A Hybrid Differential Evolution Algorithm for Solving the Terminal Assignment Problem

Eugénia Moreira Bernardino[1], Anabela Moreira Bernardino[1], Juan Manuel Sánchez-Pérez[2], Juan Antonio Gómez-Pulido[2], and Miguel Angel Vega-Rodríguez[2]

[1] Department of Computer Science, School of Technology and Management, Polytechnic Institute of Leiria, 2400 Leiria, Portugal
{eugenia,anabelab}@estg.ipleiria.pt
[2] Department of Technologies of Computers and Communications, Polytechnic School, University of Extremadura, 10071 Cáceres, Spain
{sanperez,jangomez,mavega}@unex.es

Abstract. The field of communication networks has witnessed tremendous growth in recent years resulting in a large variety of combinatorial optimization problems in the design and in the management of communication networks. One of these problems is the terminal assignment problem. The task here is to assign a given set of terminals to a given set of concentrators. In this paper, we propose a Hybrid Differential Evolution Algorithm to solve the terminal assignment problem. We compare our results with the results obtained by the classical Genetic Algorithm and the Tabu Search Algorithm, widely used in literature.

Keywords: Communication Networks, Genetic Algorithms, Hybrid Differential Evolution Algorithm, Tabu Search, Terminal Assignment Problem.

1 Introduction

The literature on telecommunication network problems has quickly grown. This is mainly due to the dramatic growth in the use of the Internet [1][2]. Terminal assignment (TA) is an important issue in telecommunication networks optimization. The objective is to minimize the link cost to form a network by connecting a set of terminals to a set of concentrators [3]. The 3 constraints imposed in this paper for solving the TA problem are [3]: (1) each terminal must be connected to only one concentrator; (2) the aggregate capacity requirement of the terminals connected to any concentrator cannot exceed the capacity of that concentrator; (3) guarantee the balanced distribution of terminals among concentrators.

The TA problem is a NP-complete combinatorial optimization problem [1]. This means that we cannot guarantee to find the best solution in a reasonable amount of time. The existing, successful methods in approximate optimization fall into 2 classes: local search and population-based search. There are many population-based optimization algorithms and various ways to handle the optimization issues. In this paper we will explore one of the most successful emerging ideas combining local search with a

S. Omatu et al. (Eds.): IWANN 2009, Part II, LNCS 5518, pp. 179–186, 2009.

population based search algorithm. In this article we report the results of the application of a Hybrid Differential Evolution Algorithm (HDE) to the TA problem. We compare the performance of HDE with the Genetic Algorithm (GA) and the Tabu Search Algorithm (TS), widely used in literature.

The paper is structured as follows. In Section 2 we describe the TA problem; in Section 3 we describe the implemented HDE algorithm; in Section 4 we present the studied examples; in Section 5 we discuss the computational results obtained and, finally, in Section 6 we report about the conclusions.

2 The Terminal Assignment Problem

The TA problem involves determine what terminals will be serviced by each concentrator. The constraints imposed in our work to represent this problem are: (1) the terminal sites and concentrators sites have fixed and known locations; (2) the capacity requirement of each terminal is known and may vary from one terminal to another; (3) each concentrator is limited in the amount of traffic that it can accommodate; (4) the capacities of all concentrators and the cost of linking each terminal to a concentrator are also known.

To represent the TA problem we use the following components: (1) a set N of n distinct terminals; (2) a set M of m distinct concentrators; (3) a vector C, with the capacity required for each concentrator; (4) a vector T, with the capacity required for each terminal; (5) a matrix CP, with the location (x,y) of each concentrator; (6) a matrix CT, with the location (x,y) of each terminal.

Figure 1 illustrates an assignment to a problem with N = 10 terminal sites and M = 3 concentrator sites. The figure shows the coordinates for the concentrators, terminal sites and also their capacities.

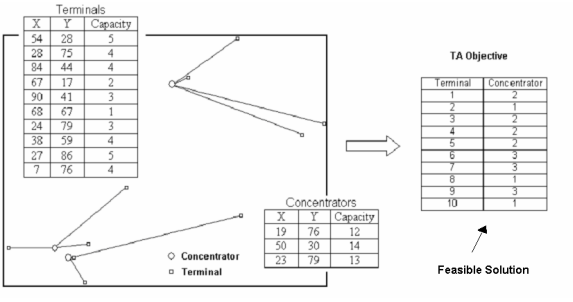

Fig. 1. TA Problem - example

3 The Proposed HDE

The HDE is an evolutionary algorithm (EA) that applies a separate local search (LS) process to refine individuals. Our algorithm combines global and local search by using an EA to perform exploration while the LS method performs exploitation. Combining global and LS is a strategy used by many successful global optimization

approaches, and this type of algorithms has in fact been recognized as a powerful algorithmic paradigm for evolutionary computing [4]. This method has proved to be of practical success in a variety of problem domains. This algorithm is also known as Memetic Algorithm, Hybrid EAs, etc. [5].

HDE uses a Differential Evolution Algorithm (DE) to explore several regions of the search space and simultaneously incorporates a mechanism (LS algorithm) to intensify the search around some selected regions.

DE was introduced by Storn and Price in 1995 [6]. It's a method of mathematical optimization of multidimensional functions and it belongs to the class of evolutionary algorithms. DE explores the candidate solutions encoded in chromosomes and exploits those with better fitness iterally until the stop conditions are reached. The LS algorithm by itself explores the solution space making specific moves in its neighbourhood. The HDE combines those two aspects by using the chromosomes that are produced by DE and optimizes them by a LS algorithm (see Fig. 2).

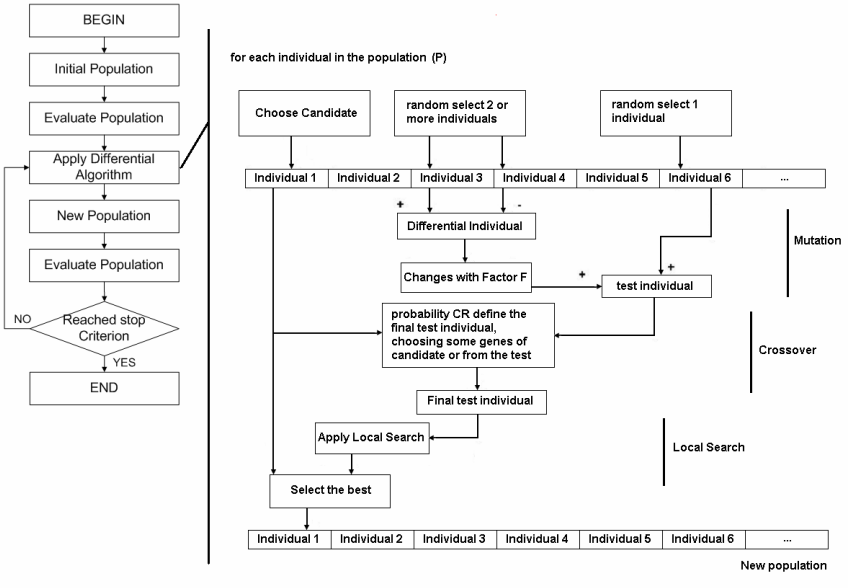

Fig. 2. HDE Algorithm

The DE algorithm is a population based algorithm like GAs, using the similar operators; crossover, mutation and selection. It resembles the structure of an EA, but differs in the generation of new candidate solutions and by the use of a 'greedy' selection scheme. The main difference in building solutions is that GAs rely on the crossover operation while DE relies on the mutation operation. The algorithm uses the mutation operation as a search mechanism and the selection operation to direct the search toward the prospective regions in the search space. The DE algorithm also uses a non-uniform crossover. By using the components of the existing population members to built trial vectors, the recombination operator efficiently shuffles information about successful combinations, enabling the search for a better solution space. The

crucial idea behind DE is a scheme for generating trial parameter vectors. There are several strategies with different approaches (see Table 1).

Table 1. Representation of DE Strategies

Nome	Mutation
Rand1bin	P(i) = ind1 + F * (ind2 – ind3)
Best1bin	P(i) = best + F * (ind1 – ind2)
Rand2bin	P(i) = ind1 + F * (ind2 + ind3 – ind4 – ind5)
Best2bin	P(i) = best + F * (ind1 + ind2 – ind3 – ind4)
RandBest2Bin	P(i) = old + F * ((best – old) + (ind1 – ind2))
Rand1exp	P(i) = ind1 + F * (ind2 – ind3)
Best1exp	P(i) = best + F * (ind1 – ind2)
Rand2exp	P(i) = ind1 + F * (ind2 + ind3 – ind4 – ind5)
Best2exp	P(i) = best + F * (ind1 + ind2 – ind3 – ind4)
RandBest2Exp	P(i) = old + F * ((best – old) + (ind1 – ind2))
Current2Rand	P(i) = old + Cr * (ind1 – old) + F * (ind2 – ind3)
Best3Bin	P(i) = best + F * (ind1 + ind2 + ind3 – ind4 – ind5 – ind6)
RandRandBin	P(i) = ind1 + z * (ind2 – ind3) z is N(0,1) Gaussian variable

The first step for the HDE implementation involves choosing a representation for the problem. In this work, the solutions are represented using integer vectors. We use the terminal-based representation (see Figure 3). Each position in the vector corresponds to a terminal. The value carried by position *i* of the chromosome specifies the concentrator that terminal *i* is to be assigned to.

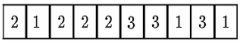

Fig. 3. Terminal Based Representation

The initial population (*P*) can be created randomly or in a deterministic form. The deterministic form is based in the Greedy Algorithm proposed by Abuali *et al.* [7]. The Greedy Algorithm assigns terminals to the closest feasible concentrator. The algorithm can fail to produce a feasible solution when [3]: (1) the total terminal capacity requirement is high than the total concentrator capacity; (2) there is not a feasible solution to the problem instance; (3) the algorithm can't reach a feasible solution.

Procedure Greedy:

```
While additional assignments of terminals to concentrators are possible
    For a randomly chosen terminal, say ti
        Determine the closest feasible concentrator ci
        Assign terminal ti to ci
```

At the mutation step, parameter vectors *ind1*, *ind2*, *ind3*, … (usually three) are selected from population. Note that all vectors in this step are distinct from each other. Mutation continues with adding the weighted difference of two (or more) of the vectors to the third. *F* is a mutation factor and it's selected between 0 and 2. At the recombination step, new individuals are created by combining the mutated vector,

with the old individual vector (*P(i)*). Combination takes place according to the applied strategy [8]. After recombination, if a gene demand of *P(i)* is outside of the allowed demand range it's necessary to apply the following transformation:

IF concentrator > totalconcentrators concentrator = concentrator - totalconcentrators
ELSE IF concentrator <0 Concentrator = totalconcentrators + concentrator

The local search is applied to new individuals generated by the mutation and recombination operators. The LS algorithm consists on the following steps:

BEST-SOLUTION = ACTUAL-SOLUTION
C1 = random (number of concentrators)
C2 = random (number of concentrators)
N = neighbourhoods of ACTUAL -SOLUTION (one neighbourhood results of interchange one terminal of C1 or C2 for one terminal of C2 or C1)
SOLUTION = FindBest (N)
If ACTUAL -SOLUTION is best than SOLUTION
    N = neighbourhoods of ACTUAL-SOLUTION (one neighbourhood results of assign one terminal of C1 to C2 or C2 to C1)
    SOLUTION = FindBest (N)
    If SOLUTION is best than BEST-SOLUTION
        BEST-SOLUTION = SOLUTION
Else
    BEST-SOLUTION = SOLUTION
ACTUAL-SOLUTION = BEST-SOLUTION

It's necessary to evaluate how good a potential solution is relative to other potential solutions. The fitness function is responsible for performing this evaluation and returning a positive number (fitness value) that reflects how optimal the solution is.

The fitness function is based on the fitness function used in [1]:

$$fitness = 0{,}9 * \sum_{c=0}^{M-1} bal_c + 0{,}1 * \sum_{t=0}^{N-1} dist_{t,c(t)} + Penaliza$$

$$bal_c = \begin{cases} 10 & if \left(total_c = round \left(\frac{N}{M} \right) + 1 \right) \\ 20 * abs \left(round \left(\frac{N}{M} \right) + 1 - total_c \right) & \end{cases}$$

c(t)= concentrator of terminal t
t = terminal c = concentrator
M = total number of concentrators
N = total number of terminals

$$Penaliza = \begin{cases} 0 \\ 500 \end{cases} \quad if\ (Feasible) \qquad total_c = \sum_{t=0}^{N-1} \begin{cases} 1 \\ 0 \end{cases} \quad if\ (c(t) = c)$$

$$dist_{t,c(t)} = \sqrt{(CP[c(t)].x - CT[t].x)^2 + (CP[c(t)].y - CT[t].y)^2}$$

The fitness function is based on three different objectives: (1) the total number of terminals connected to each concentrator (the objective is to guarantee the balanced distribution of terminals among concentrators); (2) the distance between the concentrators and the terminals assigned to them (the objective is to minimize the distances between concentrators and terminals assigned to them); (3) the penalization if a solution is not feasible (the objective is to penalize the solutions when the total capacity of one or more concentrators is overloaded). The objective is to minimize the fitness function.

The performance of the child vector and its parent is compared and the better one is selected. If the parent is still better, it is retained in the population.

The algorithm continues until a certain number of generations defined by the user, have passed. Further information on DE can be found in DE Web [9].

4 Studied Examples

In order to test the performance of our approach, we use a collection of TA instances of different sizes. We take 9 problems from literature [11].

Table 2 presents the 9 problems that were used to test our algorithm.

Table 2. TA instances

Problem	NT	NC	TTC	TCC
1	10	3	35	39
2	20	6	55	81
3	30	10	89	124
4	40	13	147	169
5	50	16	161	207
6	50	16	173	208
7	70	21	220	271
8	100	30	329	517
9	100	30	362	518

5 Results

To compare our results we consider the results produced with the classical Genetic Algorithm and with the Tabu Search Algorithm. The GA was first applied to TA by Abuali *et al*. [7]. The GA is widely used in literature to make comparisons with other algorithms. The GA adopted uses "one-point" method for recombination, "change order" method for mutation and tournament method for selection. In "change order", two genes are randomly selected and exchanged. TS was applied to this problem by Xu et al. [10] and Bernardino et al. [11]. We compare our algorithm with the TS algorithm proposed by Bernardino et al.

Table 3 presents the best-obtained results with GA, TS and HDE. The first column represents the problem number (Prob) and the remaining columns show the results obtained (Fitness, Time – Run Times) by the three algorithms.

Table 3. Results

Prob	GA		Tabu Search		HDE	
	Fitness	Time	Fitness	Time	Fitness	Time
1	65,63	<1s	65,63	<1s	65,63	<1s
2	134,65	<1s	134,65	<1s	134.65	<1s
3	284,07	<1s	270,26	<1s	270,26	<5s
4	286,89	<1s	286,89	<1s	286,89	<5s
5	335,09	<1s	335,09	<1s	335.09	<5s
6	371,48	1s	371,12	<1s	371,12	58s
7	401,45	2s	401,49	1s	401,21	118s
8	563,75	4s	563,34	1s	563,19	274s
9	703,78	5s	642,86	2s	642,83	456s

The algorithms have been executed using a processor Intel Core Duo T2300. The HDE and GA were applied to populations of 200 individuals. The initial solutions were created using the Greedy Algorithm. The run time corresponds to the average time that the algorithms need to obtain the best feasible solution. The values presented have been computed based on 100 different executions for each test instance. The three algorithms reach feasible solutions for all test instances. In comparison, the HDE obtains better solutions for larger instances. The TS is the faster algorithm because can find good solutions in a better running time. In HDE the crossover probability is applied to each gene, generating many perturbations by generation, for which the algorithm slows down. Besides, in HDE it's necessary to carry out a concentrator conversion so that the concentrator obtained stays always inside of the defined range. The HDE algorithm has great capacity of convergence. It begins to converge in the first generations. The premature convergence is a disadvantage if the algorithm is trapped in local optima. This happens many times with DE, GA and TS; nevertheless, the local search algorithm used prevents this problem.

The better results obtained with HDE use crossover probability (CR) and factor (F) between [0.1 - 0.4] and [0.1 - 2], respectively. The tests carried out have shown that the strategies that obtain best results are Best1Exp, Best2Exp, Rand1Exp, Rand2Exp, RandBest1Exp and RandRandBin (see Fig. 4). Suggestions from the literature helped to guide our choice of parameter values for the DE variants [12]. Table 4 shows the parameter values that obtain the best solutions (Prob - Problem, CR - Crossover Probability, F - Factor, AppP - Application Probability).

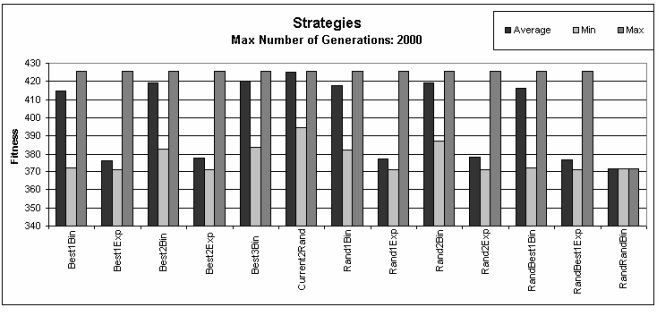

Fig. 4. Problem 6 – Comparison between strategies – Fitness Values

Table 4. DE Parameter Values

Prob	CR	F	Strategy
1	[0,1-1]	[0,1-2]	All
2	0,1	1	Rand2Exp
3	0,1	0,5	RandBest1Exp
4	0,1	0,6	RandBest1Exp
5	1	1	Best1Bin, Best1Exp, RandBest1Bin, RandBest1Exp
6	0,1	0,4	RandBest1Exp
7	0,1	0,2	Best1Exp
8	0,1	0,6	Best1Exp
9	0,1	1	Best1Exp

6 Conclusions

In this paper we present a new Hybrid Differential Evolution Algorithm to solve the Terminal Assignment Problem. We present a DE algorithm combined with a LS algorithm. The performance of' our algorithm is compared with a classical GA and with a TS algorithm. The HDE presents better results for larger problems. Our algorithm provides better solutions with smaller fitness values for larger problems. The TS is the faster algorithm.

The different strategies implemented are also compared. The best strategies are Best1Exp, Best2Exp, Rand1Exp, Rand2Exp, RandBest1Exp and RandRandBin. The different strategies implemented have driven to acceptable results.

In literature the application of DE for this problem is practically nonexistent, for that reason this article shows its enforceability in the resolution of this problem.

The implementation of parallel algorithms will speed up the optimization process.

References

1. Salcedo-Sanz, S., Yao, X.: A hybrid Hopfield network-genetic algorithm approach for the terminal assignment problem. IEEE Transaction On Systems, Man and Cybernetics. 2343–2353 (2004)
2. Yao, X., Wang, F., Padmanabhan, K., Salcedo-Sanz, S.: Hybrid evolutionary approaches to terminal assignment in communications networks. In: Recent Advances in Memetic Algorithms and related search technologies, vol. 166, pp. 129–159. Springer, Berlin (2005)
3. Khuri, S., Chiu, T.: Heuristic Algorithms for the Terminal Assignment Problem. In: Proc. of the ACM Symposium on Applied Computing, pp. 247–251. ACM Press, New York (1997)
4. MA HomePage, `http://www.ing.unlp.edu.ar/cetad/mos/memetic_home.html`
5. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms. Caltech Concurrent Computation Program, C3P Report 826 (1989)
6. Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI (1995)
7. Abuali, F., Schoenefeld, D., Wainwright, R.: Terminal assignment in a Communications Network Using Genetic Algorithms. In: Proc. of the 22nd Annual ACM Computer Science Conference, pp. 74–81. ACM Press, New York (1994)
8. Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
9. Differential Evolution Homepage, `http://www.icsi.berkeley.edu/~storn/code.html`
10. Xu, Y., Salcedo-Sanz, S., Yao, X.: Non-standard cost terminal assignment problems using tabu search approach. IEEE Conference in Evolutionary Computation 2, 2302–2306 (2004)
11. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: Tabu Search vs Hybrid Genetic Algorithm to solve the terminal assignment problem. In: IADIS International Conference Applied Computing, pp. 404–409. IADIS Press (2008)
12. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005)