A Multiobjective Hybrid Genetic Algorithm for the Capacitated Multipoint Network Design Problem

Chi-Chun Lo and Wei-Hsin Chang
Institute of Information Management
National Chiao-Tung University
1001 Ta Hsueh Road
Hsinchu, Taiwan 300, ROC
Phone:886-3-5712121#57409
email: cclo@cc.nctu.edu.tw

Abstract

The Capacitated Multipoint Network Design Problem (CMNDP) is NP-complete. In this paper, a hybrid genetic algorithm for CMNDP is proposed. The MultiObjective Hybrid Genetic Algorithm (MOHGA) differs from other genetic algorithms mainly in its selection procedure. The concept of subpopulation is used in MOHGA. Four subpopulations are generated according to the elitism reservation strategy, the shifting Prüfer vector, the stochastic universal sampling, and the complete random method, respectively. The next generation population is produced by mixing these four subpopulations. The MOHGA can effectively search the feasible solution space due to population diversity. The MOHGA has been applied to CMNDP. By examining computational results, we notice that the MOHGA can find most non-dominated solutions and is much more effective and efficient than other multiobjective genetic algorithms.

Keyword : Genetic Algorithm, Multiobjective Function, Minimal Spanning Tree, Non-Dominated Solution

I. INTRODUCTION

The problem of effectively transmitting data in a network involves the design of the communication subnetworks. A critical issue in network design is to find a set of links which connect communication nodes in a way that the weighted sum of the shortest paths between pairs of nodes is minimized, and the constraints of network capacity, delay time, and reliability are met. In the real world, network design has long been recognized as multiobjective in nature. For a centralized multipoint network, i.e., a tree network, the network design problem gives rise to a well-known combinatorial optimization problem, namely the constrained minimal spanning tree (CMST) problem. The CMST is NP-complete. Many heuristics, e.g., [3], [4], [9], [10], [12], [14], [24], have been proposed. However, their works took account only cost or delay. In recent years, genetic algorithm (GA) has been applied to various multiobjective optimization problems. Schaffer [25] proposed a vector evaluated genetic algorithm (VEGA) to solve multiobjective optimization problems. In VEGA, a population is divided into n disjoint subpopulations. For each subpopulation, different objective function is used to evaluate the fitness of chromosomes (solutions). In this paper, the capacitated multipoint network design problem (CMNDP) is considered. A multiobjective hybrid genetic algorithm (MOHGA) is proposed for CMNDP. The MOHGA differs form other genetic algorithms mainly in its selection procedure. The concept of subpopulation is used in MOHGA. Four subpopulations are generated according to the elitism reservation strategy, the shifting Prüfer vector, the stochastic universal sampling, and the complete random method, respectively. The next generation population is produced by mixing these four subpopulations. The MOHGA can effectively search the feasible solution space due to population diversity. By applying MOHGA to CMNDP, we notice that the MOHGA can find most non-dominated solutions in the feasible solution space.

In the next section, a brief introduction of genetic algorithm is given. Section III describes the MOHGA. The problem formulation of CMNDP is detailed in Section IV. In Section V, computational experiments are presented. Section VI concludes this paper with possible future directions.

II. GENETIC ALGORITHM

The concept of genetic algorithms, introduced by John Holland [19], is based on the mechanics of natural selections and natural genetics. Genetic algorithms start with an initial set of random solutions called **population**. Each individual in the population is called a **chromosome**, representing a solution to the problem. The initial population evolves through successive iterations, called **generations**. A measure of fitness defines the quality of an individual chromosome. In each generation, chromosomes are evaluated by using fitness function, also called evaluation function. After a number of generations, highly fitting individuals, which are analogous to good solutions to a given problem, will emerge. Because of this property, the GA is robust than existing direct search methods, such as hill climbing method, etc [22].

A genetic algorithm consists of five components, as described in Davis's book [6]. These five components are as follows:

(1). a method for encoding potential solutions into chromosomes,
(2). a means of creating the initial population.
(3). an evaluation function that can evaluate the fitness of chromosomes.
(4). genetic operators that can create the next generation population.
(5). a way to set up control parameters, e.g., population size, probability of applying genetic operator, etc.

III. THE MULTIOBJECTIVE HYBRID GENETIC ALGORITHM

For a multiobjective optimization problem, the GA can be used to find its non-dominated solutions. To find good solutions via GA, the population has to be very diverse. The concept of subpopulation proposed by Schaffer [25] is a promising approach. Mixing subpopulations not only keeps population diversity but also prevents the population from converging to its local optimum due to the dominance of the "super" chromosome.

Four subpopulations are generated according to the elitism reservation strategy [27], the shifting Prüfer vector, the stochastic universal sampling [2], and the complete random method, respectively. Mixing these four subpopulations creates the next generation population.

A. Subpopulation

1) Elitism reservation strategy

In traditional GAs, a chromosome in the current population is selected into the next generation with certain probability. The best chromosomes of the current generation may be lost due to mutation, crossover, and selection during the evolving process, and subsequently causes difficulty in reaching convergence. In other word, it takes more generations, i.e., running time, to get the quality solutions. Tamaki [27] proposed an elitism reservation strategy, which permits chromosomes with the best fitness to survive and be carried into the next generation.

2) Shifting Prüfer Vector

The shifting Prüfer vector is a new idea originated in this paper. The well-known problem of Prüfer encoding [24] is that it does not preserve locality. Changing one element of a Prüfer vector can change its corresponding tree topology dramatically. To remedy this problem, we suggest a new genetic operator, called the shifting Prüfer vector. This operator maintains maximum locality; i.e., this operator keeps the similarity between chromosomes. The concept of the shifting Prüfer vector is as follows: it replaces the leftmost element of a Prüfer vector by a randomly selected non-leftmost element of the vector. The new vector differs from the old one only in the leftmost element. Thus, the new topology differs from the old one in at most two edges. In most cases, the difference is only one edge. The shifting Prüfer vector is a local search method. According to the results obtained by the well-known Add and Drop searching heuristic [29][30][31], changing only one element in every iteration of the search process can always obtain a good solution. Thus, the shifting Prüfer vector can significantly improves the quality of newly found chromosomes.

Figures 1(a) and 1(b) are used to show the effect of the shifting Prüfer vector. Figure 1(a) depicts a seven-node tree with its Prüfer encoding. Figure 1(b) illustrates the new tree after the shifting Prüfer vector is applied. We notice that the new tree and the old one differ in only one edge.

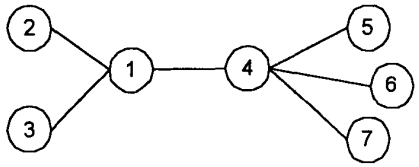

Figure 1 (a) a 7-node tree and its Prüfer encoding =[1,1,4,4,4]

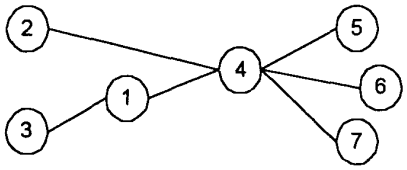

Figure 1 (b) the new tree after the shifting Prüfer vector and its Prüfer encoding =[4,1,4,4,4]

3) Stochastic universal sampling

A simple way to perform sampling is to spin a roulette wheel. Unfortunately, this sampling method does not guarantee that any particular string will actually be chosen in any given generation. This sampling error is a well-known problem in the roulette wheel selection method. Baker suggested the stochastic universal sampling method [2]. Baker's algorithm completes the whole sampling in a single pass, and requires only one random number. A wheel spin, whose size is equal to the population size, is divided into a number of equally spaced markers. A single spin is used to generate the random number. The expected value e_k for chromosome k is calculated as e_k=pop_size*p_k, where pop_size represents population size, p_k represents selection probability. The basic concept of this approach is to maintain the expected number of keeping a copy of every chromosome of the current generation in the next generation.

4) Complete random method

Population is generated according to random number and random position. The major reason for using the complete random method is to keep the diversity of the population.

B. The Multiobjective Hybrid Genetic Algorithm

MOHGA

Step 1. Set the maximum number of generation, t_{max}, and initialize the loop counter, **t**, to zero.
Step 2. Encode the problem into chromosomes by Prüfer encoding.
Step 3. Produce the initial population, **P(t)**, by the complete random method.
Step 4. Evaluate P(t); exit, if the solutions are found.
Step 5. Generate the subpopulation, $SP_1(t)$, by the elitism reserved strategy.
Step 6. Generate the subpopulation, $SP_2(t)$, by the shifting Prüfer vector.
Step 7. Generate the subpopulation, $SP_3(t)$, by the stochastic universal sampling with probability, p_c, and probability, p_m.
Step 8. Generate the subpopulation, $SP_4(t)$, by the complete random method.
Step 9. Form the next generation, P(t), by mixing $SP_1(t)$,$SP_2(t)$, $SP_3(t)$, $SP_4(t)$.
Step 10. Increase **t** by 1; if **t** is less than t_{max}, then goto Step 4; otherwise, exit.

where $\mathbf{p_c}$ represents the probability of crossover, $\mathbf{p_m}$ the probability of mutation, **P(t)** the population of the t-th generation, and $\mathbf{SP_i(t)}$ the i-th subpopulation of the t-th generation.

IV. CAPACITATED MULTIPOINT NETWORK DESIGN PROBLEM

A. Problem Formulation

The CMNDP can be formulated as follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

$$\min \quad \sum_{\forall(i,j)\in E} d_{ij}x_{ij} \tag{2}$$

subject to :

$$\sum_{(i,j)\in T_k} w_i x_{ij} < W \qquad \forall k \tag{3}$$

$$\sum_{(i,j)\in S} x_{ij} < |S| \qquad \forall S \tag{4}$$

$$\sum_{i,j} x_{ij} = N-1 \tag{5}$$

$$x_{ij} \in 0,1 \qquad \forall (i,j) \in E. \tag{6}$$

where
- N represents the set of nodes in the network,
- E the set of links in the network,
- Tk the *k*-th link; it may not exist for some k,
- W the given weight matrix,
- w_i the weight of the *i*-th link,
- S the possible spanning tree; |S| represents the number of edges of S,
- $c_{i,j}$ the cost of connecting node i to node j; i.e., link(i,j); the cost matrix($c_{i,j}$) is symmetric,
- d_{ij} the average delay on link(i,j); the delay matrix ($d_{i,j}$) is symmetric,
- x_{ij} the 0/1 decision variable, 1, if link(i,j) is selected; 0, otherwise.

(3) guarantees that the total link weight does not exceed the upper limit. (4) guarantees that the set of chosen links does not form any cycle. (5) guarantees that enough links will be chosen to connect the network.

B. Applying MOHGA to CMNDP

1) Encoding method

Prüfer encoding provides a one-to-one mapping between the set of spanning trees and the set of sequences of n-integers [13][24]. Because of this excellent property, Prüfer encoding is chosen for encoding chromosomes.

2) Initial population

Each individual in the initial population is a solution to the problem. We use the complete random method to generate the initial population.

3) Evaluation Function

Link cost and transmission delay are both chosen to be the evaluation functions for CMNDP.

V. COMPUTATIONAL EXPERIMENTS

A. Test Problems and Results

In order to evaluate the solutions of CMNDP obtained by MOHGA, we examine a set of problems, with 7,14, 28 and 56 nodes, respectively. The simulator is coded in the C++ language and is running on an Intel Pentium-166MHz PC with 64MB RAM.

The number of non-dominated solutions obtained by MOHGA is affected by control parameters; e.g., mutation probability, crossover probability, and maximum number of generation. For population diversity, mutation probability is set to be greater than 0.8. The greater the mutation probability, the more the diversity of the population. Crossover probability ranges form 0.4 to 0.6. The maximum number of generation allowed is dependent on the problem size. The lager the problem size, the larger the feasible solution space. In other words, more generations are needed to find the solutions.

1) Problem 1

For the 7-node network, nodes are randomly distributed. Control parameters are given as follows: the mutation probability is set to 0.9, the crossover probability is set to 0.5, the size of population is set to 100, and the maximum number of generation allowed is set to 200.

All the non-dominated solutions, i.e., all possible spanning trees, are enumerated. By doing this, we are able to compare the quality of the solutions obtained by MOHGA directly with the enumerated solutions. A (total cost, total delay) pair represents a solution. By enumerating the solutions, we found

all six non-dominated solutions of this problem, which are (13,92), (14,91), (15,85), (16,84), (18,78), and (19,77). With 10 runs, non-dominated solutions found by MOHGA can be summarized as follows:
Set 1: { (13,92), (14,91), (15,85), (16,84), (18,78), (19,77) };
Set 2: { (13,92), (15,85), (16,84), (18,78), (19,77) };
Set 3: { (14,91), (15,85), (16,84), (18,78), (19,77) }.
Sets 1, 2, and 3 are obtained 4, 4, and 2 times, respectively. Therefore, the MOHGA finds 90 percents of all non-dominated solutions.

For comparison, the vector evaluated genetic algorithm (VEGA)[25] and the single-objective genetic algorithm (SOGA)[27] are applied to the same test problem. In SOGA, weights w1(0.5) and w2 (0.5) are used in the fitness function. Other control parameters are the same as those of MOHGA. Computational results are summarized in Table I.

Table I.
CPU time and Number of solutions of the 7-node problem w.r.t. MOHGA, SOGA, and VEGA

Algorithm	CPU times (Sec)	Average number of non-dominated solution obtained by algorithm (A)	Ratio (A/ B)
MOHGA	2.278	5.4	0.9
SOGA	2.16	2	0.333
VEGA	2.365	1.5	0.25

where A : The number of non-dominated solutions obtained by the algorithm.
B : The number of all non-dominated solutions, which is 6 in this problem.

2) Problem 2
A network of 14 nodes is considered in Problem 2. Network nodes are randomly distributed. Weight, cost, and delay are randomly generated. Control parameters are given as follows: the mutation probability is set to 0.9, the crossover probability is set to 0.4, the size of population is set to 100, and the maximum number of generation allowed is set to 1000. Computational results are presented in Table II.

Table II.
CPU time and Number of solutions of the 14-node problem w.r.t. MOHGA, SOGA, and VEGA

Algorithm	CPU times (Sec)	Average number of non-dominated solutions obtained by algorithm (A)
MOHGA	37.90	9.5
SOGA	28.83	2.5
VEGA	45.04	2.5

3) Problem 3
A network of 28 nodes is considered in Problem 3. Network nodes are randomly distributed. Weight, cost, and delay are randomly generated. Control parameters are given as follows: the mutation probability is 0.99, the crossover probability is 0.4, the size of population is set to 100, and the maximum number of generation allowed is set to 1000. Computational results are shown in Table III.

Table III.
CPU time and Number of solutions of the 28-node problem w.r.t. MOHGA, SOGA, and VEGA

Algorithm	CPU times (Sec)	Average number of non-dominated solutions obtained by algorithm (A)
MOHGA	56.792	13
SOGA	76.192	1.5
VEGA	83.4	1.5

4) Problem 4
A network of 56 nodes is considered in Problem 4. Network nodes are randomly distributed. Weight, cost, and delay are randomly generated. Control parameters are given as follows: the mutation probability is set to 0.99, the crossover probability is set to 0.4, the size of population is set to 100, and the maximum number of generation allowed is set to 10000. Computation results are listed in Table IV.

Table IV.
CPU time and Number of solutions of the 56-node problem w.r.t. MOHGA, SOGA, and VEGA

Algorithm	CPU times (Sec)	Average number of non-dominated solutions obtained by algorithm (A)
MOHGA	2167.58	15
SOGA	3780.462	1.3
VEGA	4138.066	1.3

5) Problem 5
A network of 56 nodes is considered in Problem 5. Network nodes are not randomly distributed. Most nodes reside at the upper left corner of the network. Weight, cost, and delay are randomly generated. Control parameters are given as follows: the mutation probability is set to 0.99, the crossover probability is set to 0.4, the size of population is set to 100, and the maximum number of generation allowed is set to 10000. Computation results are reported in Table V.

Table V.
CPU time and Number of solutions of the 56-node problem w.r.t. MOHGA, SOGA, and VEGA

Algorithm	CPU times (Sec)	Average number of non-dominated solutions obtained by algorithm (A)
MOHGA	2181.68	13
SOGA	3793.26	1.3
VEGA	4147.34	1.3

B. Analyses

1) Running time (complexity) analysis
Tables 1 through 5 detail the average CPU time used by MOHGA, SOGA, and VEGA. For small size problems; e.g., Problems 1 and 2, the MOHGA takes more time than the SOGA, but less time than the VEGA does. For large size problems; e.g., Problems 3, 4 and 5, the MOHGA takes the least time to generate non-dominated solutions.
The comparison of the running time of MOHGA, SOGA, and VEGA is shown in Figure 2. Figure 2 illustrates that the rate of increase of the running time of SOGA, and VEGA, is close to an exponential function. For MOHGA, the rate of increase of the running time is only 57.6 percents of that of SOGA, and VEGA. By examining computational results, we have the following observations: the SOGA takes the least time for small size problems, but obtains the worst results; the VEGA takes the most computing time; and the MOHGA is much more efficient than the SOGA and the VEGA.

2) Quality analysis
For Problem 1, the MOHGA, which finds 90 percents of all non-dominated solutions, performs much better than the SOGA and the VEGA. The same phenomena can be found for Problems 2 through 5, as illustrated in Tables 2,3,4, and 5. The MOHGA always finds more non-dominated solutions than the SOGA and the VEGA. Therefore, the MOHGA is much more effective than the SOGA and the VEGA.

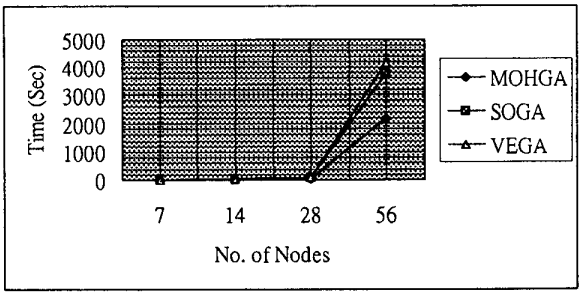

Figure 2. The running time (complexity) analysis

IV.CONCLUSIONS

A. Summary
The MOHGA incorporates the subpopulation concept. The elitism reservation strategy, the shifting Prüfer vector, the stochastic universal sampling, and the complete random method are used to produce the next generation population. The MOHGA has been applied to CMNDP. By examining computational results, we notice that the MOHGA finds most non-dominated solutions and is much more effective and efficient than the SOGA

and the VEGA.

B. Future Directions

Here, we would like to mention the following areas, which may merit further investigation.

(1) Compare the MOHGA with Niched Pareto's GA[20].
(2) Apply the fuzzy concept on cost and delay; a fuzzy weighted edge in a tree is a hard problem in general.
(3) Apply the MOHGA to other multiobjective optimization problems.

References

[1]Abuali, F. N.,Wainwright, R. L., and Schoenefeld D. A., "Determinant Factorization : A New Encoding Scheme for Spanning Tree Applied to the Probability Minimum Spanning Tree Problem", Proc. of 6th ICGA, 1995, pp.470-477.

[2]Baker, J., "Adaptive selection methods for genetic algorithms", Proc. of 2nd ICGA, 1987, pp.100-111.

[3]Boorstyn, R. R. and Frank, H., "Large Scale Network Topological Optimization", IEEE Trans. on Comm., COM-25(1), pp29-47, 1977.

[4]Clarke, L.W. and Anandalingam, G. "An Integrated System for Designing Minimum Cost Survivable Telecommunications Networks", IEEE Trans. on System, Man, and Cybernetics, Part A: System and Humans, vol. 26, no. 6, Nov. pp. 856-862, 1996.

[5]Cox, L. A., Davis, L. and Qiu, Y., "Dynamic Anticipatory Routing in Circuit-Switched Telecommunication Networks.", Handbook of Genetic Algorithms, Davis, L. (ed.), Van Nostrand Reinhold, 1991, pp. 124-143.

[6]Davis, L., Genetic Algorithms and Simulated Annealing. Research Notes in Artificial Intelligence, Morgan kaufmann, Los Altos, CA., 1987.

[7]Davis, L., "A Genetic Algorithm for Survivable Network Design", Proc. of 5th ICGA, pp.408-415, 1993.

[8]De Jong, K. A., "An Analysis of the Behavior of a Class of Genetic Adaptive systems", Doctoral dissertation, University of Michigan, 1975.

[9]Elbaum, R. and Sidi M., "Topological Design of Local-Area Networks Using Genetic Algorithms", IEEE/ACM Trans. on Networking, vol. 4, no. 5, pp766-778, 1996.

[10]Ersoy, C. and Panwar, S.S., "Topological Design of Interconnected LAN/WAN Networks", IEEE JSAC. vol. 11, no. 8, Oct. 1993.

[11]Fonseca, C. M. and Fleming. P. J., "Genetic Algorithms for multiobjective optimization: formulation, discussion and generalization", Proc. of 5th ICGA, pp.416-423, 1993.

[12]Gavish, B., "Topological Design of Telecommunication Networks-Local Access Design Methods", Annals of Operations Research 33, pp17-71, 1991.

[13]Gen, M. and Cheng, R., Genetic Algorithms & Engineering Design, John Wiley & Sons, Inc, 1997.

[14]Gerla, M. and Kleinrock, L., "On the Topological Design of Distributed Computer Networks", IEEE Trans. on Comm., COM-25(1), pp. 48-60, 1977.

[15]Gibbons, A., Algorithmic Graph Theory, Cambridge University, New York, 1985.

[16]Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA., 1989.

[17]Grenfenstette, J.J.,"Optimized Control of Parameters for Genetic Algorithms", IEEE Trans. on Systems, Man, and Cybernetics 16, pp. 122-128, 1986.

[18]Grefentstte, J.J., "Incorporation Problem Specific Knowledge into Genetic Algorithms", in: L.Davis (ed.), Genetic Algorithms and Simulated Annealing, Los Altos, CA, 1987.

[19]Holland, J. H., "Adaptation in Natural and Artificial Systems", University of Michgan Press, Ann Arbor, MI; reprint by MIT press, Cambridge, MA (1992)

[20]Horn, J., Nafpliotis, N. and Goldberg, D.E., "A Niched Pareto Genetic Algorithm for multiobjective optimization", Proc. of 1st ICEC, pp.82-87, 1994.

[21]Hu, T. C., "Optimum Communication Spanning Trees", SIAM J. Comput. (3) pp 188-195, 1994.

[22]Michalewicz, Z., Genetic Algorithms + Data Structure = Evolution Programs, Springer-verlag Berlin, 1996.

[23]Murata, T. and Ishibuchi, H., " MOGA: Multi-Objective Genetic Algorithms", Proc. of 2nd ICEC, pp.289-294, 1995.

[24]Palmer, C. C. and Kershenbaum, A., "An Approach to a Problem in Network Design Using Genetic Algorithms", Networks, vol. 26, pp151-163, 1995.

[25]Schaffer, J. D., "Multiple objective optimization with Vector Evaluated Genetic Algorithms", Proc. of 1st ICGA, pp.93-100, 1985.

[26]Schaffer,J.D., Carunam R.A., Eshelman, L.J., and Das, R., "A Study of Control Paramters Affecting Online Performance of Genetic Algorithms for Function Optimization", in: J.D. Schaffer(ed), Proc. of 3rd ICGA, Morgan Kaufmann, New York, pp. 51-60, 1989.

[27]Tamaki, H., Kita, H, and Kobayahi, S., "Multi-Objective Optimization by Genetic Algoritms : A Review", Proc. of 3rd ICEC, pp.517-522, 1996.

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

[29]Kershenbaum, A., Telecommunication Network Design Algorithms. McGraw-Hill, pp. 225-236, 1993.

[30]Prim, R. C., "Shortest connection networks and some generallizations", Bell System Technical Jurnal, vol. 36, pp. 1389-1401, 1957.

[31]Moret, B. M. E. and Shapiro, H. D., Algorithms from P to NP : Design and Efficiency, vol. I, Benjamin/Cummings, pp. 254-292, 1991.