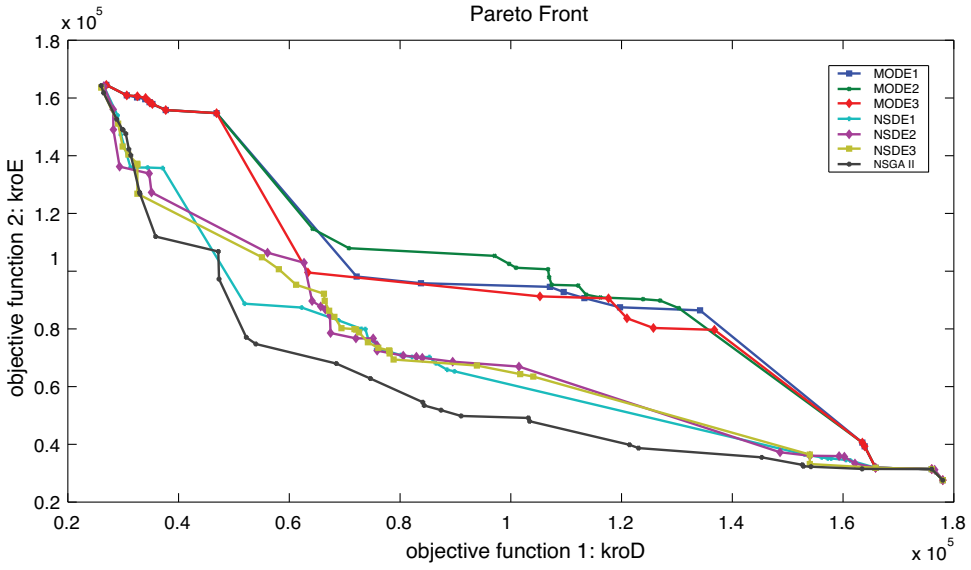


Fig. 3. Pareto front of all methods for kroDE and 100 nodes.

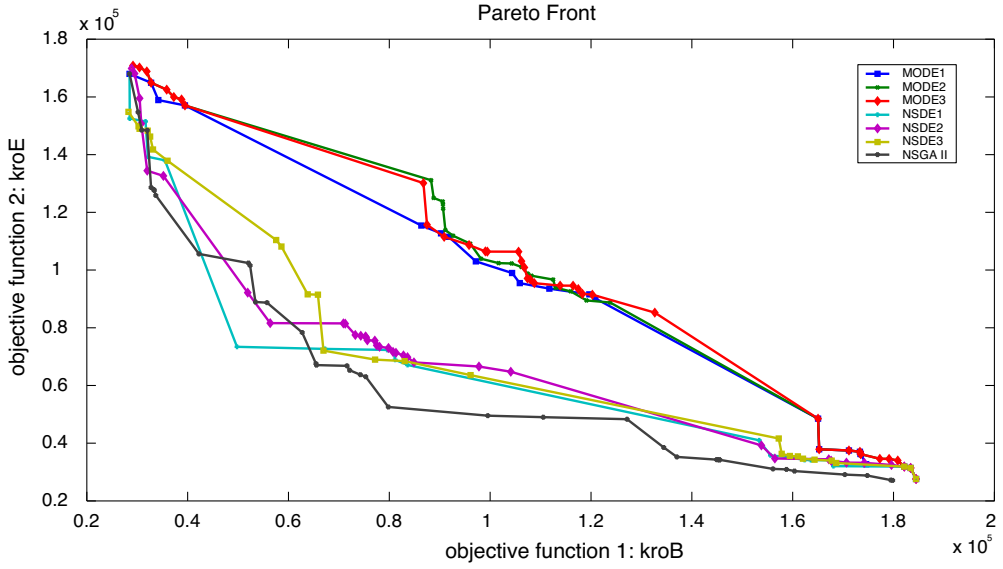


Fig. 4. Pareto front of all methods for kroBE and 100 nodes.

Table 4
Results for five objective functions.

Methods	kro A–B–C–D–E		
	L	M_k	*Spacing*
NSDE1	207	892.39	10621.00
NSDE2	220	892.09	8304.20
NSDE3	187	887.14	8558.50
MODE1	191	873.73	7995.80
MODE2	243	893.14	5923.60
MODE3	245	893.14	5995.80
NSGA II	250	893.14	6842.90

better in 1 instance, respectively. Finally, when we have five objective functions, MODE3 performs better than the other methods. In the M_k measure and using three objective functions, the three MODE versions find the same values in 8 instances, the NSDE3 performs better in 2 and NSDE2 in 1 while using four and five objective functions, MODE versions find the same values in all instances. Finally, concerning the spacing measure and using three objective functions, NSDE3 performs better in 3 instances, NSDE1, NSDE2, and MODE3 in 2 instances each one and MODE1 in one while when we have four objective functions, MODE1 prevail in 3 and NSDE3 and MODE3 in 1 each, respectively. Finally, when 5 objective functions are used, MODE1 finds a better value in spacing measure.

In Table 5, the results of large scale instances using 200 nodes are presented. In general, the same conclusions as in the previous four tables are derived. More precisely, concerning the number of non-dominated solutions the NSDE versions perform better than the other versions. The NSDE2 prevail in 8 instances and NSDE1 and NSDE3 in 4 instances, respectively. Regarding the M_k measure, MODE versions perform better or equally between them or with other versions in the most instances. Thus, MODE2 and MODE3 performs better or equally well with others in 12 instances, MODE1 in 11, NSDE2 and NSDE3 in 3 and NSDE1 in 2 instances. Finally, for the spacing measure, there is a similarity between MODE and NSDE versions in the number of instances where each of them performs better than all the other versions. More precisely, NSDE1 performs better in 6 instances, MODE2 in 5, MODE3 and NSDE2 in 2 and NSDE3 in 1. Totally, if we present the results for all instances using 2–5 objective functions simultaneously, we can see that in the number of solutions the NSDE versions perform better than the others (NSDE2 and NSDE3 in 13, NSDE1 in 11, NSGA II in 4, MODE3 in 2 and MODE2 in 1), in the M_k measure, the MODE versions perform better than the others (MODE2 and MODE3 in 34, MODE2 in 32, NSDE3 in 5, NSDE2 in 4 and NSDE1 and NSGA II in 2) and in the spacing measure, there is a balance between the instances that the MODE and NSDE versions perform better than the others (NSDE1 in 9, NSDE2 and MODE2 in 8, NSDE3 in 7, MODE3 in 6 and MODE1 in 4).

Taking into account the previous observations, someone could say that MODE versions perform better than the other versions. However, if someone observes Figs. 3–5 where the non–dominated solutions for a specific instance and for the three MODE versions, the three NSDE versions and the hybrid NSGA II are presented, he could observe that the non-dominated solutions produced from MODE versions are inferior from the non-dominated solutions produced by NSGA II and by NSDE versions. MODE versions prevail from the others only in the fact that the produced fronts by MODE versions are expanded more

Table 5
Results for two objective functions for 200 cities.

kro	NSDE1			NSDE2			NSDE3			NSGA II		
	L	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*
A–B 200	39	764.61	4588.70	44	765.02	5817.80	41	764.98	7500.00	34	762.88	10171.00
AB–CD	40	770.10	3178.20	44	765.87	4329.80	36	770.65	9509.40	26	767.27	10019.00
AB–CE	45	776.61	4599.30	48	775.58	8167.90	47	777.82	11099.00	30	773.89	10522.00
AB–DE	45	765.63	9045.50	43	772.08	3375.00	36	773.07	3233.00	26	754.48	13896.00
AC–BD	40	780.91	7406.70	41	781.18	7207.30	46	780.81	10565.00	37	778.68	7386.70
AC–BE	29	780.43	6583.30	44	782.09	6433.00	42	780.43	8394.00	29	771.94	11312.00
AC–DE	36	778.18	9781.50	48	778.06	7378.10	40	778.18	7837.90	30	775.75	5915.50
AD–BC	35	769.49	8560.90	44	767.94	5330.30	29	760.46	9245.60	25	756.88	10435.00
AD–BE	49	772.03	8160.30	43	771.98	6998.80	44	774.25	4557.20	26	772.16	6700.40
AD–CE	38	780.64	9477.30	34	780.22	7826.40	39	780.34	6660.10	24	775.19	11973.00
AE–BC	47	778.70	4081.30	49	778.48	12240.00	58	779.50	4925.70	28	776.02	11642.00
AE–BD	37	772.18	10140.00	44	772.18	4787.00	39	772.17	6526.90	31	771.39	8433.00
AE–CD	54	798.80	4189.50	46	799.28	9118.30	49	798.80	10464.00	31	794.06	10313.00
BC–DE	35	790.67	4911.80	51	790.67	5124.20	45	790.67	12880.00	29	789.38	12147.00
BD–CE	54	780.43	5921.10	29	779.08	7118.10	47	780.21	5987.50	32	774.47	8967.20
BE–CD	38	772.56	6312.30	38	769.21	4872.50	49	770.96	5366.40	25	762.29	15730.00

kro	MODE1			MODE2			MODE3		
	L	M_k	*Spacing*	*L*	M_k	*Spacing*	*L*	M_k	*Spacing*
A–B 200	30	767.37	11478.00	27	767.37	4736.80	25	767.37	5196.60
AB–CD	24	771.13	4806.00	30	771.13	4089.80	31	771.13	6446.20
AB–CE	28	779.43	7462.10	33	779.43	6663.60	30	779.43	5189.20
AB–DE	25	774.47	7658.80	33	774.47	8281.50	38	774.47	6291.00
AC–BD	28	781.84	5782.70	31	781.84	5458.60	30	781.84	5279.80
AC–BE	23	781.77	5392.20	27	781.77	4770.50	26	781.77	5325.20
AC–DE	17	778.06	11118.00	28	778.06	4655.70	29	778.06	5848.10
AD–BC	19	772.05	7884.80	31	772.05	3636.00	35	772.05	8328.20
AD–BE	22	773.78	6161.10	31	774.25	2928.30	35	774.25	2655.40
AD–CE	14	781.67	12947.00	23	781.66	5291.00	27	781.66	6327.40
AE–BC	19	781.17	4399.50	30	781.17	6054.00	27	781.17	5583.60
AE–BD	17	772.18	23665.00	30	772.62	5205.20	20	772.62	6281.10
AE–CD	29	796.63	10971.00	29	796.63	10791.00	32	796.63	10540.00
BC–DE	26	790.67	11294.00	32	790.67	4325.60	30	790.67	6602.80
BD–CE	14	780.63	25466.00	26	780.63	11528.00	19	780.63	15830.00
BE–CD	18	773.34	12497.00	24	773.34	11786.00	24	773.34	7219.80

Table 6
Results using the coverage evaluation measures.

	C(A,B) with 2 objectives and 100 nodes				C(A,B) with 3 objectives and 100 nodes			
A/B	NSDE1	NSDE2	NSDE3	NSGA II	NSDE1	NSDE2	NSDE3	NSGA II
NSDE1	0.00	0.47	0.41	0.10	0.00	0.52	0.52	0.14
NSDE2	0.52	0.00	0.51	0.10	0.54	0.00	0.52	0.12
NSDE3	0.49	0.48	0.00	0.10	0.50	0.51	0.00	0.12
NSGA II	0.86	0.88	0.87	0.00	0.79	0.80	0.80	0.00
	C(A,B) with 4 objectives and 100 nodes				C(A,B) with 2 objectives and 200 nodes			
A/B	NSDE1	NSDE2	NSDE3	NSGA II	NSDE1	NSDE2	NSDE3	NSGA II
NSDE1	0.00	0.59	0.62	0.26	0.00	0.33	0.43	0.03
NSDE2	0.54	0.00	0.60	0.25	0.55	0.00	0.54	0.02
NSDE3	0.53	0.59	0.00	0.25	0.47	0.33	0.00	0.03
NSGA II	0.82	0.79	0.81	0.00	0.95	0.96	0.93	0.00
	C(A,B) with 5 objectives and 100 nodes				C(A,B) Average of all instances			
A/B	NSDE1	NSDE2	NSDE3	NSGA II	NSDE1	NSDE2	NSDE3	NSGA II
NSDE1	0.00	0.61	0.76	0.49	0.00	0.44	0.48	0.11
NSDE2	0.91	0.00	0.80	0.59	0.55	0.00	0.54	0.10
NSDE3	0.79	0.59	0.00	0.49	0.50	0.44	0.00	0.11
NSGA II	0.83	0.61	0.76	0.00	0.87	0.87	0.87	0.00

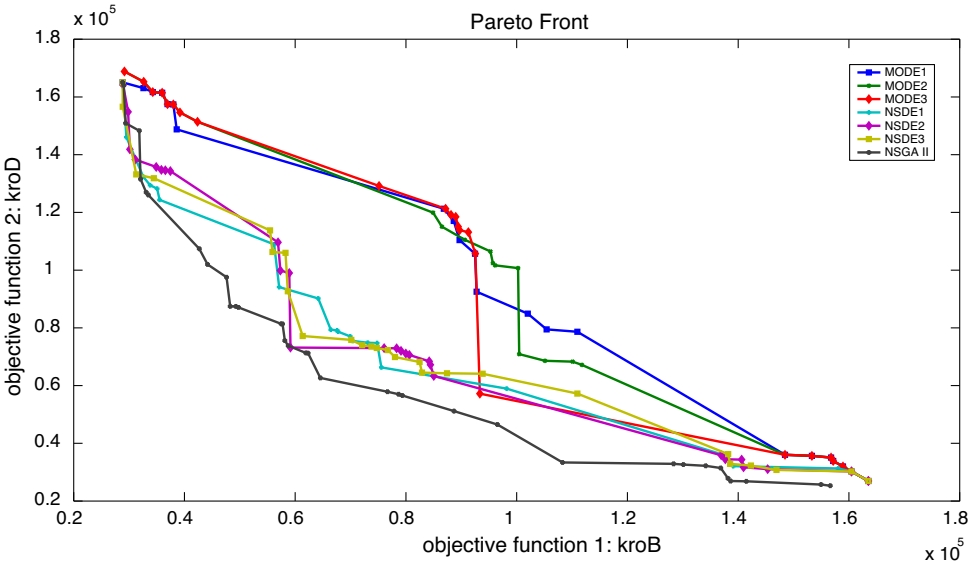


Fig. 5. Pareto front of all methods for kroBD and 100 nodes.

Table 7
Average CPU time (s).

Objective functions	NSDE1	NSDE2	NSDE3	NSGA II	MODE1	MODE2	MODE3
	Instances with 100 cities						
2	574.34	499.02	503.86	277.04	455.98	454.98	454.47
3	647.57	649.13	664.54	347.96	593.44	600.55	593.66
4	771.20	763.16	792.83	413.09	703.53	704.94	704.50
5	907.63	886.00	881.16	470.97	829.00	850.90	835.60
	Instances with 200 cities						
2	1045.16	1054.63	1041.86	624.56	999.80	1065.83	1022.91

than the fronts produced by the other methods. However, although this outcome is very important, the whole distribution of the non-dominated solutions of the seven methods and the fact that the values of the M_k are better in the MODE versions but with small differences compared to the other methods, lead us to the conclusion that the MODE versions perform worst than the other methods. For these reasons we will continue the analysis of the results without the inclusion of the MODE versions.

In Tables 1–5, if we analyze the results for all instances using 2–5 objective functions without the results of the MODE versions, we can see that NSDE versions perform better than the NSGA II. More precisely, concerning the number of Pareto solutions when we have 100 nodes and 2 objective functions, the NSGA II performs better in 4, the NSDE1 and NSDE3 in 3 and the NSDE2 in 2 instances, respectively. For 3 objective functions, the NSDE3 performs better in 5 instances, the NSDE1 in 4 and the NSDE2 in 1, respectively. For 4 objective

functions, the NSDE2 performs better in 2 and all the other methods in 1 instance. For 5 objective functions, the NSDE2 performs better. When we tested the algorithms in the large scale instances with 200 nodes, the NSDE2 performs better in 8 instances while the NSDE1 and NSDE3 perform better in 4 instances, respectively. Totally, the NSDE2 performs better in 14 instances, the NSDE3 in 13, the NSDE1 in 12 and the NSGA II in 5. Concerning the M_k measure, when we have 100 nodes and 2 objective functions, the NSDE2 performs better in 6 instances, the NSDE1 in 4 and the NSGAII and NSDE3 in 2 instances, respectively. In the instances with 3 objective functions, the NSDE2 performs better in 6 instances, the NSDE3 in 4, the NSDE1 in 2 and the NSGA II in 1. For 4 objective functions, the NSDE1 performs better in 3 and the NSDE2 and NSDE3 in 2 while for 5 objective functions, the NSDE1 performs better. Finally, when we tested the algorithms in the large scale instances with 200 nodes, the NSDE1 and NSDE3 perform better in 7 instances while the NSDE2 in 6 instances. Totally, the NSDE2 performs better in 20 instances, the NSDE1 in 17, the NSDE3 in 15 and the NSGA II in 3. Finally, concerning the spacing measure when we have 100 nodes and 2 objective functions, the NSDE2 performs better in 6 instances and the NSDE1 and NSDE3 in 2 instances, respectively. For 3 objective functions, the NSDE3 and NSDE2 perform better in 3 instances, and the NSDE1 and NSGA II in 2 instances. For 4 objective functions, the NSDE3 and the NSGA II perform better in 2 instances each one of them and the NSDE2 in 1 instance while in the instance with 5 objective functions, the NSGA II performs better. When we tested the algorithms in the large scale instances with 200 nodes, the NSDE1 performs better in 7 instances, the NSDE2 in 5, the NSDE3 in 3 and the NSGA II in one instance. Totally, the NSDE2 performs better in 15 instances, the NSDE1 in 11, the NSDE3 in 10 and the NSGA II in 6.

From these tables, we can say that the NSDE versions perform better than the NSGA II version. Also, the NSDE2 version performs better than the other two versions of NSDE. However, by observing Figs. 3–5 we can see that the Pareto front of NSGA II is always smoother and gives better solutions, especially, in the center of the Pareto front than the other three algorithms. As we would like to see which of the three versions of NSDE and of the NSGA II performs better we use another measure, the Coverage measure. In this measure, for a two different non-dominated sets (A,B) produced by two different methods, we calculate the fraction of solutions in B that are weakly dominated by one or more solutions in A. In Table 6, we present, initially, the Coverage measure for the instances with 100 nodes using 2, 3, 4 and 5 objective functions and the instances with 200 nodes using 2 objective functions and, then, in the last part of the table we present all the instances together. As we have already mentioned, the analysis is performed only for the methods NSGA II and NSDE as in general the results using MODE versions are inferior to the results produced by the other methods. Using this measure, a method is better than another method if it has larger values in the rows and smaller values in the columns. The values are always between zero and one and a value equal to 1 means that method A dominates method B in all solutions of the two sets and a value equal to 0 means that method A is completely dominated in all solutions from method B. As it can be seen NSGA II algorithm performs better in all comparisons. The NSDE2 algorithm performs slightly better than the other two algorithms (NSDE1 and NSDE3) while these two algorithms have almost the same performance.

In Table 7, the average CPU times in seconds for each run of the algorithm and for each version of the NSDE, the MODE and NSGA II using two, three, four or five objective functions are presented. As it can be seen, the fastest of the three algorithms is the NSGA II as it needs 40% less computational time than the MODE versions and 50% less computational time than the NSDE versions. The slower from the three algorithms is the NSDE algorithm. In general, we could say that with different evaluation measures, different versions of the algorithms perform better. Thus, it is very difficult for someone to conclude which method performs better than the others and which is the most suitable for the solution of the selected problem. The conflicts in the results lead us to the conclusion that we cannot say which one of the evaluation measures is the most suitable to be used for checking the effectiveness of the methods. However, the NSGA II algorithm performs better than the other algorithms in most of the evaluation measures as with the NSGA II a more efficient Pareto front was produced, in the Coverage measure the results were clearly better and the computational time was less than the one needed for the other methods. Concerning the spacing evaluation measure and the number of solutions in the Pareto front, the NSDE versions (especially the NSDE2) produced better results. Finally, in the M_k evaluation measure, MODE versions perform better than the others. It is very important to mention that the very good solutions for all three methods are, also, due to the hybridization with the very efficient metaheuristic algorithm, the Variable Neighborhood Search algorithm, which gave to the algorithms more exploration abilities.

5. Conclusions and future research

In this paper, two efficient hybridized versions of the Differential Evolution algorithm and one hybridized version of NSGA II for the solution of the Multiobjective Traveling Salesman Problem are presented. The Traveling Salesman Problem is an NP-hard problem and, thus, each variant of the Traveling Salesman Problem produces, also, an NP-hard problem. Thus, the solution of this kind of problems is very difficult to be performed using an exact algorithm and it is very time consuming to be solved by a human expert when the number of nodes is larger than 10. Thus, the need of an Expert System for the solution of the problem is inevitable. The Expert System, ideally, should substitute the human expert. However, this is very difficult to be performed in this kind of problems. There are two reasons for this, the first one is that the algorithms need the interference interaction of the human expert in order to give the parameters (population number, iterations, etc.) and the second one is that in a multiobjective optimization problem there are more than one equivalent produced solutions (the solutions that create the final Pareto front) and, thus, a human expert (the decision maker) is needed in order to select one of the equivalent solutions. We could not use an exact algorithm in order to avoid the different results with different set of parameters as it is impossible to solve large scale Multiobjective Traveling Salesman Problems with an exact algorithm. Thus, we developed two Differential Evolution based Multiobjective algorithms and we compared the results with the ones produced from the most known evolutionary multiobjective optimization algorithm, the NSGA II. In all algorithms, a number of novelties were added in order to solve some problems that were produced during the implementation of the algorithm. The main novelties were described initially in the introductory section of the paper and, then, they were analyzed in detail in the Section that the algorithm was presented analytically. The proposed algorithm is very simple to be implemented and gave very stable results in all executions (both in the same and in different benchmark instances). We used two different objective functions that represent distance cost of different sets of data. However, these objective functions could be easily be replaced in practical applications with a number of other objective functions as the minimization of the fuel consumption, the minimization of travel time, etc.

In the proposed algorithm, in order to cope with the first limitation of the algorithm (and of any evolutionary algorithm), i.e. the fact that different set of parameters may lead to different results, we performed a number of tests with different parameters finding the set of parameters that gave us the most stable results. Then, we tested all the instances with this set of parameters and the results of the algorithm were proved to be very stable based on the number of measures that were selected for the evaluation of the results. The second limitation of the proposed algorithm (and of every

evolutionary multiobjective optimization algorithm) is that the expert system produces a set of equivalent solutions, the optimal Pareto front, and, then, the human expert has to find the most suitable one for the problem. However, the finding of the best solution from the set of the Pareto front solutions was not the goal of this research. Our main target was to find an algorithm that produces a non-dominated set of solutions that is as near as possible to the optimal Pareto front. In continuous optimization problems the finding of the optimal Pareto front is easier than in combinatorial optimization problems as the one dealing with in this paper. Thus, the stability of the algorithm was one of our main targets. This is the reason that we used a different number of evaluation measures.

Our future research will be focused on the two limitations that were described previously. Our first target will be to find a way to produce automatically the parameters (or to adapt the parameters during the iterations) in order to reduce the interference of the human expert as the finding of the suitable set of parameters is a difficult procedure and sometimes needs from the human expert knowledge of the structure of the algorithm and of the problem. Our second target will be to find a way to produce from the set of the non-dominated solutions (Pareto front) the one that is the most suitable for the current human expert. However, this procedure will be a different procedure and it will be seen as a completed different research as each human expert (end user or decision maker) will give his/her own criteria in order to decide which one of the solutions is the most suitable for the problem at hand. Finally, in the point of view of combinatorial optimization problems, we will focus on the solution of problems arising in supply chain management, like Multiobjective Vehicle Routing Problem or Multiobjective Location Routing problem.

References

Abbass, H. A. (2001). A memetic pareto evolutionary approach to artificial neural networks. In *Proceedings of the Australian joint conference on artifficial intelligence* (pp. 1–12).

Abbass, H. A. (2002). The self-adaptive pareto differential evolution algorithm. In *Proceedings of IEEE congress on evolutionary computation (CEC'2002): Vol. 1* (pp. 831–836).

Abbass, H. A., & Sarker, R. (2002). The pareto differential evolution algorithm. *International Journal on Artifficial Intelligence Tools, 11*, 531–552.

Abbass, H. A., Sarker, R., & Newton, C. (2001). PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems. In *IEEE Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001): Vol. 2* (pp. 971–978).

Abraham, A., Jain, L., & Goldberg, R. (2005). *Evolutionary multiobjective optimization: theoretical advances and applications*. London: Springer-Verlag.

Angus, D. (2007). Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In *Proceedings of the 2007 IEEE symposium on computational intelligence in multicriteria decision making (MCDM 2007)* (pp. 333–340).

Ariyasingha, I. D. I. D., & Fernando, T. G. I. (2015). Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem. *Swarm and Evolutionary Computation, 23*, 11–26.

Bolanos, R. I., Echeverry, M. G., & Escobar, J. W. (2015). A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem. *Decision Science Letters, 4*, 559–568.

Bouzoubia, S., Layeb, A., & Chikhi, S. (2014). A multi-objective chemical reaction optimisation algorithm for multi-objective travelling salesman problem. *International Journal of Innovative Computing and Applications, 6*(2), 87–101.

Chakraborty, U. K. (2008). Advances in Differential Evolution. *Studies in computational intelligence*: Vol. 143. Berlin, Heidelberg: Springer-Verlag.

Chang, C. S., Xu, D. Y., & Quek, H. B. (1999). Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. In *IEE Proceedings on Electric Power Applications: Vol. 146* (pp. 577–583).

Changdar, C., Mahapatra, G. S., & Pal, R. K. (2014). An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. *Swarm and Evolutionary Computation, 15*, 27–37.

Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, & H.-P. Schwefel (Eds.), *Proceedings of the parallel problem solving from nature VI conference, LNCS 1917* (pp. 849–858).

Deb, K., & Chichester, U. K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons.

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

Elaoud, S. (2010). Multiple crossover genetic algorithm for the multiobjective traveling salesman problem. *Electronic Notes in Discrete Mathematics, 36*, 939–946.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. England: John Wiley and Sons.

Feoktistov, V. (2006). *Differential evolution – in search of solutions*. NY: Springer.

Florios, K., & Mavrotas, G. (2014). Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation, 237*, 1–19.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the fifth international conference on genetic algorithms* (pp. 416–423). Morgan Kauffman.

Garcia-Martinez, C., Cordon, O., & Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research, 180*, 116–148.

Gutin, G., & Punnen, A. (2002). *The traveling salesman problem and its variations*. Dordrecht: Kluwer Academic Publishers.

Hansen, M. P. (2000). Use of substitute scalarizing functions to guide a local search based heuristic: the case of MOTSP. *Journal of Heuristics, 6*, 419–431.

Hansen, P., & Mladenovic, N. (2001). Variable neighborhood search: principles and applications. *European Journal of Operational Research, 130*, 449–467.

He, J. (2014). Solving the multiobjective multiple traveling salesmen problem using membrane algorithm. *Bio-Inspired Computing, Theories and Applications, Communications in Computer and Information Science, 472*, 171–175.

He, Q. F., Wang, T. Z., Pan, L., & Wang, Z. T. (2014). Multiobjective traveling salesman problem under stochastic environment. *Applied Mechanics and Materials, 530*, 512–516.

Horn, J., & Nafpliotis, N. (1993). *Multiobjective optimization using the niched pareto genetic algorithm. Illigal report 91011*. Illinois Genetic Algorithms Laboratory, University of Illinois.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence: Vol. 1* (pp. 82–87).

Iorio, A. W., & Li, X. (2004). Solving rotated multi-objective optimization problems using differential evolution. In *proceedings of Australian joint conference on artificial intelligence (AI 2004): advances in artificial intelligence* (pp. 861–872).

Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research, 137*, 50–71.

Jaszkiewicz, A., & Zielniewicz, P. (2009). Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research, 193*, 885–890.

Johnson, D. S., & Papadimitriou, C. H. (1985). Computational complexity. In E. L. Lawer, J. K. Lenstra, A. H. D. Rinnoy Kan, & D. B. Shmoys (Eds.), *The traveling salesman problem: a guided tour of combinatorial optimization* (pp. 37–85). 1985.

Kukkonen, S., & Lampinen, J. (2004). An extension of generalized differential evolution for multi-objective optimization with constraints. In *Proceedings of the 13th international conference on parallel problem solving from nature – PPSN VIII, LNCS 3242* (pp. 752–761).

Kumar, R., & Singh, P. K. (2007). Pareto evolutionary algorithm hybridized with local search for biobjective TSP. *Studies in Computational Intelligence (SCI), 75*, 361–398.

Labadie, N., Melechovsky, J., & Prins, C. (2014). A parallel search system for dynamic multi-objective traveling salesman problem. *Applications of multi-criteria and game theory approaches*. London: Springer-Verlag.

Lawer, E. L., Lenstra, J. K., Rinnoy Kan, A. H. G., & Shmoys, D. B. (1985). *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley and Sons.

Li, W. (2014). A parallel search system for dynamic multi-objective traveling salesman problem. *Journal of Mathematics and System Science, 4*, 295–314.

Lichtblau, D. (2002). Discrete optimization using mathematica. In N. Callaos, T. Ebisuzaki, B. Starr, J. Abe, & D. Lichtblau (Eds.), *Proceedings of world multiconference on systemics, cybernetics and informatics (SCI 2002): Vol. 16* (pp. 169–174). International Institute of Informatics and Systemics.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Systems Technical Journal, 44*, 2245–2269.

Luo, Y., Liu, M., Hao, Z., & Liu, D. (2014). An improved NSGA-II algorithm for multi-objective traveling salesman problem. *TELKOMNIKA Indonesian Journal of Electrical Engineering, 12*(6), 4413–4418.

Lust, T., & Jaszkiewicz, A. (2010). Speed-up techniques for solving large-scale biobjective TSP. *Computers and Operations Research, 37*, 521–533.

Lust, T., & Teghem, J. (2010a). The multiobjective traveling salesman problem: a survey and a new approach. In C. C. Coello, et al. (Eds.), *Advances in multi-objective nature inspired computing, SCI: Vol. 272* (pp. 119–141).

Lust, T., & Teghem, J. (2010b). Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics, 16*, 475–510.

Marinakis, Y., Iordanidou, G. R., & Marinaki, M. (2013). Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing, 13*(4), 1693–1704.

Mezura-Montes, E., Reyes-Sierra, M., & Coello Coello, C. (2008). Multi-objective optimization using differential evolution: a survey of the state-of-the-art. *Advances in Differential Evolution, Studies in Computational Intelligence*: Vol. 143. Berlin, Heidelberg: Springer-Verlag.

Paquete, L., & Stutzle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In C. Fonseca, et al. (Eds.), *Proceedings of EMO 2003, LNCS: VOl. 2632* (pp. 479–493).

Paquete, L., & Stutzle, T. (2009). Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers and Operations Research, 36*, 2619–2631.

Parsopoulos, K. E., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., & Vrahatis, M. N. (2004). Vector evaluated differential evolution for multiobjective optimization. In *Proceedings of 2004 Congress on Evolutionary Computation (CEC 2004): Vol. 1* (pp. 204–211). Portland, Oregon, USA: IEEE Service Center.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: a practical approach to global optimization*. Berlin: Springer.

Reyes-Sierra, M., & Coello Coello, C. A. (2006). Multi-objective particle swarm optimizers: a survey of the state of the art. *International Journal of Computational Intelligence Research, 2*(3), 287–308.

Robic, T., & Filipic, B. (2005). Demo: Differential evolution for multiobjective optimization, evolutionary multi-criterion optimization. In *Proceedings of the third international conference, EMO 2005* (pp. 520–533).

Samanlioglu, F., Ferrell, W. G., & Kurz, M. E. (2008). A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Computers and Industrial Engineering, 55*, 439–449.

Santana-Quintero, L. V., & Coello Coello, C. A. (2005). An algorithm based on differential evolution for multi-objective problems. *International Journal of Computational Intelligence Research, 1*, 151–169.

Sarker, R., & Coello Coello, C. A. (2002). Assessment methodologies for multiobjective evolutionary algorithms, evolutionary optimization*: Vol. 48. International series in operations research and management science* (pp. 177–195). Springer.

Schaffer, J. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms (Ph.D. thesis)*. Vanderbilt University.

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, 2*, 221–248.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society* (pp. 519–523).

Storn, R. (1999). System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation, 3*(1), 22–34.

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Wang, P., Sanin, C., & Szczerbicki, E. (2015a). Evolutionary algorithm and decisional dna for multiple traveling salesman problem. *Neurocomputing, 150*, 50–57.

Wang, Z., Guo, J., Zheng, M., & Wang, Y. (2015b). Uncertain multiobjective traveling salesman problem. *European Journal of Operational Research, 241*, 478–489.

Xue, F. (2004). *Multi-objective differential evolution: theory and applications (Ph.D. thesis)*. New York: Rensselaer Polytechnic Institute.

Xue, F., Sanderson, A. C., & Graves, R. J. (2003). Pareto-based multiobjective differential evolution. In *IEEE proceedings of the 2003 congress on evolutionary computation (CEC'2003): Vol. 2* (pp. 862–869).

Yang, M., Kang, Z., & Kang, L. (2008). A parallel multi-algorithm solver for dynamic multi-objective TSP (DMO-TSP). In D. Huang, et al. (Eds.), *Proceedings of ICIC 2008, LNAI: Vol. 5227* (pp. 164–173).

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation, 8*(2), 173–195.