A Hybrid Evolutionary Algorithm for the p-Median Problem

István Borgulya
University of Pécs, Faculty of Business
and Economics, Hungary
H-7621 Pécs, Rákóczi út 80
Tel: 36 72 501599
e-mail:borgulya@ktk.pte.hu

ABSTRACT
A hybrid evolutionary algorithm (EA) for the p-median problem consist of two stages, each of which is a steady-state hybrid EA. These EAs encode selections of medians as subsets of the candidate sites, apply a recombination operator tailored to the problem, and select symbols in chromosomes to mutate based on an explicit collective memory (named virtual loser). They also apply a sequence of two or three local search procedures to each new solution. Tests e.g. on the benchmark problem instances of ORLIB returned results within 0.03% of the best solutions known.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - *heuristic methods*

General Terms
Algorithms.

Keywords
p-median problem, parallel computing, local search, evolutionary algorithm.

1. INTRODUCTION
The p-median problem is the most commonly researched problem in the area of location analysis. In the *p*-median problem there is a set of demand locations and potential facility sites and the aim is to identify the locations of a given number of facilities, *p*, in such a way as to minimize the total distance that demand must traverse to reach its nearest facility. Mathematically, this can be described as follows: Given the set $L=\{v_1, v_2, ..., v_m\}$ of potential locations for the facilities and the set $U=\{u_1, u_2, ..., u_n\}$ of demands, the entries of an $n \times m$ matrix $D=(\mathrm{Dist}(u_i, v_j))_{n \times m}$ give the distances between u_i and v_j, for all $v_j \in L$ and $u_i \in U$.
The objective of the p-median problem is to minimize the sum of these distances, i.e.

$$\text{minimize } f(X) = \sum_{ui \in U} \min_{vj \in X} Dist(ui, vj)$$

where $X \subseteq L$ and $|X|=p$. In our interpretation L is equal with U.

In practice, the *p*-median problem is used in many areas such as material distribution, transportation, location of industrial plants or public facilities. This problem is *NP*-hard and many heuristics and exact methods have been proposed to solve it.

In this paper we present a hybrid evolutionary algorithm for the p-median problem, named EPMED (Evolutionary algorithm for the p-median problem).

GECCO'05, June 25-29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

2. THE EPMED ALGORITHM
EPMED has a two stage algorithm structure, where all stages are steady-state hybrid EAs. In the first stage, to speed up the convergence, the algorithm generates new individuals periodically. A new individual is generated the same way as a descendant, but it will increase the number of the population size. Thus, in the first stage new individuals are periodically generated until a maximum number of individuals are completed. In the second stage the algorithm continues the application of the operations of the first stages, but the population-size isn't changed yet (We found similar structure in [3]).

The main functions and characteristics of the EAs are the following:

Representation. We encode selections of medians as subsets of the candidate sites (e.g. (3, 5, 19, 120) by p=4).

Initial population. The same population and individuals are used in all stages. The first individuals of the P *population* are randomly generated. At beginning the population size is less than the maximum population size.

Fitness function. The algorithm uses the objective function *f(x)* as the fitness function.

Selection operator. In all stages the algorithm selects with tournament selection two different parents.

Recombination operator. The algorithm chooses the two parents with p_r probability from the mating pool and the descendent is constructed with a special recombination operator, which is a simpler version of the *fusion* operator of Beasley [2]. Otherwise it chooses the best parent from the mating pool and the descendent is built by copy-making.

Mutation operator. The algorithm makes some swaps by mutation with p_m probability. For a swap the mutation operator selects randomly one of the open facilities and moves this to another site. The new site is also picked randomly from the empty possible places to locate facilities. Only the *virtual loser (VL)* modifies the random selection of the open facilities: the probability of the selection of an open facility depends on the *VL* (We choose to adopt the method of [7] that memorises the past failures of evolution through a virtual individual, the *virtual loser)*.

Local search procedure. A lot of metaheuristics use a variant of the *fast interchange* local search procedure as a subroutine. E.g. the first is the *Variable Neighborhood Search* (*VNS*) [4]. The *VNS* uses the *fast interchange* method with a minor difference (We call this variant *FIHM)*. The second is a *GRASP* version [6], that uses a variant of the *fast interchange* method too (We call this variant *FIRW)*. The third is the *Variable Neighborhood Decomposition Search* [5], where the local search method is used in some subspace instead of applying it in the whole solution space (We call this variant *FIDS)*.

Our algorithm has better performance if we apply more local search procedures to each new solution. So we apply a sequence of two or three local search procedures (CALL *FIRW*, CALL *FIHM* or CALL *FIRW*, CALL *FIHM*, CALL *FIDS*). We apply an optimized version of the recombination operator using the *FIDS* procedure too.

Reinsertion. The algorithm compares the descendent with the best parent. If the descendent is better then the parent, the parent is deleted and the descendent will be a new individual. If the number of the individuals is less then the maximum population size (we increase the number of the individuals), the descendent is inserted into the population without comparison to the parent.

Deleting. In the seconds stage the *Deleting* procedure deletes a given percentage of the weakest solutions periodically.

Restart procedure. In the second stage if no new best individual in the population is found for more than 50 generations, the EPMED begins the second stage with a smaller population. All individuals except the best 30 % of the population are deleted.

Stopping criteria. The algorithm is terminated if the running time (in CPU seconds) is more than a prefixed time limit.

Table 1. Comparative results for the problems

Methods	**ORLIB**		**Koerkel**		**fl1400**	
/Tasks	AD	AT	AD	AT	AD	AT
Hybrid	0.022	572			1.48	12479
EPMED	0.075	3981	1.72	11286	10.34	32419
VNS	0.260	3135			3.35	27591
TS-2	1.420	2715			6.52	20650
ADE	4.400	736	1.85	2373		

3. COMPARATIVE TEST RESULTS

We have tested the EPMED with four sets of test problems: the benchmark set of the ORLIB, the problems from Koerkel [1], two instances from the TSPLIB: the problem fl1400 and the problem pcb3038 (The EPMED was implemented in Visual Basic and ran on a Pentium 4 1.8 GHz with 256 MB RAM).

As for comparison, we chose different methods: the TS heuristics from [4] (TS_2), the genetic algorithm from [1] (ADE), the *VNS* by Hansen et al. [4] and a hybrid heuristic from [6] (Hybrid). All method ran on different computers.

Table 1 includes the results of three test-problem sets, and it presents mean values calculated from a number of runs: all problems (tasks) were run 10 times. We give the average relative percentage deviation of the solution from the best known solution (AD), and the average running time to the best solutions (AT) (CPU time in seconds). (The running times do not include the time to compute the all-pairs shortest paths from the graph read and the preprocessing time of the *FIRW* procedure). Analyzing the values AD of Table 1 we can confirm that the Hybrid algorithm has the best results, and the *VNS* has the second best results in general. In the case of the ORLIB problems our method has the second best results.

Both the quality of the results and the running time can be improved using an island model (We used a complete topology between 2-8 processors, chosen the best 1-4 individuals for migrating). The island model for EPMED was simulated on our PC. We tested the model with 10 instances from the fl1400 problem ($10 \leq p \leq 100$). For comparison, we chose the same methods as earlier (Table 2). We can confirm that the quality of the results (AD) in our island model (with 8 processors) is better as the results of the *VNS*. Based on this result we hope better results on other test problems too.

Table 2. Comparative results by the island model

TS-2		**VNS**		**Hybrid**		**EPMED**	
AD	AT	AD	AT	AD	AT	AD	AT
3.49	9001	1.33	12208	1.02	880	1.09	7458

4. ACKNOWLEDGMENTS

The Hungarian Research Foundation OTKA T 042448 supported the study.

5. REFERENCES

[1] Alp, O., Erkut, E., Drezner, Z.: An Efficient Genetic Algorithm for the *p*-Median Problem. Annals of Operations research 122. 2003. 21-42

[2] Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. European Journal of Operational Research, 94. 1996. 392-404.

[3] Borgulya, I.: A Heuristics Method for the Quadratic Assignment Problem. Central European Journal of Operations research 11(1) 2003. 3-16.

[4] Hansen, P., Mladenović, N., Perez-Brito, D.: Variable neighborhood decomposition search. Journal of Heuristics. 7(3) 2001. 335-350.

[5] Hansen, P., Mladenović, N.: Variable Neighborhood Search for the p-median. Location Science, Vol.5 No. 4. 1997. 207-226.

[6] Resende, M.G.C., Werneck, R.F.: A Hybrid Heuristic for the *p*-Median Problem. AT&T Labs research Technical Report TD-5NWRCR, 2003.

[7] Sebag, M., Schoenauer, M., Ravisé, C.: Toward Civilized Evolution: Developing Inhibitions. In: Bäck, T. (ed): Proc. of the 7th International Conference on Genetic Algorithm. Morgan Kaufmann Pub. San Francisco, 1997 291-298.