

# Evolutionary Codings and Operators for the Terminal Assignment Problem

Bryant A. Julstrom  
Department of Computer Science  
St. Cloud State University  
St. Cloud, MN 56301 USA  
julstrom@stcloudstate.edu

## ABSTRACT

Given a collection of terminals, each with a demand, a collection of concentrators, each with a capacity, and costs of connecting the terminals to the concentrators, the terminal assignment problem seeks a set of such connections of minimum cost and without the total demand at any concentrator exceeding its capacity. One genetic algorithm for this problem encodes candidate solutions as strings of concentrator labels; in three other GAs, chromosomes are permutations of terminal labels decoded by a greedy decoder. In comparisons on 40 instances of the problem, the string-coded GA consistently performs poorly, while among the three permutation-coded GAs, one that applies only mutation almost always outperforms the other two, which use crossover operators as well as mutation.

## Categories and Subject Descriptors

G.2.1 [Mathematics of Computing]: Discrete Mathematics—Combinatorics; I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic Methods

## General Terms

Algorithms

## Keywords

Terminal assignment problem, string coding, permutations, operators

## 1. INTRODUCTION

In the terminal assignment problem (TAP), a communications network will connect  $n$  leaf nodes, called *terminals* and each with demand  $w_i$ , via  $m$  interior nodes, called *concentrators* and each with capacity  $p_j$ . No terminal's demand exceeds the capacity of any concentrator. The cost of connecting terminal  $i$  to concentrator  $j$  is  $c_{ij}$ . We seek an assignment of terminals to concentrators under which the total demand at each concentrator does not exceed its capacity and whose total cost is a minimum.

Evolutionary algorithms for the TAP have been described by Abuali, Schoenefeld, and Wainwright [1], Khuri and Chiu [3], Salcedo-Sanz and Yao [4], and others. Here, four GAs

encode candidate solutions as strings of concentrator labels and as permutations of terminal labels decoded by a greedy decoder. The more effective of these use the latter coding.

## 2. TWO CODINGS, FOUR ALGORITHMS

One genetic algorithm uses a coding described by Khuri and Chiu [3]: A chromosome  $c[\cdot]$  is a string of length  $n$  over the set of concentrator labels:  $c[i] = j$  indicates that terminal  $i$  is assigned to concentrator  $j$ .

These chromosomes may represent invalid solutions, so Khuri and Chiu implemented a penalty function that added two terms to invalid chromosomes' fitnesses: the product of the number of terminals and the maximum distance on the grid that contains the terminals and concentrators, and the product of the number of concentrators whose capacities are exceeded and the total of that excess. Variation operators are two-point crossover and position-by position mutation.

Abuali, Schoenefeld, and Wainwright [1] described a coding in which a chromosome is a permutation of the terminals, and a greedy decoder identifies the corresponding assignment of terminals to concentrators. The decoder scans the terminals in chromosome order and assigns each to the nearest concentrator with sufficient capacity.

An appropriate crossover operator is partially mapped crossover (PMX) [2]. Another crossover operator is alternation: merge the two parental permutations and delete duplicates. Mutation swaps two random values.

The string coding and the permutation coding were implemented in four genetic algorithms for the TAP, all of the same design: An initial population of random chromosomes; parent selection in  $k$ -tournaments; independent application of crossover and mutation; 1-elitism; and a fixed number of generations. The string-coded algorithm is called GA0. Among the permutation-coded algorithms, GA1 applies PMX and mutation; GA2 applies alternation and mutation; GA3 applies only mutation.

On instances of the TAP with  $n$  terminals and  $m$  concentrators, each GA's population contained  $2n$  chromosomes. Each chose parent chromosomes in tournaments of size two. Each generated offspring by crossover with probability 60% and by mutation with probability 40%, except for GA3. Each ran through  $20n$  generations.

## 3. COMPARISONS

Four sets of ten instances placed  $n$  terminals and  $m$  concentrators on a  $300 \times 300$  grid, with the concentrators scattered randomly, the terminals placed either randomly or bi-

**Table 1: Results of the sets of 50 trials of the four genetic algorithms on the TAP instances with  $n = 200$  terminals and  $m = 64$  concentrators. In the first ten instances, terminals’ positions are random; in the second ten, they are biased to the left. In each row, the best result on one trial is italicized, and the mean of the trials’ best results is bold face.**

Instance	GA0			GA1			GA2			GA3		
	Best	Mean	StD	Best	Mean	StD	Best	Mean	StD	Best	Mean	StD
0	4183.0	4426.3	90.3	<i>4105.1</i>	4108.2	4.5	<i>4105.1</i>	4108.6	3.5	<i>4105.1</i>	<b>4108.0</b>	4.2
1	4983.0	8104.7	14641.5	<i>4854.6</i>	4860.5	3.8	4861.6	4866.5	3.2	<i>4854.6</i>	<b>4857.9</b>	3.0
2	5463.9	56029.7	36474.1	5116.3	5147.5	18.9	5147.3	5200.4	23.1	<i>5109.7</i>	<b>5128.6</b>	13.9
3	5344.8	73828.3	30077.8	4815.9	4838.7	12.6	4839.6	4903.4	30.9	<i>4815.7</i>	<b>4828.6</b>	10.6
4	4930.9	64589.8	33803.3	<i>4523.4</i>	4539.8	12.0	4557.2	4592.8	17.9	<i>4523.4</i>	<b>4534.4</b>	11.5
5	4752.2	44287.8	39637.1	<i>4549.9</i>	4566.8	11.5	4559.5	4580.9	10.9	<i>4549.9</i>	<b>4553.7</b>	4.7
6	4219.3	46001.4	40266.4	<i>4050.6</i>	4059.6	5.7	<i>4050.6</i>	4061.2	6.1	<i>4050.6</i>	<b>4057.6</b>	3.3
7	4455.0	6081.2	10594.8	<i>4347.5</i>	4348.0	0.6	<i>4347.5</i>	4348.0	0.1	<i>4347.5</i>	<b>4347.8</b>	0.2
8	4517.2	4662.4	67.6	<i>4394.2</i>	4395.6	2.7	<i>4394.2</i>	<b>4394.4</b>	0.4	<i>4394.2</i>	4394.5	1.2
9	4712.8	27523.0	36574.8	<i>4480.9</i>	4497.5	11.2	4491.0	4523.1	10.7	<i>4480.9</i>	<b>4485.8</b>	4.9
10	79535.6	79900.3	135.2	4665.8	4705.3	25.3	4724.6	4815.3	35.4	<i>4644.9</i>	<b>4690.1</b>	25.9
11	84572.6	84731.5	98.6	<i>6024.6</i>	<b>6093.3</b>	36.7	6160.5	6275.2	46.7	6048.0	6106.8	36.7
12	7632.1	85324.8	11212.5	5634.8	5687.0	29.1	5772.7	5880.1	54.5	<i>5603.2</i>	<b>5684.1</b>	33.4
13	85942.9	86446.7	182.5	<i>6807.2</i>	<b>6924.4</b>	65.8	7073.8	7224.3	73.7	6826.9	6964.7	64.1
14	85745.8	85936.5	88.4	<i>4970.5</i>	5016.2	24.0	5045.2	5122.4	35.5	<i>4970.5</i>	<b>4996.0</b>	17.2
15	86446.2	86907.9	197.4	<i>9565.3</i>	<b>9684.5</b>	59.5	9936.7	10140.7	92.0	9657.9	9830.0	75.4
16	90920.1	91193.0	151.0	<i>9032.1</i>	<b>9141.8</b>	62.1	9393.8	9619.8	100.0	9112.3	9244.1	60.6
17	88205.5	88552.7	149.2	<i>6738.4</i>	<b>6828.4</b>	47.4	6947.5	7107.4	72.2	6740.6	6863.4	53.9
18	8601.4	83803.6	10853.2	<i>6010.9</i>	<b>6080.8</b>	46.2	6146.1	6304.0	75.1	6017.6	6086.6	42.4
19	92377.1	92753.5	214.1	<i>9903.6</i>	<b>10048.3</b>	73.8	10341.5	10511.1	77.3	10068.3	10203.1	62.3

ased to the left, and minimum distance between the elements of seven units. Terminals’ demands were integers chosen with uniform probabilities from 3 to 10, and concentrators’ capacities were similarly chosen from 15 to 25.

Two sets of ten instances have  $n = 100$  terminals and  $m = 32$  concentrators, with random and biased terminal positions, respectively. Similarly, two sets of ten instances have  $n = 200$  terminals and  $m = 64$  concentrators. Each GA was run 50 independent times on the 40 instances. Table 1 summarizes the results of the trials on the larger instances.

On the smaller unbiased instances, GA3 returned the best single and mean results on every trial. The mean results of GA1 and GA2 were inferior. GA0’s mean results were often much larger, reflecting the contribution of the penalty function. On some trials, GA0 failed to identify any valid solutions. On the smaller biased instances, the results were similar, but GA3’s advantage was less pronounced.

On the larger unbiased instances, as shown by Table 1, GA3 achieved the best single result on every instance and the best mean result nine times out of ten. GA1 tied GA3 for best single result on nine instances, and again GA0 performed poorly; it often failed to find any valid solutions.

On the larger biased instances, GA1 slightly outperformed GA3. GA1 returned the best single result eight times out of ten and the best mean result seven times, while GA3 returned the best single and mean results on three instances. GA2 performed less well, and GA0 rarely identified valid solutions.

## 4. OBSERVATIONS

GA0’s poor performance is not surprising. Its search space is the set of all  $m^n$  assignments of terminals to concentrators;

in general, valid solutions constitute only a small proportion of this space. Though there are  $n!$  permutations of  $n$  terminals, the decoder of GA1, GA2, and GA3 implements a mapping of these chromosomes onto *valid* solutions only, and its greediness biases the search towards low-cost solutions.

## 5. REFERENCES

- [1] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. Terminal assignment in a communications network using genetic algorithms. In *Proceedings of the 22nd Annual ACM Computer Science Conference*, pages 74–81, New York, 1994. ACM.
- [2] D. E. Goldberg and J. Robert Lingle. Alleles, loci, and the traveling salesman problem. In J. J. Greffentette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.
- [3] S. Khuri and T. Chiu. Heuristic algorithms for the terminal assignment problem. In B. Bryant, J. Carroll, D. Oppenheim, J. Hightower, and K. M. George, editors, *Proceedings of the 1997 ACM Symposium on Applied Computing*, pages 247–251, New York, 1997. The ACM Press.
- [4] S. Salcedo-Sanz and X. Yao. A hybrid Hopfield network-genetic algorithm approach for the terminal assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 34(6):2343–2353, 2004.