Applied Mathematical Sciences, Vol. 8, 2014, no. 62, 3075 - 3091
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2014.44297

Proposed a Hybrid Algorithm for Facility Location Problems

Danielle Durski Figueiredo

Federal University of Paraná - PPGMNE, Jardim das Américas
Cx Postal 19011, Curitiba- Paraná, 81531-980, Brazil

Luzia Vidal de Souza

Federal University of Paraná - DEGRAF, Jardim das Américas
Cx Postal 19081, Curitiba- Paraná, 81531-980, Brazil

Luiz Fernando Nunes

Federal Technological University of Paraná - DAMAT, Av. Sete de Setembro 3165, Curitiba- Paraná, 80230-901, Brazil

Abstract

The study of facility location problems is directly related to organizational problems of society, such as the location of schools, health centres, etc. In its general form, the problem of *p*-medians and maximum coverage is *NP*-Hard, and heuristic methods are used to resolve it. The Differential Evolution algorithms are powerful evolutionary optimization algorithms, originally proposed for problems in continuous spaces. Recently, it has been proposed that adjustments can be made to the mechanism of differential mutation for its application to combinational problems. This paper presents a new hybrid algorithm, using Differential Evolution Algorithms and Tabu Search, to address problems of *p*-medians and maximum coverage. For the Differential Evolution algorithm certain adaptations are presented, some unpublished, in order to solve the problems in a discrete search space. Computational tests were performed with instances from the litera-

ture, and the results suggest that the proposed technique is promising and appropriate for the resolution of the problems addressed.

Keywords: Combinatorial Optimization, Hybrid Algorithm, Facilities Location.

1 Introduction

Difficulties can be encountered when optimization problems labelled *NP*-hard are approached with exact methods, especially in cases of large dimensions, since the computational time required to obtain the global optimum value grows exponentially as the input data increases.

Metaheuristics are procedures to find a positive solution, possibly optimal, consisting of the application at each step of a subordinate heuristic, which should be modelled for each specific problem. According to Chaves [2], the main feature of metaheuristics is their ability to escape from great locations giving a certain flexibility to the restrictions of the objective function.

Among the heuristic methods are evolutionary algorithms, which were developed with inspiration from the biological process of evolution. In recent decades, new members of this family have emerged, for example, differential evolution algorithms.

The algorithms of Differential Evolution (DE) are powerful evolutionary optimization algorithms, originally proposed in the 1990s for the optimization of systems with continuous variables. Recently, adaptations have been proposed to the differential mutation mechanism for combinatorial optimization problems.

There are many proposals in the literature with the aim of improving the efficiency of metaheuristics, looking to add local search engines or make combinations with other resolution techniques. These approaches are commonly called hybridization and there is no major rule that determines how these combinations or additions are made.

The objective of this paper is to present a new hybrid algorithm, using the Differential Evolution and Tabu Search algorithms, to address problems of *p*-median and maximum coverage. Moreover, adaptations to Differential Evolution algorithms are presented, so that this may resolve the problem in a discrete search space.

This article is organized as follows: Section 2 examines the problem of locating facilities where the mathematical formulation is presented for the problems of p-median and maximum coverage; Section 3 elaborates on the classic DE algorithm in continuous domain and presents some proposals to adapt the ED for combinatorial problems found in the literature; the Tabu Search metaheuristic is discussed in Section 4, and Section 5 cites works which used hybridization method based on the ED algorithm. The hybrid algorithm proposed in this work is presented in Section 6, and finally the results of computational tests and the final conclusion are given in Section 7.

2 Facility location problems

Optimization algorithms that address problems of locating facilities, look at the problem of identifying the best locations, in a specific area, for the service facilities. Facility location models are often discussed in the literature as they allow scientific methods application to real problems.
According Minieka [14], the location problems are divided into two basic types: the centre location problems, which seek to minimize the longest distance to be travelled, and the medians location problems (*p*-median). The *p*-median problem is discussed in detail in the following section.

2.1 *P*-median problem

The *p*-median problem (PMP) is a classical combinatorial optimization question that aims to determine the best location for *p* facilities (medians) in a region, minimizing the sum of all the distances from each demand point to its nearest facility. Methods that address the PMP have been presented in the literature and may cite the study by Freitas *et al.* [6], which used the results of the spectral theory in the construction of a hybrid algorithm based on the method of Teitz & Bart. The study of PMP is highly significant as it refers directly to problems of life within society, such as determining the location of sampling sites and data transmission in the Brazilian electoral process [4], station installations for public health care of dengue epidemics in the city of Salvador - Brazil [12].
According to Christofides [5], the *p*-median problem can be formulated as a binary integer linear programming as follows:

$$Z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, x_{ij} \tag{1}$$

$$\text{Subject to: } \sum_{i=1}^{n} x_{ij} = 1 \qquad j \in N \tag{2}$$

$$\sum_{i=1}^{n} x_{ii} = p \tag{3}$$

$$x_{ij} \leq x_{ii} \qquad i, j \in N \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad i, j \in N \tag{5}$$

Where: $[d_{ij}]_{n \times n}$ is a symmetric matrix of weighted distances; $[x_{ij}]_{n \times n}$ is the allocation matrix, with $x_{ij} = 1$ if vertex *i* is allocated to vertex *j*, and $x_{ij} = 0$ if not; $x_{ii} = 1$ if vertex i is a median, and $x_{ii} = 0$ if not; p is the number of medians (facilities) to be located, and n is the number of vertices in the network, and $N = \{1, ..., n\}$.
The objective function (Eq.1) minimizes the sum of distances considered d_{ij}, where d_{ij} is the result of multiplying the distance between vertices v_i and

$v_j\,(d(v_i,v_j))$ by weight w_j, the weight w_j being the condition of each vertex v_j. Thus $d_{ij} = w_j d(v_i,v_j)$.

Constraints (Eq. 2) and (Eq. 4) dictate that each vertex *j* is allocated to only one vertex *i*, which should be a median. The constraint (Eq. 3) determines the number of *p* medians to be located, and the constraint (Eq. 5) corresponds to the conditions of completeness.

2.2 Maximum Coverage Problem

The Maximum Coverage problem (MCP) is a variant of the *p*-median, whose goal is to find a limited number of facilities to cover the maximum number of demand points, but not necessarily all. This dilemma typically occurs in locating radars and telecommunications facilities, such as antennas, transmission towers etc. A variant of the maximum coverage problem, known as the Coverage of Sets Problem, is the case where it is desired to meet all the demand and minimize the number of necessary facilities. It should be emphasized that meeting the demand means that the maximum departure *R* is kept between demand and supply. The MCP was introduced by Church and Revelle [3], and the mathematical formulation of the PMC is:

$$Z = \max \sum_{i \in I} a_i y_i \tag{6}$$

Subject to:

$$\sum_{j \in N_i} x_j \geq y_i \,\forall\, i \in I \tag{7}$$

$$\sum_{j \in J} x_j = p \tag{8}$$

$$x_j \in \{0, 1\},\ \forall\, j \in J \tag{9}$$

$$y_i \in \{0, 1\},\ \forall\, i \in I \tag{10}$$

Where: $N_i = \{j \in J / d_{ij} \leq R\}$, $\forall\ i \in I$; $x_j = 1$, if the vertex $j \in J$ is selected to become a facility, and $x_j = 0$ if not; $y_i = 1$ if the vertex $i \in I$ is served by a facility, and $y_i = 0$ if not; a_i indicates the demand on vertex $i \in I$; p is the number of facilities to be activated.

The objective function (Eq. 6) maximizes the demand attended to; constraint (Eq. 7) states that a customer will be attended if there is at least one facility located within the coverage distance. The constraint (Eq. 8) limits the number of located facilities to exactly p, and restrictions (Eq. 9) and (Eq. 10) define the decision variables of the binary type.

The following are the algorithms of Differential Evolution and Tabu Search, used in this research.

3 Differential Evolution Algorithm

Price and Storn [16] developed the Differential Evolution (DE) algorithm with the intention of solving the problem of Chebychev polynomial fit.
Initially, a population group $G = \{V_1, V_2, \cdots, V_n\}$ of individuals (vectors) is randomly generated and should cover the entire search space. In the absence of any knowledge of the search space, a uniform distribution is used for the initial population. In an operation defined as mutation, the DE algorithm generates new parameter vectors, called donor vectors, by adding the weighted difference between two vectors to a third individual, as in the following operation:

$$V_d = V_i + F(V_j - V_k) \text{ ou } V_d = V_{best} + F(V_j - V_k) \quad (11)$$

Where V_d is the donor vector, F is a weight scale applied to the vector difference, V_{best} is the population vector with the best fitness, and V_i , V_j and V_k represent random and mutually distinct individuals, chosen from the population.
After the transfer is made, another vector is chosen randomly and is called the target vector (V_{target}), whose components are mixed with the components of the donor vector, resulting in the vector called experimental. This process is known as crossover. In the basic version of the DE algorithm, one binomial crossover with crossover probability $PC \in [0,1]$ is presented according to the following rule:

$$e_i = \begin{cases} a_i, & se\ r_i \leq PC \\ d_i, & se\ r_i > PC \end{cases} \qquad i = 1, 2, \ldots, n \quad (12)$$

With r_i representing random numbers within the range of [0,1], e_i, d_i and a_i are the respective vector components: experimental, donor and target.
The selection between target and trial vectors is performed in a greedy manner, that is, if the value of the objective function applied to the trial vector is higher than that applied to the target vector, the trial vector replaces the target vector in the next generation, otherwise the target vector is maintained in the next generation.
The mutation operation applied in the DE is simple, however the way it is defined makes it impractical for problems of discrete optimization. Below are some papers that propose changes in mutation operation of the DE in order to insert the operating algorithm in the discrete domain.

3.1 Differential Evolution for discrete optimization

Note that each new discrete combinatorial optimization problem to be solved using the algorithm of Discrete Differential Evolution (DDE), generates new proposals for improvements to the DE method. The proposals generally refer to adaptations in genetic operators, particularly in the mutation operator, in order to generate feasible solutions and greater speed of convergence for the optimal solution. Below are some papers that propose changes in mutation operation for DE in order to insert the operating algorithm in the discrete domain.

A metaheuristic approach for the Differential Evolution is proposed by Prado *et al*. [15] for discrete optimization, defining the difference between two candidate solutions as a list of moves in the search space, and thus preserving the search engine in discrete domains. The method was applied to the traveling salesman problem and the problem of *N*-Queens. Tasgetiren *et al*. [19] also presented an adaptation of the DDE algorithm for the traveling salesman problem.
In this article, we propose a new way of applying the mutation operator in the Discrete Differential Evolution metaheuristic, which provides an approach to the problems within the discrete space. This metaheuristic is then combined with Tabu Search algorithm as presented below**.**

4 Tabu Search

A Tabu Search (TS) was originally developed by Glover [7] as a proposed solution to integer programming problems. Since then, the author published a series of papers containing several applications of the concept. Experience has shown the effectiveness of TS in solving various problems of different nature [8] and it can currently be affirmed that this is a highly consolidated technique.
In general terms, the TS algorithm is an adaptive procedure of a local search, endowed with a memory structure, which accepts worsening moves (when there is no scope for improvement) to escape from optimal locations [10, 17]. Being a local search procedure is based on the notion of neighbourhood. At each iteration, the current solution s changes to another which is its neighbour in the search space, that is, to a solution s' which differs from s by a modification.
Starting from an initial solution s_0 TS algorithm explores, at each iteration, a subset S of vertices neighbouring the current solution s. The member s' of S with best value in this region according to the fitness function f becomes the new current solution, even if s' is worse than s, that is, $f(s') > f(s)$ for a minimization problem.
Memory usage is an essential feature of Tabu Search. While most search processes essentially save the value of $f(s^*)$ of the best solution s^* obtained so far, the TS archives information in a kind of itinerary of the last visited solutions. Such information is used to drive the movement of one solution into another to be selected in the search space. The function of memory is to restrict the choice of certain vertex, prohibiting movements to some neighbouring solutions [9].
The prohibition of these movements is intended to prevent the return to a previously visited solution. The non-veto of certain movements may cause the algorithm to cycle. An artifice created in order not to "authorize" the occurrence of these movements is the *Tabu_List*, consisting of a list containing the solutions visited during the last $|K|$ iterations sequenced as FIFO (First In First Out).
The main control parameters of the method are the size of *Tabu_List*, the cardinality of the subset S of neighbour solutions tested at each iteration and the maximum number of iterations without improvement in the objective function.

The Tabu Search algorithm was used after obtaining an initial solution generated by a constructive procedure, by Subramanian *et al.* [18], to a problem of allocation of classes to rooms in an educational institution which was very efficient, generating high quality solutions when compared with manual solution.

5 Hybridization

With the development of generic optimizers for combinatorial problems, there is an increasing trend of using hybrid methods in the search for solutions to optimization problems as reported in Jourdan *et al.* [11]. Hybrid methods use a combination of different methods in order to enhance the best features of each and to obtain better quality solutions to the problems addressed. Regarding jobs involving hybridization with evolutionary algorithms, Almeida *et al.* [1] can be referenced, who proposed a hybrid algorithm using the Cultural Algorithm and Genetic Algorithm, to support better alternatives in the development of an oil field. Below follows a proposal for a hybrid algorithm, based on metaheuristics of Differential Evolution and Tabu Search, which addresses some combinatorial optimization problems.

6 Hybrid Algorithm Proposed

Adjustments to the DE algorithm presented in the literature for combinatorial optimization basically do not address problems of facility location. A new generic version and the mutation operator of the DE algorithm for solving combinatorial problems is proposed. This DE algorithm, adjusted and combined with the Tabu Search algorithm, is used to thin the search space contributing to a more rapid convergence and accurate, together forming the hybrid algorithm proposed, called this work DEDHp. Next, in Table 1 (below), we present the components and variables defined in the proposed algorithm.

Below are the steps of DEDHp algorithm:

Step1: Assign values for $G_{máx}$, N_p, N, PC, TB and K parameters;
Step2: Having generated the initial population, define the V_{target} and choose the V_{best};
Step3: The following in no particular order:

- Two vectors V_1 e V_2 of the population, so that V_1, V_2, V_{target} and V_{best} are distinct from one another;
- N medians of V_{best}, which take part in the mutation operation, forming a non-empty subset A;
- A component of the vector V_1 and of vector V_2 whose Cartesian coordinates are respectively ($X_{n1,}$ Y_{n1}) and (X_{m2}, Y_{m2}).

Step4: obtain the Cartesian coordinates (X,Y) belonging to a point P of the continuous space as follows:

$$X = F((1-\lambda)X_{n1} + \lambda X_{m2}) \quad (13)$$

$$Y = F((1-\lambda)Y_{n1} + \lambda Y_{m2}) \quad (14)$$

Step5: Determine the vertex P_0 of the dilemma closest to the point P so that:

- The number of times P_0 has been chosen, in previous iterations for the mutation, is less than or equal to TB;
- The movement between P_0 and the selected median a_i of subset $A \notin Tabu\_List$, where $i \in \{1, 2, ..., N\}$;
- $P_0 \notin V_{best}$;

Step6: Substitute median a_i, chosen from the subset A, by vertex P_0. END.

Table 1 - Description of variables and components DEDHp

Variables	Description
$G_{máx}$	Maximum number of algorithm iterations.
N_p	Number of population vectors $G = \{V_1, V_2, ..., V_{Np}\}$.
N_{med}	Number of medians to be determined.
N_v	Total number of vertices.
(X_{ij}, Y_{ij})	Cartesian coordinate of the i-th median j-th vector of the population $G = \{V_1, V_2, ..., V_{Np}\}$.
λ e PC	Real values between 0 e 1.
F	Real values between 0,5 e 1,5.
V_{best}	Vector that is better fitted to the current iteration.
N	Positive integer that indicates the number of medians V_{best} that will undergo the mutation.
V_{target}	Target vector referring to the population $G = \{V_1, V_2, ..., V_{Np}\}$.
V_d	Donor vector generated in the operation of the algorithm mutation.
TB	The integer value representing the maximum number of times the vertex will participate in the motions of mutation.
Tabu_List	List storing the last K movements performed between two vertices.
K	Represents the size of *Tabu_List.*

The donor vector (V_d) is obtained after the medians N, belonging to the subset A, undergo a mutation.

It is emphasized that after the movement between the median a_i and the vertex P_0 takes place, as described in Step6, it is inserted in *Tabu_List* and will leave it after $|K|$ iterations.

In the mutation operator, operations between vectors coded as integers, do not usually generate viable solutions to be multiplied, for example, a vector by a scalar F or λ between 0 and 1. The proposal to use the Cartesian coordinates of the vertices of this operator aims to generate new vertices in the continuous space,

which are located in the search space. Step5 of the proposed algorithm returns to the discrete space by identifying the vertex of the closest problem to that generated in continuous space, thus presenting viable solutions.
Figure 1 shows a two-dimensional example illustrating the vectors involved in generating a donor vector component (V_{donor}).

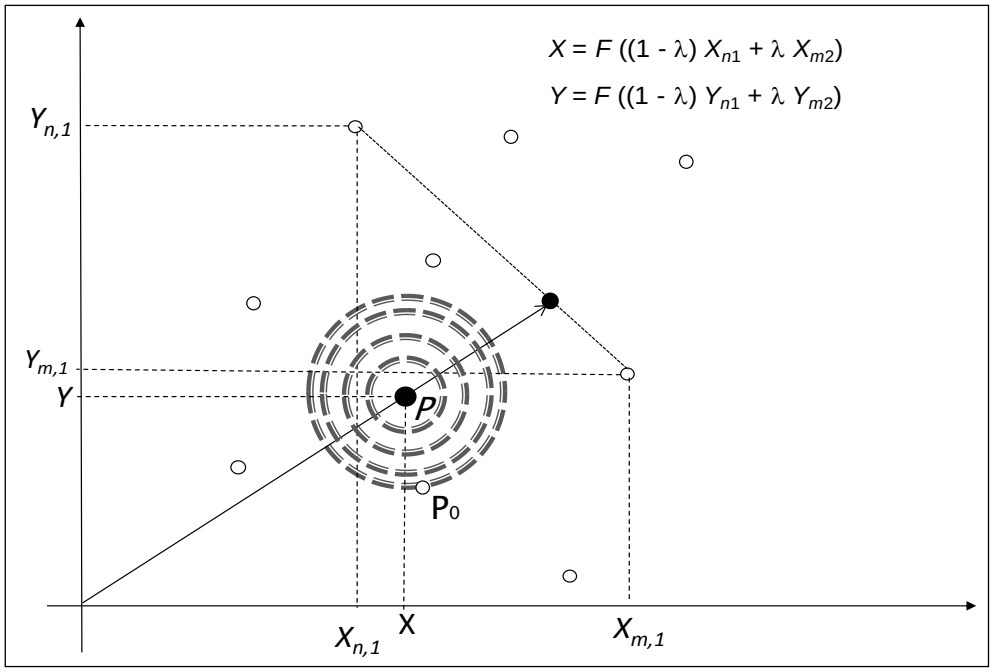

Figure 1 - The process of generating a component of the V_{donor}

After the mutation operation the binomial crossing is concluded, as previously described (see Section 3), between the V_{donor} and V_{target}, with a probability of intersection $PC \in [0,1]$, resulting in the experimental vector (V_{exp}), and applies to the greedy selection between V_{exp} and V_{target}, as described in the continuous Differential Evolution algorithm, obtaining a new population.
Below are the results obtained using the algorithm DEDHp with the adaptations displayed for some combinatorial optimization problems. These results were compared with instances in the literature and those randomly generated.

7 Results and Further Considerations

The algorithms discussed have been programmed in Visual Basic, version 2010, and a Sony computer with Intel Core i5 processor with 640GB HD, 6GB RAM and operating system Windows 7 was used for computational analysis.
The goal was to evaluate the performance of the algorithm presented in this article, in relation to the quality of solutions and computational time required to

obtain the optimal solution algorithm. We carried out ten (10) simulations for each test problem, using the maximum number of iterations as an end criterion.
Two groups of cases were used for the tests conducted in *p*-median problems. The first group presents twelve (12) cases, containing 324 and 818 vertices, whose results were compared with those obtained by Lorena *et al.* [13], referred to here as Lorena and available at [20], where the author integrates some GIS (Geographical Information Systems) with the implementation of a recent approach to heuristic Lagrangean/surrogate that has proven effective in several classes of Combinatorial Optimization problems.
Given the difficulty of finding instances in the literature that provide the Cartesian coordinates of the vertices, the second group, containing thirty (30) instances and available at [21], was randomly generated with the minimum of 100 and maximum of 600 vertices. We compared the results of this group with the optimal solutions obtained by Lingo 13.0 software.
For tests conducted on problems of maximum coverage, we used 22 (twenty-two) instances, available at [20], containing 324, 500 and 708 vertices, whose results of the proposed hybrid algorithm were also compared with the optimal solutions obtained by Lingo 13.0 software.

The caption of the symbols used in Tables 2, 3 and 4 is described below:

- *n_p*: number of vertices of the problem_number of medians (facilities);
- $S_{Optimal}$: optimal solution obtained through Lingo 13.0;
- Coverage: percentage of demand attended by p facilities;
- S_{Lorena}: best solution presented by Lorena;
- S_{Best} and S_{Worst}: respectively best and worst solutions, located by the proposed algorithm, within the 10 simulations carried out in each instance;
- *Dp*: percentage deviation addressed by the presented methods which were determined as follows: $D_p = 100 \cdot (S_{Reference} - S_{Optimal}) / S_{Optimal}$. Where $S_{Reference}$ equals to S_{Lorena} for the methods presented by Lorena, and the S_{Best} for the method proposed in this paper.
- Time: computational time in seconds of the methods applied.

Table 2 (below) shows the results obtained by the cases tested for the p-median problem, the first group.

Analyzing the information in Table 2, it was found that the DEDHp algorithm presented approximately 83.33% of the instances better or equal to the solutions presented by Lorena. The algorithm of the computation time increased as the input data increased. In 50% of cases, lower computational time was obtained than those in the literature discussed. The results between methods according to the number of vertices are shown below.

Table 2: Results of computational tests *P*-Medians regarding Lorena cases

n_p	$S_{Optimal}$	Lorena			DEDHp			
		S_{Lorena}	*Dp*	Time	S_{Best}	S_{Worst}	*Dp*	Time
324_5	122518	122518.0	0	4.7	122518	122518	0	0.8
324_10	79256.3	79256.3	0	7.3	79256.3	79256.3	0	1.5
324_20	54505.3	54533.1	0.1	7.3	54505.3	55326.5	0	5.4
324_50	32101.5	32101.5	0	7.7	32399.3	32624.6	0.9	19.2
324_108	18721.9	19683.6	5.1	7.9	18892.3	18997.9	0.9	40.0
818_5	605856	605856	0	102.6	605856	609092.2	0	9.2
818_10	384342	385371.4	0.3	97.5	384342	387148.7	0	12.7
818_20	251713	251713	0	60.4	251713	253330.1	0	29.4
818_50	146340	149251.1	2.0	43.7	147204.8	148297.8	0.6	69.3
818_100	97796.3	98992.3	1.2	57.9	99050.2	101477.5	1.3	165.0
818_150	75501.8	77440.6	2.6	66.2	76918.9	78047.7	1.9	258.1
818_272	47492.3	50086.6	5.5	85.6	49120	49442.8	3.4	433.6

In five (5) cases where n = 324, the DEDHp algorithm got 80% better or equal solutions compared with those presented by Lorena, except in the problem where p = 50. Of these cases, the computational times were better 60% of the time, indicating the greatest difference of 32 seconds improvement as compared to method presented in the literature, in the problem where p = 108, but for this problem the algorithm proposed in this article showed the biggest advantage of improved solution, which was 4.2 %.

In seven (7) instances, where n = 818, the DEDHp algorithm obtained approximately 85.71% better or equal solutions than those presented by Lorena, with the exception of the problem where p = 100. The computational times were better in 42.86 % of these cases. The major difference between the computational time was 348 seconds for the method of Lorena, the problem where p = 272, but this DEDHp algorithm showed the highest gain in improving the solution was 2.1%.

Table 3 (below) shows the results obtained for the cases tested for the p-median problem, or the second group.

Analysing the information in Table 3, it was found that the DEDHp algorithm presented optimal solutions in approximately 56.67% of cases. In 100% of cases, the computational time obtained was smaller than that obtained by Lingo 13.0 software. The instance where n = 600 and p = 10, showed the greatest difference between the computational times for the DEDHp algorithm, which was 1351 seconds.

As for the 10 (ten) cases where n = {100, 200}, the DEDHp algorithm obtained 90% of optimal solutions, presenting a deviation of 0.29% only in the case where n = 200 and p = 40. Regarding the computational time, the longest recorded in this group was where n = 200 and p = 67, equivalent to approximately 39.13% of the time as given by the software Lingo 13.0 to achieve the optimal solution.

Table 3: Results of computational tests *p*-medians for the instances generated

n_p	Lingo		DEDHp			
	$S_{Optimal}$	Time	S_{Best}	S_{Worst}	*Dp*	Time
100_5	351.46	18	351.46	351.46	0	< 1
100_10	219.23	15	219.23	219.23	0	< 1
100_20	130.97	16	130.97	131.46	0	< 1
100_33	86.88	15	86.88	87.70	0	2
200_5	3315.98	47	3315.98	3320.355	0	< 1
200_10	2173.28	39	2173.28	2173.28	0	2
200_20	1361.58	42	1361.58	1364.30	0	4
200_30	1025.12	44	1025.12	1029.60	0	7
200_40	836.3	42	838.73	843.92	0.29	11
200_67	524.27	46	524.27	525.92	0	18
300_5	15205.3	260	15205.3	15209.08	0	2
300_10	10404	256	10404	10455.03	0	4
300_30	5265.75	272	5276.21	5332.76	0.20	13
300_60	3095.3	265	3144.86	3171.56	1.60	24
300_100	1980.49	259	2003.99	2022.69	1.19	34
400_5	67035.20	455	67035.20	67035.20	0	3
400_10	46454	481	46454	46557.82	0	4
400_40	20009.9	519	20146.00	20503.37	0.68	17
400_80	12003.5	521	12162.82	12300.97	1.33	52
400_133	7679.36	511	7771.03	7840.12	1.19	88
500_5	85382.42	901	85382.42	85785.38	0	3
500_10	58590.9	848	58590.9	58904.86	0	6
500_50	22136.2	856	22379.18	22575.93	1.10	32
500_100	13301.6	886	13598.47	13808.22	2.23	64
500_167	8425.24	877	8612.28	8734.35	2.22	148
600_5	102849	1336	102849	102879.9	0	5
600_10	71249.4	1362	71249.4	71627.84	0	11
600_60	24210.7	1311	24402.6	24497.19	0.79	77
600_120	14364.2	1345	14647.88	14850.08	1.97	168
600_200	9074.87	1340	9187.14	9321.63	1.24	356

For 10 (ten) cases where n = {300, 400}, the DEDHp algorithm obtained 40% optimal solutions, presenting a deviation of approximately 0.62%, and the biggest deviation is the case where n = 300 and p = 60 equivalent to 1.60%. Regarding the computational time, the longest time registered in this group was in the case where n = 200 and p = 67 equivalent to approximately 39.13% of the time as given by the software Lingo 13.0 to achieve the optimal solution.

For 10 (ten) cases where n = {500, 600} the DEDHp algorithm obtained 40% optimal solutions, presenting a deviation of approximately 0.96%, and the biggest deviation is the case where n = 500 and p = 100 equivalent to 2.23%. Regarding

the computational time, the longest time registered in this group was in the case where n = 600 e p = 200 equivalent to approximately 26.57% of the time as given by the software Lingo 13.0 to achieve the optimal solution.
In Table 4 below, the results are resented which were obtained by DEDHp, for maximum coverage problems, as compared to the optimal solutions recorded.

Table 4: Results of Computational tests for Maximum Coverage

n_p	Lingo			DEDHp			
	$S_{Optimal}$	Coverage	Time	S_{Best}	S_{Worst}	*Dp*	Time
324_20	7302	60.09	247	7302	7213	0	4
324_30	9127	75.11	230	9103	8954	0.26	8
324_40	10443	85.94	256	10382	10252	0.58	10
324_50	11397	93.79	244	11293	10912	0.91	15
324_60	11991	98.68	246	11868	11635	1.03	17
324_80	12152	100.00	244	12150	12132	0.02	24
500_40	13340	67.69	745	13220	12936	0.90	23
500_50	14773	74.96	725	14647	14452	0.85	31
500_60	15919	80.78	724	15802	15605	0.73	38
500_70	16908	85.80	750	16710	16411	1.17	43
500_80	17749	90.06	751	17470	17302	1.57	66
500_100	18912	95.97	741	18683	18465	1.21	76
500_130	19664	99.78	734	19510	19361	0.78	92
500_167	19707	100.00	734	19706	19700	0.01	117
708_70	19481	80.53	1670	19151	18798	1.69	93
708_80	20533	84.88	1744	20180	19854	1.72	109
708_90	21449	88.66	1643	21078	20726	1.73	129
708_100	22173	91.65	1680	21849	21568	1.46	151
708_120	23200	95.90	1661	22805	22741	1.70	191
708_140	23861	98.63	1604	23521	23318	1.42	237
708_180	24185	99.97	1766	24110	24068	0.31	309
708_236	24192	100.00	1692	24192	24191	0	403

According to the information in Table 4, it was found that the DEDHp algorithm presented optimal solutions in two (2) instances. In 100% of the tests, the computational times obtained were smaller than those obtained by Lingo 13.0 software. The largest difference occurred in the case where n = 700 and p = 80, equal to 1534 seconds. The average percentage deviations presented by the proposed algorithm was approximately 0.91.
Below are presented the results of the method proposed according to the number of vertices. For the six (6) cases where n = 324, the DEDHp algorithm had an average percentage deviation approximately equal to 0.47. The largest deviation equal to 1.03% was recorded in the instance where p = 60, and for p = 20 the algorithm has reached the optimal solution. The average computing time was 13 seconds, which equals to approximately 5.34% of the average computational times presented by the software Lingo 13.0.

In eight (8) cases where n = 500, the DEDHp algorithm had an average percentage deviation approximately equal to 0.90. The largest deviation equal to 1.57% was recorded in the instance where p = 80, and for p = 167 the algorithm had the lowest percentage deviation approximately equal to 0.01. The average computing time was 61 seconds, which equals to approximately 8.23% of the average computational times presented by Lingo 13.0 software for the optimal solution.
In eight (8) cases where n = 708, the DEDHp algorithm had an average percentage deviation approximately equal to 1.26. The largest deviation equal to 1.73% was recorded in the instance where p = 90, and for p = 236 the algorithm reached the optimal solution. The average computational time was 203 seconds, which equates to approximately 12.04% of the average computational times presented by the software Lingo 13.0.

7.1 Conclusions

This paper presented a new hybrid heuristic approach based on Differential Evolution Algorithms and Tabu Search for the p-median problems and maximum coverage. A proposal is also submitted for adapting the operator of differential mutation of the Differential Evolution algorithm, originally developed for problems where the solution space is continuous.
For the *p*-median problems, tests were conducted in two groups of 42 (forty two) cases, whose solutions used for comparison were obtained by a method which uses the integration of the GIS ArcView Lagrangian/surrogate heuristics as described by Lorena *et al.* [13] and by 13.0 Lingo software, used for obtaining optimal solutions.
For the group of Lorena cases, designed to test *p*-median problems, the DEDHp algorithm obtained, in most of the tested instances, better solutions as compared to those presented by Lorena *et al.* [13], and the computational time required to achieve the best results was lower for about half of the tested cases.
For the group of randomly generated cases, designed to test *p*-median problems, it can be observed that the values of the DEDHp percentage deviations are small, with an average value of approximately 0.53% at its maximum, equal to 2. 23% in the case where n = 500 and p = 100, demonstrating the effectiveness of the algorithm for the instances tackled.
For maximum coverage problems, tests were performed with 22 (twenty-two) cases, whose optimal solutions used for comparison were obtained by software Lingo 13.0. It was found that the average percentage values for the tested instances is below 1%, showing that the DEDHp approaches are competitive for solving this problem in reasonable computational times.
Facility location problems generally require that the algorithm is used during project planning. Thus, it is preferable that the algorithm presents a higher computational cost since it provides improvement in the quality of the solution. As an example, we can mention planning the location of a school unit, where a

decrease of the distance travelled is a daily benefit to users of this unit. Thus, the DEDHp algorithm is more relevant than the method employed by Lorena.
When compared with the optimal solutions, the DEDHp algorithm proved to be robust, presenting low percentage deviations within a computational time, which on average accounted for less than 11% of the those submitted by Lingo 13.0 software.
It was also found that the proposed method shows better computational results for problems where the tested number of vertices is larger.
Future research is intend to include the investigation of the performance of the DEDHp algorithm for other combinatorial problems, such as routing problems, transportation, Knapsack problem and so on.

References

[1] L. F. Almeida; M. A. Pacheco; M. M. Vellasco, in: *Algoritmos Evolucionários na Otimização de Alternativas para o Desenvolvimento de Campos de Petrólio,* XXIII Encontro Nacional de Eng. De produção, Ouro Preto - MG, Brasil. (2003)

[2] A. A. Chaves, in: *Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios*. Universidade Federal de Ouro Preto, Departamento de Computação. Ouro Preto - MG. (2003)

[3] R. Church; C. Revelle, in: *The maximal covering location problem*. Papers of the Regional Science Association, Vol. 32 (1974), p. 101–118.

[4] J. H. Correia; G. F. Sousa; I. Q. Nascimento; L. A. C. Formiga; R. Q. Nascimento, in: *Otimização e implementação de um sistema de alocação ótima de serviços públicos utilizando a metaheurística Grasp.* XLII Simpósio Brasileiro de Pesquisa Operacional, Bento Gonçalves - RS, Brasil, Set. (2010)

[5] N. Christofides, in: *Graf theory – An algorithmic approach*. New York: ed. Academic Press. (1975)

[6] C. R. Freitas; C. M. S. Machado; M. R. Retamoso, in: *Um método baseado na substituição de vértices e teoria espectral para problemas de p-medianas***.** Claio, Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro - RJ, Brasil. (september. 2012)

[7] F. Glover, in: *Future paths for integer programming and links to artificial intelligence.* Computers and Operations Research, Vol. 3 (1986), p.533-549.

[8] F. Glover; M. Laguna, in: *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.

[9] A. Hertz; E. Taillard; D. Werra, in: *A tutorial on tabu search*. Proc. of Giornate di Lavoro AIRO'95, Entreprise Systems: Management of Technological and Organizational Changes (1995), p. 13-24.

[10] A. J. Higgins, in: *A dynamic tabu search for large-scale generalized assignment problems.* Computers and Operations Research. Vol. 28 (2001), p. 1039-1048.

[11] L. Jourdan; M. Basseur; E.G. Talbi, in: *Hybridizing exact methods and metaheuristics: A taxonomy.* European Journal of Operational Research, Vol. 199, n. 3, p. 620-629. (2009)

[12] R. R. Junior; L. B. L. Santos, in: *Análise iterativa dos problemas de p-centros e p-medianas para um crescente número de facilidades: estudo de caso na epidemia de dengue, Salvador.* 9th Conference on Dynamics, Control and their Applications, Jun 07-11 (2010).

[13] A. N. G. Lorena; E. L. F. Senne; J. A. C. Paiva; M.A. Pereira, in: *Integração de modelos de localização a sistemas de informações geográficas.* Gestão & Produção, Vol.8 n.2, São Carlos – SP, Aug. (2001)

[14] E. Minieka, in: *Optimization Algorithms for networks and grafs*. New York: ed. Marcel Dekker, INC. (1978)

[15] R. S. Prado; R. C. P. Silva; F. G. Guimarães; O. Magela, in: *Uma nova abordagem para a evolução diferencial em otimização discreta.* XVIII Congresso Brasileiro de Automática. Bonito-MS. (2010).

[16] K. V. Price; R. M. Storn, in: *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. Berkeley: International Computer Science Institute. (1995)

[17] M. J. F. Souza, in: *Programação de horários em escolas: uma aproximação por metaheurísticas.* Tese (Doutorado), COPPE/UFRJ, Rio de Janeiro - RJ, Brasil. (2000)

[18] A. Subramanian; J. M. F. Medeiros; L. F. Cabral; M. F. Souza, in: *Aplicação da metaheurística busca tabu ao problema de alocação de aulas a salas em uma instituição universitária.* Revista Científica Eletrônica de Engenharia de Produção, Vol. 11, nº 1, UFSC, Florianópolis – SC, Brasil. (2011)

[19] M. F. Tasgetiren; P. N. Suganthan; Q. K. Pan, in: *An Ensemble of Discrete Differential Evolution Algorithms for Solving the Generalized Traveling Salesman Problem.* Applied Mathematics and Computation, Vol. 215 (2010), p. 3356-3368.

[20] Information on http://www.lac.inpe.br/~lorena/ArsigIndex.html

[21] Information on http://paginapessoal.utfpr.edu.br/durski

Received: April 25, 2014