*Proceedings of the 22nd Annual Computer Science Conference*, March 8-10, 1994, Phoenix, AZ, pp. 74-81.

# Terminal Assignment in a Communications Network Using Genetic Algorithms

Faris N. Abuali, abuali@euler.mcs.utulsa.edu
Dale A. Schoenefeld, dschoen@euler.mcs.utulsa.edu
Roger L. Wainwright, rogerw@penguin.mcs.utulsa.edu

Department of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, Oklahoma 74104-3189

## ABSTRACT

Genetic algorithms (GA) are investigated as a heuristic technique for obtaining a near optimal solution to the terminal assignment (TA) problem. The TA problem involves determining minimum cost links to connect a given collection of terminal sites to a given collection of concentrator sites in order to create a communications network. The cost of the link from each terminal site to each concentrator site is known. The capacity requirement of each terminal is known and may vary from one terminal to another. The capacities of all concentrators are assumed to be the same. The constraint in the TA problem is that the aggregate capacity requirement of the terminals connected to any one concentrator must not exceed the capacity of that concentrator. The cost of the concentrators is not considered. The problem is NP-complete. This research has applications to centralized information network design and can reduce the cost significantly for communications network installations. Two chromosome encodings were developed and the resulting GA implementation of the TA problem outperformed a widely used greedy algorithm in all of the test cases. In tightly constrained cases, the GA implementation was able to find good solutions while the greedy algorithm was often unable to find a feasible solution.

## 1. Introduction

In this paper the authors investigate genetic algorithms (GA) as a heuristic technique for obtaining a near optimal solution for the terminal assignment (TA) problem. The TA problem involves minimizing the cost of connecting a collection of terminal sites to a collection of concentrator sites in order to create a communications network. The requirement is that each terminal site be connected to one and only one concentrator site. For each terminal, the problem is to identify the concentrator to which the terminal should be attached in order to minimize the cost of the connections. The cost of connecting each terminal to each concentrator is known. The capacity requirement of each terminal is known and may vary from one terminal to another. The capacities of all concentrators are assumed to be the same. The constraint in the TA problem is that the aggregate capacity requirement of the terminals connected to any one concentrator must not exceed the capacity of that concentrator. The cost of the concentrators is not considered. This problem is NP-complete. [12] This research has applications to centralized information network design and can reduce significantly the cost for communications network installations.

A greedy algorithm, with a tradeoff computation that gives preference to terminals which pay a big premium for not connecting to their nearest concentrator, is widely used to obtain a near optimal cost terminal assignment that satisfies the constraint. This research develops two different GA chromosome representations for the TA problem. Subsequently, this research compares the results of the greedy algorithm with the results of the GA algorithm using the two chromosome representations with several standard crossover and mutation operators. The data sets for terminal locations and the concentrator locations were randomly generated and Euclidean distance was used as the cost. A randomly generated capacity requirement was associated with each terminal.

## 2. The Terminal Assignment Problem

In Figure 1, we indicate a collection of T = 10 terminal sites and C = 3 concentrator sites. The site locations are to scale on a 100 x 100 Euclidean grid. The capacity requirement and the coordinates for each terminal site are indicated in Figure 1(a). The coordinates for the concentrator site locations are indicated in Figure 1(b). In this research, the capacity of each concentrator is assumed to be W = 12. The cost of connecting site i to site j, $cost_{i,j}$, is assumed to be the result of rounding Euclidean distance to the nearest integer. The matrix of *cost* values for this illustration is shown in Figure 1(c).

Figure 2 shows the assignment of terminal sites to concentrator sites as determined by the greedy algorithm using a tradeoff computation. The total cost , computed as the sum of the link costs for the 9 selected links, is 260. In this case, the greedy algorithm is unable to assign terminal 10.

Another collection of links between the terminal sites and the concentrator sites is indicated in Figure 3. The assignment of terminal sites to concentrator sites in Figure 3 is the one determined by a genetic algorithm using both the LC1 and the LC2 chromosome representation strategies developed as part of this research. The total cost for the assignment of terminals illustrated in Figure 3 is 276. By exhaustive case analysis, the optimal terminal assignment was determined to be the same as the solution found by the genetic algorithm.

The terminal assignment (TA) problem is to find the collection of links having the least cost subject to the constraint that the capacity of each concentrator is not violated. Formally, the problem can be stated as follows [12]:

Find

$$Z = min \sum_{j=1}^{C} \sum_{i=1}^{T} cost_{ij} \, X_{ij}$$

subject to:

$$\sum_{j=1}^{C} X_{ij} = 1 \qquad i = 1, 2, ..., T \qquad (1)$$

$$\sum_{i=1}^{T} W_i \, X_{ij} \leq W \qquad j = 1, 2, ..., C \qquad (2)$$

Z is the cost of all of the links in the network. C is the number of concentrators, T is the number of terminals, and $cost_{ij}$ is the distance from terminal i to concentrator j. $X_{ij}$ is 1 if terminal i is assigned to concentrator j and $X_{ij}$ is 0 otherwise. $W_i$ is the capacity requirement of terminal i and W is the capacity of each concentrator. The first constraint, from equation (1) above, guarantees that each terminal is associated with one and only one concentrator. The second constraint, from equation (2), guarantees that the capacity constraint on each concentrator is not violated.

## 3. The Greedy Algorithm

A simple heuristic algorithm for the problem is to assign terminals to "nearest feasible" concentrators in a greedy fashion. Feasibility refers to the ability of the concentrator to service terminal capacity requirements. Specifically, a greedy algorithm is [12]

```
while additional assignments of terminals to
concentrators are feasible

{ for each unassigned terminal, say t_i,
   { determine cost_{i,j_i} , the distance from t_i to c_{j_i}
     where c_{j_i} is the closest feasible concentrator for
     terminal t_i ;
   }
   let i' be the value for i that minimizes cost_{i,j_i} ;
   assign terminal i' to concentrator j_{i'} ;
}
```

An algorithm can fail to produce a feasible solution when: (1) the total concentrator capacity is less than the total of the terminal capacity requirements, (2) the total concentrator capacity is greater than the total of the terminal capacity requirements, but there is no feasible assignment of the terminals to the concentrators, or (3) the total concentrator capacity exceeds the total terminal capacity and there is a feasible assignment of the terminals to the concentrators, but the algorithm fails to find one. Even when the assignment of all terminals is feasible, the above greedy algorithm can force the terminals that are considered last to be assigned to concentrators that are very far away.

A modification of the algorithm computes a "tradeoff" that can lead to assigning a terminal, say $t_{i_2}$, to a given concentrator even in the presence of another closer unassigned terminal, say $t_{i_1}$. The modification may be advantageous if the alternative would be to attach $t_{i_1}$ to the given concentrator and to subsequently attach $t_{i_2}$ to a second concentrator that is much further away. Specifically, the modified algorithm is referred to as the *greedy algorithm with tradeoff* $\alpha$ . The parameter $\alpha$ is between 0 and 1. The algorithm is as follows [12]:

```
while additional terminal assignments are feasible

{ for each unassigned terminal, say t_i,
   { determine cost_{i,j_i1} and cost_{i,j_i2} , the distance
     from t_i to c_{j_i1} and c_{j_i2} where
     c_{j_i1} and c_{j_i2} are the first and second closest
     feasible concentrators to terminal t_i ;
     let tradeoff_i = cost_{i,j_i1} - α cost_{i,j_i2} ;
   }
   let i' be the value of i that minimizes tradeoff_i ;
   assign terminal i' to concentrator j_{i'1} ;
}
```

The algorithm is implemented easily and efficiently by maintaining heaps. The links resulting from applying the greedy algorithm with tradeoff $\alpha = 0$ to the data in Figure 1 is shown in Figure 2. As indicated previously, the

algorithm does not assign terminal 10. In fact, the *greedy algorithm with tradeoff* does not produce any feasible assignment for $\alpha$ between 0 and 1.

The problem addressed assumes that the terminals have unequal capacity requirements. If one alternately assumes that the capacity requirements of the terminals are equal, the problem is no longer NP complete, and alternating chain algorithms can be used to improve the *greedy algorithm with tradeoff* $\alpha$. The alternating chain algorithms produce a provably optimal solution for the terminal assignment problem when the capacity requirements of the terminals are equal. [12]

## 4. Genetic Algorithms

Genetic algorithms differ from traditional algorithms in several ways. The genetic algorithm works with a population of encodings for some parameter rather than with the actual parameter. The genetic algorithm uses probabilistic transition rules and the application of the genetic operators causes information from the previous generation to be carried over to the next. Furthermore, genetic algorithms produce "close" to optimal results and they make no assumptions about the problem space. In addition, genetic algorithms are often simple to develop and they are suitable for parallel processing.

In a genetic algorithm the parameters of the model to be optimized are encoded into a finite length string called a chromosome. The fitness of a chromosome determines its ability to survive and reproduce offspring. The genetic algorithm creates an initial population of solutions and then recombines them in such a way to guide the search to the most promising areas of the state space. The transition rules that produce one population from another are called genetic recombination operators. These include Crossover, Reproduction and Mutation. Crossover provides new points in the solution space to investigate. Mutation, which usually occurs rarely, guarantees that the entire search space has the opportunity to be searched, given enough time. Initial population size, crossover method, evaluation function, and mutation rate are parameters which have a great impact on convergence rate as well as the quality of the solution [3].

Several researchers have investigated the benefits of solving combinatorial optimization problems using genetic algorithms [2,3,5,6,7,13,17]. Genetic algorithm packages for a single processor have been available for several years. A generational GA such as GENESIS [10], and a steady-state GA such as GENITOR [15,16] and LibGA [4], which offers the ability to use both a generational or a steady-state approach, are three examples of packages that are readily available. In a steady-state GA the parents and offspring are in the same pool. Each time an offspring is generated it is placed into the pool, and the weakest chromosome is removed from the pool. In a generational GA, all of the offspring are saved in a separate pool until the pool size is reached. Then the children's pool replaces the parent's pool for the next generation. The generational approach and the steady state approach represent two extremes in pool management for genetic algorithms. It is assumed the reader is familiar with the fundamentals of genetic algorithms. Davis, Goldberg, and Rawling provide an excellent in depth study of genetic algorithms [5,6,9,14].

## 5. The Test Cases

The *greedy algorithm with tradeoff* $\alpha$, for $\alpha = 0.0$ to 1.0 in increments of 0.1, and various genetic algorithm (GA) strategies were applied to ten different collections of 100 terminal sites and to one collection each of 200, 300, and 400 terminal sites. In each case, the capacity of each concentrator was assumed to be W = 12, and the capacity requirement (weight) of each terminal was selected to be a random integer in the range 1 - 6 . In each case, the quotient, Q, resulting from dividing the total terminal capacity requirement by the capacity of a concentrator was computed. The number of concentrators was selected to be the least integer that was 7% in excess of Q. The computed number of concentrators for each data set is recorded in Table 1. The site locations for both the terminal sites and the concentrator sites were randomly generated on a 100 unit x 100 unit rectangular grid. In each case, the cost of a link between a terminal site and a concentrator site was assumed to be the integer nearest to the Euclidean distance.

Subsequently, for comparison, concentrators of size W = 24 and W = 48 were used. In fact, the terminal locations and the terminal sizes, random integers in the range 1 - 6, were the same as for case VI of the ten data sets of 100 terminals. For each of W = 24 and W= 48 the number of concentrators was selected to be the least integer that was 14% in excess of Q, since the greedy was unable to find a solution for the least integer that was 7% in excess of Q. Execution of the genetic algorithm for each of W = 24 and W = 48 was repeated ten times.

## 6. GA Implementation for the TA Problem

The authors are not aware of any previous genetic algorithm implementation for the TA problem. Two different chromosome representations for the GA implementations were developed. Both representations are permutations of the integers 1 through T, the number of terminal sites. The first representation is referred to as the *least cost 1* (LC1) encoding and the second representation is referred to as the LC2 encoding. The chromosome representations for the LC1 and the LC2 encodings are presented in the next two sections. In each case, the encoding is a permutation encoding with a greedy decoding strategy. The PMX and the Cycle [9] crossover functions were used with each of the two encoding strategies. A fixed mutation rate of 0.1 and, alternately, an adaptive mutation rate were used for each of the representation and crossover possibilities. A generational genetic algorithm strategy with roulette wheel selection was used for evolving the populations in all cases.

An alternate *group number encoding* was considered but was not used because previous work on a related problem indicated that it would be less likely to succeed in comparison to either of the LC encodings [1]. With a group number encoding, the length of the chromosome is equal to the number of terminal sites. The chromosome allele positions would correspond to the terminal sites and the content of an allele is an integer indicating the concentrator site to which the terminal site is assigned. [11]

## 7. A Least Cost Permutation Encoding (LC1)

Jones and Beltramo [11] investigated partitioning problems using genetic algorithms. The goal is to partition the permutation into k groups according to some criterion. They presented several methods for encoding a partition as a permutation of N objects. One of the methods is a greedy heuristic which they attribute to L. Davis. The greedy heuristic takes the first k objects in the permutation to initialize the k groups. The remaining objects of the permutation are then added to the groups one at a time. Each of the remaining objects is added to the group that yields the best value for the objective function.

The first permutation encoding, LC1, is an application of the Davis encoding method to the TA problem. A chromosome is a permutation of the T terminal sites to be partitioned into C subsets according to a greedy assignment strategy. C is the number of concentrators. For LC1, the number of concentrator sites actually used is C and does not change as the population of chromosomes evolves, even though using a proper subset of the concentrator sites might further improve the cost. The original intent was to use only the concentrator sites that were actually used by the best application of the *greedy algorithm with tradeoff* $\alpha$. However, due to the tightness of the number of concentrators in relation to the capacity requirement of the terminal sites, the random locations of the terminal sites and the concentrator sites, and the assumption that there is no cost associated with the concentrators, it turned out that the best *greedy algorithm with tradeoff* $\alpha$ used all concentrator sites in all test cases where the algorithm was successful.

The evaluation function assumes that the first C terminals are assigned to C different concentrators, one terminal per concentrator as suggested by the Davis encoding method. In particular, the first terminal is assigned to the first concentrator, the second terminal is assigned to the second concentrator, etc., even though the terminals might be assigned initially to concentrators that are quite far away. The remaining T - C terminal sites are examined individually. As each new terminal site is considered, the cost consequence of assigning it to each feasible concentrator is determined. The terminal is assumed to be assigned to the feasible concentrator for which the cost is minimum. The process iterates.

The following chromosome illustrates the LC1 permutation encoding for the best observed chromosome corresponding to the T = 10 and C = 3 configuration presented in Figure 1: 9 3 7 10 8 6 4 2 5 1
The chromosome is interpreted to mean that $t_9$, $t_3$, and $t_7$ are assigned to $c_1$, $c_2$ and $c_3$, respectively. The remaining capacities of $c_1$, $c_2$, and $c_3$ become 6, 9, and 8. One begins consideration of $t_{10}$, having a capacity requirement of 4, and determines that each concentrator is still feasible for $t_{10}$. Since the costs from $t_{10}$ to concentrators $c_1$, $c_2$, and $c_3$ are 46, 91, and 34, one assigns $t_{10}$ to $c_3$. The remaining capacities of $c_1$, $c_2$, and $c_3$ then become 6, 9, and 4. One iterates the process by considering $t_8$. The chromosome fitness is the total cost of the links that assign each terminal to the corresponding concentrator. If, while considering some allele, there is no feasible assignment to a concentrator, the chromosome is infeasible and the fitness reflects a penalty.

The chromosomes for the initial population were randomly generated. To improve the number of feasible chromosomes in the initial population, alleles in positions C+1 through T of each chromosome were sorted in decreasing order. The first ten percent (i.e. 10% of T) of the alleles beginning in position C+1 of a chromosome were swapped with the same number of alleles beginning in position 1 of the same chromosome. The swap has the potential advantage of distributing terminals with high capacity requirements among different concentrators in order to enhance the feasibility of assigning terminals to concentrators during subsequent considerations of alleles in the chromosome. The process of swapping alleles within chromosomes of the initial population along with LC1 encoding is referred to as the *seeded LC1* strategy. The result of applying the genetic algorithm with the seeded LC1 strategy to the data in Figure 1 is shown in Figure 3. The total cost of assigning the ten terminals to the three concentrators is 276.

## 8. Another Least Cost Permutation Encoding (LC2)

An alternate least cost permutation encoding, LC2, is similar to LC1. Motivated by the observation that LC1 may place the terminals represented by the first C alleles far away and, consequently, at larger than necessary cost, the LC2 encoding strategy immediately begins a greedy decoding process with the first allele. Although LC2 seems more natural in terms of decoding an allele, LC2 is more likely to have infeasible chromosomes in the initial population pool as compared to the seeded LC1.

The following chromosome illustrates the LC2 permutation encoding for the best observed chromosome corresponding to the T = 10 and C = 3 configuration presented in Figure 1: 5 6 1 9 10 8 2 3 4 7
The initial capacities of $c_1$, $c_2$, and $c_3$ are 12, 12, and 12. One begins consideration of $t_5$, having a capacity

requirement of 1, and determines that each concentrator is still feasible for $t_5$. Since the cost from $t_5$ to concentrator $c_1$, $c_2$, and $c_3$ is 56, 18, and 107, respectively, one assigns $t_5$ to $c_2$. The remaining capacities of $c_1$, $c_2$, and $c_3$ then become 12, 11, and 12. One iterates the process by considering $t_6$. As before with LC1, the chromosome fitness using LC2 encoding is the total cost of the links that assign each terminal to the corresponding concentrator. The result of applying the genetic algorithm with LC2 encoding to the data in Figure 1 produces the same assignment as LC1 and is shown in Figure 3.

## 9. Results

Table 1 summarize the results of executing the *greedy algorithm with tradeoff* $\alpha$. The tables also summarize the results of executing the genetic algorithm for various choices of chromosome representation, crossover and mutation.

The results for the ten random data sets, each having 100 terminals, are indicated in Table 1. The *greedy algorithm with tradeoff* $\alpha$ frequently failed to find a feasible assignment of terminal sites to concentrator sites. For each of the ten data sets, the number of values of $\alpha$ for which the *greedy algorithm with tradeoff* $\alpha$ failed are recorded. The average failure rate is 5.8 out of 11 choices for $\alpha$ in the range from 0.0 to 0.1 with increment 0.1. For the second of the ten data sets the greedy algorithm failed for all choices of $\alpha$. For each of the nine other data sets, the minimum cost and the value of $\alpha$ producing the minimum cost is recorded. Similarly, for each combination of chromosome representation, crossover, and mutation, the least cost resulting from executing the genetic algorithm is recorded. In all cases, the genetic algorithm converged on a feasible assignment. The terminal assignment determined by the best genetic algorithm was lower in cost than the best assignment produced by the *greedy algorithm with tradeoff* $\alpha$ for each of the ten data sets. For the nine out of ten cases where the greedy algorithm found a solution, the mean percentage improvement of the best genetic algorithm solution over the best greedy algorithm solution was 10.4% and the standard deviation was 3.8%. Even though the LC2 encoding strategy, in comparison to LC1, caused the genetic algorithm to be slower in evolving a feasible pool of chromosomes, the results indicate that the LC2 encoding strategy was generally superior to the LC1 encoding strategy. There seems to be no significant difference between the PMX and the cycle crossover strategies. The adaptive mutation strategy produced better results that the fixed mutation strategy.

Similarly the results for the one random data set for each of 200, 300, and 400 terminals are indicated in Table 1. As with the 100-terminal data sets, the *greedy algorithm with tradeoff* $\alpha$ was executed for each of the eleven choices for $\alpha$ between 0 and 1. The results for the 100-terminal data sets motivated using the LC2 encoding, cycle crossover, and adaptive mutation for the 200, 300, and 400-terminal data sets. The results are similar and show significant improvement for the genetic algorithm over the greedy heuristic. The average improvement is 11.8% with a standard deviation of 2.72%. For comparison, the results of executing the genetic algorithm with seeded LC1 encoding, cycle crossover, and adaptive mutation are also included.

Using the terminal locations and the terminal sizes for case VI of the ten data sets of 100 terminals, the greedy algorithm found solutions with a total cost of 1462, 1713, and 2359 for W = 12, 24, and 48, respectively. The corresponding mean total cost for ten repetitions of the GA (using the LC2 chromosome representation with cycle crossover and adaptive mutation) was 1312, 1638, and 2286, representing improvements of 10.2%, 4.3%, and 3.1% with a standard deviation of 1.1%, 0.2%, and 0.3%, respectively.

The GA implementation of the TA problem outperformed the greedy algorithm in all of our test cases. It should be noted that the percentage improvement for the genetic algorithm over the greedy algorithm is quite significant.

An added benefit of the genetic approach is that, in almost every context, the genetic algorithm also generates many other feasible assignments that are "nearly as good." In the context of communications network design, another "nearly as good" solution may in fact be superior after other considerations that are not fully reflected in the model of the problem. A more deterministic algorithm might not reveal the other good feasible assignments.

## 10. Future work:

This research has demonstrated the applicability of genetic algorithm techniques to communications network design problems. There may be additional encodings and crossover strategies that further improve the assignment of terminals to concentrators while doing centralized network design. A graphical visualization of the terminal assignments would also be useful. Quite significantly for network planning, similar techniques can be used to solve other related and more difficult problems in distributed communications networks. The problems include concentrator location, network flow and capacity planning.

## Acknowledgements

This research has been supported by OCAST Grant AR2-004. The authors also wish to acknowledge the support of Sun Microsystems Inc.

## References

[1] Faris N. Abuali, D. A. Schoenefeld, R. L. Wainwright, "The Design of a Multipoint Line Topology for a Communications Network Using Genetic Algorithms", *Proceedings of the Seventh*

*Oklahoma Conference on Artificial Intelligence*, November, 1993

[2] J.L. Blanton and R.L. Wainwright, "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms", *Proceedings of the Fifth International Conference on Genetic Algorithms* (GA93), S. Forrest, ed., Morgan Kaufmann, San Mateo, CA.

[3] A.L. Corcoran and R.L. Wainwright, "A Genetic Algorithm for Packing in Three Dimensions", *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing, 1992*, pp. 1021-1030, ACM Press.

[4] A.L. Corcoran and R.L. Wainwright, "LibGA: A User-friendly Workbench for Order-based Genetic Algorithm Research", *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, pp. 111-118, 1993, ACM Press.

[5] L. Davis, ed., *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publisher, 1987.

[6] L. Davis, ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[7] K.A. De Jong and W.M. Spears, "Using Genetic Algorithms to Solve NP- Complete Problems", *Proceedings of the Third International Conference on Genetic Algorithms*, June, 1989, pp. 124-132.

[8] M.R. Garey and D.S. Johnson, Computers and Intractability: *A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[9] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[10] J. Grefenstette, GENESIS, Navy Center for Applied Research in Artificial Intelligence, Navy Research Lab., Wash. D.C. 20375-5000.

[11] D.R. Jones and M.A. Beltramo, "Solving Partitioning Problems with Genetic Algorithms", *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 442-449, Morgan Kaufmann, 1989.

[12] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill, 1993.

[13] P.P. Mutalik, L.R. Knight, J.L. Blanton and R.L. Wainwright, "Solving Combinatorial Optimization Problems Using Parallel Simulated Annealing and Parallel Genetic Algorithms", *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, pp. 1031-1038, 1992, ACM Press.

[14] G. Rawling, ed., *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, 1991.

[15] D. Whitley and J. Kauth, GENITOR: A Different Genetic Algorithm, *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, Co., 1988, pp. 118-130.

[16] D. Whitley and T. Starkweather, "GENITOR II: A Distributed Genetic Algorithm, *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1990) 189-214.

[17] D. Whitley, T. Starkweather, and D. Fuquat, "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator", *Proceedings of the Third International Conference on Genetic Algorithms*, June, 1989.

| label | weight | coordinates |
|---|---|---|
| 1 | 4 | (77,23) |
| 2 | 2 | (35,57) |
| 3 | 3 | (57,39) |
| 4 | 3 | (25,88) |
| 5 | 1 | (91,20) |
| 6 | 4 | (69,30) |
| 7 | 4 | (19,82) |
| 8 | 4 | (62,84) |
| 9 | 6 | (74,05) |
| 10 | 4 | (20,64) |

(a) Terminal capacity requirements (weight) and terminal coordinates.

| label | coordinates |
|---|---|
| 1 | (66,71) |
| 2 | (87,02) |
| 3 | (17,98) |

(b) Concentrator coordinates, capacity = 12.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 49 | 23 | 96 |
| 2 | 34 | 75 | 44 |
| 3 | 33 | 47 | 71 |
| 4 | 44 | 106 | 12 |
| 5 | 56 | 18 | 107 |
| 6 | 41 | 33 | 85 |
| 7 | 48 | 104 | 16 |
| 8 | 13 | 85 | 47 |
| 9 | 46 | 26 | 92 |
| 10 | 46 | 91 | 34 |

(c) Terminal to concentrator cost (distance) matrix.

Figure 1

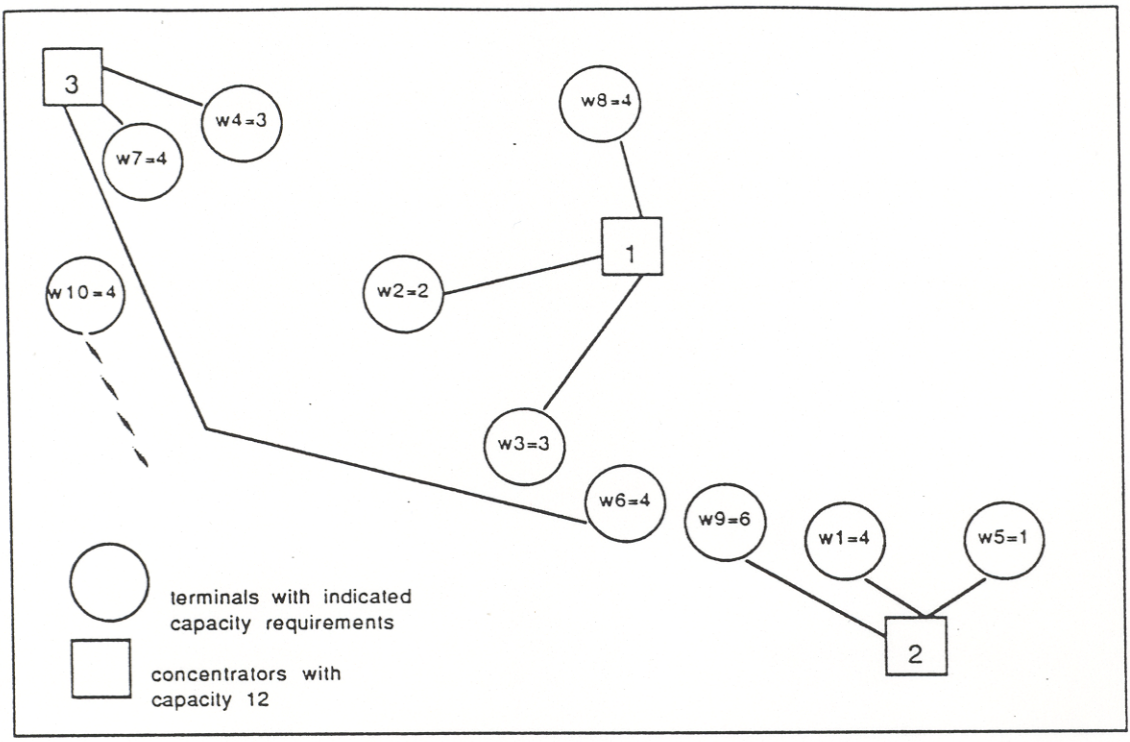


Figure 2. Terminal assignment to concentrators
Greedy algorithm with tradeoff $\alpha = 0$.
Total cost = 260 with terminal 10 stranded.

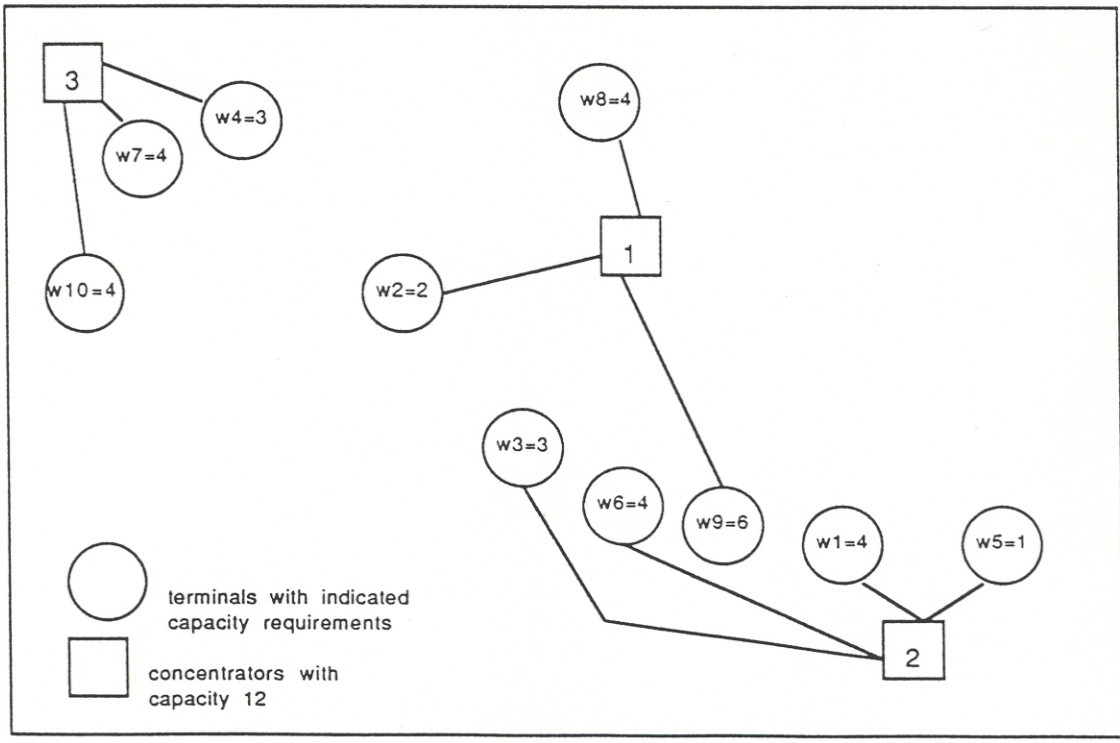


Figure 3. Terminal assignment to concentrators
Genetic algorithm with LC1 and LC2 encoding.
Total cost = 276 with all terminals assigned.

| Number of terminal sites: | | | 100 | | | | | | | | 200 | 300 | 400 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Trial Number: | | | I | II | III | IV | V | VI | VII | VIII | IX | X | | | |
| **Greedy Heuristic Algorithm:** | | | | | | | | | | | | | |
| Number of concentrator sites used: | | | 32 | 32 | 31 | 33 | 27 | 27 | 31 | 27 | 31 | 31 | 63 | 96 | 128 |
| Number of $\alpha$ s producing no solution: | | | 7 | 11 | 0 | 9 | 7 | 8 | 4 | 2 | 1 | 9 | 6 | 4 | 3 |
| Value of $\alpha$ producing best cost: | | | 0.8 | 0.0 | 0.0 | 0.9 | 0.7 | 0.8 | 1.0 | 0.7 | 0.1 | 0.9 | 0.8 | 0.3 | 0.5 |
| Best greedy cost: | | | 1161 | † --- | 1221 | 1485 | 1582 | 1462 | 1850 | 1929 | 1646 | 1265 | 1884 | 2552 | 3209 |
| **Genetic Algorithm:** | | | | | | | | | | | | | |
| chromosome representation | crossover | mutation | | | | | | | | | | | |
| seeded LC1 | pmx | fixed 0.1 | 1184 | 1257 | 1305 | 1436 | 1534 | 1428 | 1832 | 1795 | 1581 | 1321 | | | |
| seeded LC1 | pmx | adaptive | 1122 | 1148 | 1119 | 1327 | 1451 | *1309 | 1775 | 1695 | 1458 | 1195 | | | |
| seeded LC1 | cycle | fixed 0.1 | 1222 | 1259 | 1275 | 1405 | 1672 | 1434 | 1790 | 1759 | 1524 | 1296 | | | |
| seeded LC1 | cycle | adaptive | 1156 | 1168 | 1131 | 1351 | 1464 | 1348 | 1732 | *1615 | 1438 | 1266 | 1769 | 2369 | 3169 |
| LC2 | pmx | fixed 0.1 | 1120 | *1146 | 1111 | 1334 | 1459 | 1324 | 1762 | 1675 | 1402 | 1200 | | | |
| LC2 | pmx | adaptive | 1106 | 1157 | *1090 | 1328 | 1429 | 1321 | 1734 | 1630 | *1394 | *1182 | | | |
| LC2 | cycle | fixed 0.1 | 1134 | 1161 | 1144 | 1355 | 1453 | 1318 | 1813 | 1674 | 1460 | 1224 | | | |
| LC2 | cycle | adaptive | *1099 | 1162 | 1107 | *1303 | *1423 | 1334 | *1725 | 1624 | 1400 | 1193 | *1721 | *2213 | *2775 |
| % improvement: | | | 5.3% | ---- | 10.7% | 12.3% | 10.1% | 10.5% | 6.8% | 16.3% | 15.3% | 6.6% | 8.7% | 13.3% | 13.5% |

**Table 1.**

The concentrator capacity is W = 12.

The terminal site requirement is random in 1-6.

* best cost. † no feasible solution found.