◈ Research Paper

Evolutionary Algorithms for Finding the k Most Vital Arcs in Minimum Spanning Tree Problem[1)]

Ho Yeon Chung*

Abstract

The purpose of this study is to present methods for determining the k most vital arcs (k-MVAs) in the minimum spanning tree problem(MSTP) using evolutionary algorithms. The problem of finding the k-MVAs in MSTP is to find a set of k arcs whose simultaneous removal from the network causes the greatest increase in the total length of minimum spanning tree. Generally, the problem which determine the k-MVAs in MSTP has known as NP-hard. Therefore, in order to deal with the problem of real world the heuristic algorithms are needed. In this study we propose to three genetic algorithms as the heuristic methods for finding the k-MVAs in MSTP. The algorithms to be presented in this study are developed using the library of the evolutionary algorithm framework(EAF) and the performance of the algorithms are analyzed through the computer experiment.

1. Introduction

Minimum spanning tree problem(MSTP) is the problem which finds a spanning tree that has the smallest total length of its constituent arcs, measured as the sum of costs of the arcs in the spanning tree[1]. The optimum solutions for this problem can be found by using the Kruskal's, Prim's, and Sollin's algorithm. Finding the optimum solution with these algorithms is important, however, in the position of a network administrator, it is also important to understand how the arcs affect the performance of the total network when some arcs cause problems in a network. The best known application for this problem is in a conflict situation where a logistics or communications network are under attacks[8,9]. In this situation, the user wishes to know which arcs are the most vital to him so that he can reinforce these arcs against attack, while his enemy wants to destroy those arcs which most increase the total length of the minimum spanning tree[5]. The problem of finding the k-MVAs in MSTP is to find a set of k arcs whose simultaneous removal from the network causes the greatest increase in the total length of the spanning tree. In the past, research on this problem has been conducted by Lin & Chern, H.Shen, and Hsu. Lin & Chern[6] showed that the problem of finding the k-MVAs in MSTP is to be NP-hard and suggested an exact algorithm using the branch and bound method. H. Shen[4] also proposed a randomized algorithm based on the approach of an arc replacement method. However, there exists some problems in applying these algorithms to the real

1) This work was supported by a Korea Science and Engineering Foundation Grant No. 98-0200-09-01-3

* Department of Industrial Engineering, Jeonju University

world. Although there may be an optimum algorithms for k-MVAs problem, it has limits in computer memory and computation time since k-MVAs problem has been proven NP-hard. Therefore, it is more practical to find a heuristic solution rather than an optimum solution for k-MVAs problem.

In this study, we developed three kinds of evolutionary algorithms for finding the k-MVAs in MSTP within a short time, and through computer experiments, we also evaluated the performance of these algorithms applied to 20 problems generated from a standard problem generator of DIMACS.

2. Evolutionary Algorithms for Finding the k-MVAs

2.1 Express of individuals

In order to apply genetic algorithms, a method of expressing individuals reflecting the characteristics of the problem is required. A binary expression as a traditional method has been widely used[11]. However, it is difficult to express the solution of the network problem treated in this study with the binary expression. Although it is possible to use binary expression, it is not appropriate for the genetic operator to maintain the feasible solution in the form of binary expression[1]. In this study, instead of the binary expression, we consider two kinds of methods for expressing arcs of network. One is to express arcs using nodes and the other is to express arcs by giving serial numbers to arcs. The former is easy to express and to analyze arcs but it may cause infeasible arcs so that it may require an additional repairing method. On the contrary, the latter does not cause infeasible arcs but it can cause repeated arcs. Therefore, if we can just resolving the problem of repeated arcs, the latter is much easier than the former to be applied. In this study we use the latter method. Using the latter expression method, an individual can be expressed with arc factors consisting of arcs and then a potential solution for k-MVAs can be expressed.

2.2 Fitness evaluation and calculation

Each individual defined in this study consists of arcs set which is considered as k-MVAs. The fitness value of this individual is calculated by the difference between the length of the minimum spanning tree after removing k arcs and the length of the original minimum spanning tree. Therefore we can tell that the greater fitness value is the more vital.

2.3 Selection methods

For the selection methods, the elitism proposed by Goldberg[3] is used to maintain the elite individuals of the present population. For this purpose, after arranging the fitness value by descending order for each individuals of the generation $P(t)$, the best individual is copied. Then this best one is transmitted to next generation $P(t+1)$ and then the rest individuals are generated from $P(t)$ using the tournament selection method. In this method, the most elite individual can be protected in each population as generation is repeated and the better individual than that of the previous generation can be found by the tournament selection method.

2.4 Genetic operators

Genetic operators are composed of crossover and mutation. In this study, order crossover is used to prevent folding which can occur when crossover and mutation are performed. The order crossover is a method of inheriting some partial factors from one parent and the other partial factors from another parent according to their relative order. We modified the concept of inheriting factors based on their relative order with the concept of directly inheriting the entire factors of the parent in the next generation only when overlap occur. Figure 1 shows an example of a crossover when k = 2. In Figure 1, offsprings, o_1 and o_2, are generated by transmitting one factor from each parent, p_1 and p_2, based on one-point cut method. When parents, p_1 and p_2, are given like in Figure 2, their offspring o_1 has folding factors, that is, like $o_1 = \{m_i \ \ m_i\}$ In this case, all factors of the parent p_1 are directly transmitted to its offspring o_1 instead of succeeding m_i from the parent p_2. As it is done like this, the folding factor get removed.

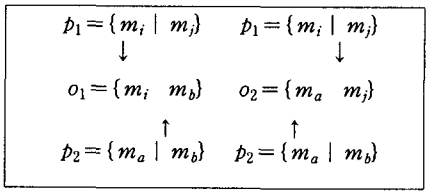

Figure 1. Crossover(no redundancy)

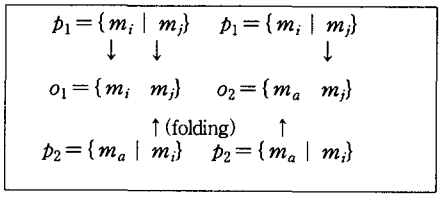

Figure 2. Crossover (redundancy)

Mutation, as a step of generating entirely new genes, takes on a role of widely searching the solution space. In this study, mutation is performed based on each factor. When overlap occurs, we generate another factor and then mutate the factor into a factor that cannot incur overlap.

2.5 Termination condition

For termination, if the number of newly generated individuals is more than 1000, then the experiment is terminated because it is impossible to improve the solution in that case.

Based on the above contents, genetic algorithms for determining k-MVAs in MSTP can be summarized as follows.

Simple Genetic Algorithm (SGA) for Determining the k-MVAs

Step 1 (Generating an initial population) Generating an initial population $P(t)$ using a random generation method.

Step 2 (Fitness evaluation) Evaluate the fitness value of all individuals in $P(t)$.

Step 3 (Selection) Select $P(t+1)$ from $P(t)$. For this purpose, arrange the fitness values of $P(t)$ by descending order, and then compare the best individual in $P(t)$ with the best individual in $P(t-1)$ After comparing, inherit the individual with the greatest

value into $P(t+1)$ Using a tournament selection method, the other individuals are generated one by one.

Step 4 (Crossover) Performing the order crossover for the two individuals that are randomly selected from the present population, generate offsprings.

Step 5 (Mutation) According to the mutation rate, select a random individual from the population. Select a random factor from the selected individual, and then mutate it another factor that is not overlapped.

Step 6 (Test of termination condition) When the termination condition is satisfied, the experiment is terminated. Otherwise, go to **Step 2** after setting $t = t + 1$.

Steady State Genetic Algorithm (SSGA) for Finding the *k*-MVAs

The standard genetic algorithm evolves the entire population by generation. On the contrary, SSGA regenerates only a few individuals (usually, one or two individuals) repeatedly and replaces them with individuals having high fitness values. The merit of this algorithm is to maintain the best individuals by replacing individuals having low fitness values. Also, once a good individual is generated, this individual can participate in the regeneration process. The steps for SSGA are the following:

Step 1 (Initial population) Set $t \leftarrow 0$ and then generate an initial population $P(t)$.

Step 2 (Fitness evaluation) Evaluate the fitness of all individuals in $P(t)$.

Step 3 (Selection) Set $t \leftarrow t+1$ and then select a few individuals from $P(t-1)$ Let a set of selected individuals be $sub\text{-}P(t)$.

Step 4 (Crossover and mutation) Perform crossover and mutation for $sub\text{-}P(t)$.

Step 5 (Replacement) According to the replacement strategy, select individuals from $P(t)$, and then replace these individuals with individuals in $sub\text{-}P(t)$.

Step 6 (Fitness evaluation) Evaluate the fitness of $sub\text{-}P(t)$ that are newly generated offsprings.

Step 7 (Termination condition) When the termination condition is satisfied, the experiment is ended. Otherwise, go to **Step 3**.

Ecosystem Genetic Algorithm (ECOGA) for Finding the *k*-MVAs

In the ECOGA, individuals evolve by comparing themselves with other individuals in a limited space, that is, neighbor. Initial population consists in the form of a torus with a two dimensional a lattice-shaped structure and a neighbor in the form of an $r \times r$ is used. Individuals in the neighbor evolve based on the SSGA. A neighbor N_{ij} has a square structure centering on an individual in the position of (i, j). The ECOGA prevents the population from converging into a partial optimum due to the super elite individual. Also, it can search various solutions. The steps for the ECOGA are the following.

Step 1 (Initial population) Set $t \leftarrow 0$ and then generate an initial population $P(t)$.

Step 2 (Fitness evaluation) Evaluate the fitness of all individuals in $P(t)$.

Step 3 (Selecting Neighbor) Select a random position (i, j) and define its neighbor N_{ij}.

Step 4 (Evolvement)

(4.1) Based on the fitness value of N_{ij}, select two individuals according to the value of probability.

(4.2) Perform genetic operation (crossover and mutation) and then generate two offsprings.

Step 5 (Replacement) According to the replacement strategy, replace two offspring with two individuals in N_{ij}.

Step 6 (Fitness evaluation) Evaluate the fitness of newly generated offspring.

Step 7 (Termination condition) When the termination condition is satisfied, the experiment is ended. Otherwise, go to **Step 3.**

3. Experiments and Analysis

The experiments for determining the k-MVAs in MSTP were performed for the 20 problems generated from the standard problem generator of DIMACS. The proposed genetic algorithms are programmed using Visual C++ language and performed on an IBM PC with 128M RAM and a Pentium-III CPU (650MHz).

3.1 Preliminary experiment to determine the value of genetic parameters

To find a good solution by the proposed genetic algorithms, we performed a preliminary experiment for the spagrid3 problem generated from DIMACS. Table 1 shows the plan of the preliminary experiments to find the optimum combination of a crossover and mutation rate. The preliminary experiment is performed 20 times for each combination. In this experiment, the interaction between parameters is neglected and the optimum combination of genetic parameters is determined based on the average value of the best fitness value in each experiment. If the number of newly generated individuals is more than 1000, the experiment is terminated since there is no possibility of improving the solution in that case.

Table 1. Preliminary experiment plan

Parameter \ Algorithm	SGA	SSGA	ECOGA
Population size	100/200/300/400/500	100/200/300/400/500	-
Selection method	Tournament method	Tournament method	Tournament method
Crossover rate	0.5/0.6/0.7/0.8/0.9	0.5/0.6/0.7/0.8/0.9	0.5/0.6/0.7/0.8/0.9
Mutation rate	0.1/0.2/0.3	0.1/0.2/0.3	0.1/0.2/0.3
Newly generated individuals	1000	1000	1000
Replacement rate	-	0.1/0.15/0.2	-
ECO. size	-	-	10/15/20/25/30
Neighboring population size	-	-	3

The experimental results are shown in Table 2.

Table 2. Results of preliminary experiment(in case of 2-MVAs)

(Crossover rate, Mutation rate)	SGA		SSGA (0.1)		SSGA (0.15)		SSGA (0.2)		EcoGA	
	Best fitness	Average fitness	Best fitness	Average fitness	Best fitness	Average fitness	Best fitness	Average fitness	Best fitness	Average fitness
(0.5,0.1)	1639	837	1007	515	935	627	1007	589	1199	763
(0.5,0.2)	1268	673	1058	561	1458	617	1012	454	1793	605
(0.5,0.3)	1472	500	1041	590	1815	567	1494	611	1071	457
(0.6,0.1)	1639	882	1007	482	1007	555	1409	593	1992	766
(0.6,0.2)	1494	724	1028	569	1460	508	1148	570	1639	639
(0.6,0.3)	1992	664	1532	634	1073	589	1074	595	1367	606
(0.7,0.1)	1586	706	1608	709	1061	736	991	529	1078	570
(0.7,0.2)	1793	749	1483	477	960	428	1095	498	1191	544
(0.7,0.3)	1494	744	1082	616	1042	458	1838	689	1355	796
(0.8,0.1)	1659	843	1048	600	1007	525	1458	642	1253	620
(0.8,0.2)	1432	678	1070	599	1815	608	1074	556	1415	556
(0.8,0.3)	1168	674	1008	693	1007	397	1485	648	1494	600
(0.9,0.1)	1793	778	1070	531	1554	640	1483	725	1086	600
(0.9,0.2)	1992	725	1028	545	1472	575	1113	505	1480	657
(0.9,0.3)	1483	500	1068	437	1487	566	1084	558	1472	676

In the Table 2, the optimum parameter combinations are determined at(0.6/0.1) for SGA, at (0.7/0.1/0.15) for SSGA and at (0.7/0.3) for ECOGA. After setting the parameter combination for each algorithms to the optimum level obtained from the preliminary experiment, we performed the experiment 20 times while changing the population size in order to determine both optimum population size and ecosystem size. The change of the average fitness value in response to the population size is shown in Table 3.

Table 3. Change of the fitness for population size

Population size	SGA		SSGA		Eco. size	ECOGA	
	Best fitness	Average fitness	Best fitness	Average fitness		Best fitness	Average fitness
100	1494	969	1085	600	10	1113	605
200	1992	907	1355	739	15	1035	644
300	1067	810	1793	886	20	1096	790
400	1793	1042	1992	987	25	1281	1005
500	1089	952	1554	962	30	1053	991

As a results, the optimum parameter combinations for each algorithms are determined as Table 4.

Table 4. Optimal parameter combinations

Algorithm / Parameters	SGA	SSGA	ECOGA
Population size	400	400	-
Crossover rate	0.6	0.7	0.7
Mutation rate	0.1	0.1	0.3
Newly generated individuals	1000	1000	1000
Replacement rate	-	0.15	-
Eco. size	-	-	25
Neighboring population size			3

Applying these results to the spagrid3 problem, we have found 2-MVAs as shown in Table 5. Figure 3 is the program execution image[1]. It shows that the best and the worst fitness of the final population generated in each experiment (above left), the process of updating showing the convergence of fitness into the best fitness (above right), the change in the average fitness of all individuals showing the evolution of the entire population (below left), and finally, the change of the average fitness of the best fitness showing the evolution of the best fitness (below right).

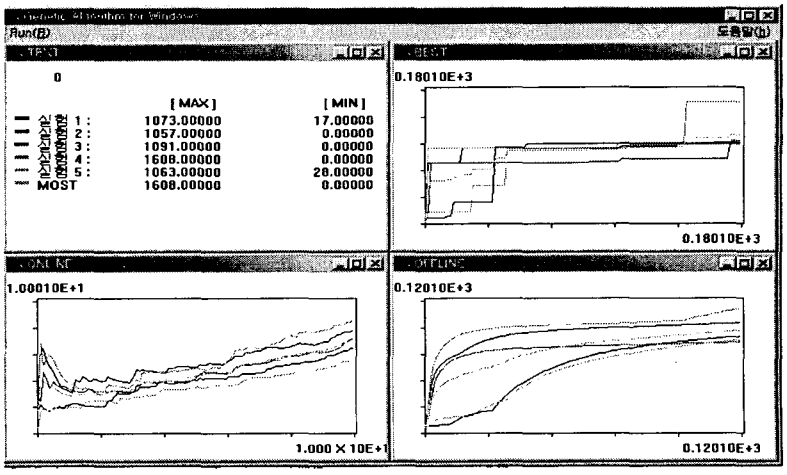

Figure 3. Program execution image

3.2 Experimental results and analysis

Before performing the experiment for the problems of DIMACS, we evaluated the performance of the proposed algorithms using 10 test problems whose optimum solution are already known. These 10 test problems are the MSTP with 5 to 18 arcs. In this experiment, we found that the proposed algorithms have a good accuracy since all the three algorithms have found the optimum solutions for the 10 test problems as shown in Table 5.

Table 5. Experiment results for small-size test problems

Problem (nodes,arcs)	Best fitness	2- MVAs	Optimum
Test 1 (5,7)	∞	{1,4},{6,7}	optimum
Test 2 (5,8)	12	{4,5}	optimum
Test 3 (6,10)	∞	{1,2},{8,10}	optimum
Test 4 (7,10)	∞	{1,2},{8,9}, {8,10},{9,10}	optimum
Test 5 (7,11)	∞	{1,2},{10,11}	optimum
Test 6 (8,13)	∞	{9,12},{10,13}	optimum
Test 7 (9,13)	∞	{1,2},{7,11},{12,13}	optimum
Test 8 (7,15)	∞	{1,2},{14,15}	optimum
Test 9 (10,15)	∞	{4,6},{7,11},{4,13},{6,13}	optimum
Test 10 (10,18)	∞	{2,3},{17,18}	optimum

Using the optimum parameter combinations of Table 4, we performed the experiment for the 20 problems generated from DIMACS. The results of this experiment are shown in Table 6. Table 6 shows that the SSGA was the best algorithm, the ECOGA was the second best one and the SGA is the third best one.

4. Conclusions

In this study, we proposed three kinds of evolutionary algorithms for finding the k-MVAs in MSTP. So far, the efficient algorithms for this problem have not been known and the algorithm proposed recently is also difficult to apply to the real world. On the contrary, the three algorithms proposed in this study have successfully found the k-MVAs in a large scales network problems with more than 20,000 arcs. Especially, for the small sized network problem with 5 to 18 arcs, the three algorithms have found perfectly the optimum solution. Based on the experiment for the 20 problems generated from DIMACS, we found that the SSGA was the best algorithm, the ECOGA was the second and the SGA was the third.

Table 6. Results of the main experiment

Problem (nodes,arcs)	SGA			SSGA			ECOGA		
	Best Fitness	Average Fitness	2-MVA	Best Fitness	Average Fitness	2-MVA	Best Fitness	Average Fitness	2-MVA
Spagrid3 (661,1980)	1253	954	{1549,1385}	1992	962	{1365,1476}	1063	1029	{1246,1365}
Spagrid4 (1289,3864)	499	316	{1034,3314}	392	282	{2408,3142}	611	365	{3142,3700}
Spagrid5 (1036,3105)	1679	1305	{2808,1152}	2642	1409	{2650,2550}	1653	1468	{2808,76}
Spagrid6 (1601,4800)	1093	576	{4457,781}	1120	496	{4457,3245}	1054	765	{4457,1605}, {4457,388},{4457,129},{370,4457},{4457,3899}, {4457,1209}
Spagrid7 (1701,5100)	395	237	{3361,3730}	384	248	{3730,1070}	383	273	{3730,2583}
Spagrid10 (2001,6000)	1489	1015	{5869,5065}	1953	1014	{4392,5567}	1341	1135	{5438,2633}
Sprand3 (409,2000)	11045	7488	{363253}	10420	7754	{1745,167}	10661	7115	{363,369}
Sprand4 (508,3000)	6261	4960	{994,806}	7795	5363	{417,1012}	7158	4955	{1970,299}
Sprand5 (890,5000)	7225	5865	{1643,2114}	10139	7117	{4755,2113}	8783	5385	{1865,1251}
Sprand5 (999,5003)	9594	6648	{425,1121}	8510	6130	{2049,3506}	7926	5699	{4473,3106}
Sprand7 (1200,6000)	8454	6186	{5411,889}	10832	7119	{4142,4198}	10195	6558	{546,4142}
Sprand8 (1367,7645)	7619	5849	{7626,7443}	9360	6374	{3156,3081}	6656	5276	{1246,806}
Sprand9 (1789,3837)	17147	13969	{3583,255}	17776	14448	{92,920}	∞	∞	{1186,1185}
Sprand10 (2000,8999)	9272	7191	{7826,6612}	11548	8305	{5278,69}	9260	6578	{3409,1576}
Sprand11 (2789,10000)	13025	9040	{3268,60}	14446	10023	{877,5995}	11255	8996	{1009,4249}
Spacyc5 (530,2450)	8746	6764	{1420,268}	9600	7144	{200,268}	7918	6465	{234,1383}
Spacyc6 (770,3050)	13668	8724	{682,675}	13232	9721	{64,675}	12054	6389	{675,2278}
Spacyc7 (1120,7025)	8028	5207	{3140,5570}	7042	5264	{6252,2494}	6637	5039	{3013,1110}
Spacyc8 (1597,14364)	5642	3614	{11350,2999}	67600	4110	{3247,9012}	5355	5311	{8775,218}
Spacyc10 (1956,20140)	3421	2516	{12262,18189}	4139	2863	{2351,4991}	3849	2445	{816,8274}

References

[1] Ahuja. R. J., Magnanti. T. L., Orin. J. B., Network Flows : Theory, Algorithms, and Applications, Prentice-Hall, 1992

[2] F. Suraweera, P. Maheshwari, and P. Bhattacharya, "Optimal Algorithms to Find the

Most Vital Edge of a Minimum Spanning Tree", Technical Report CIT-95-21, School of Computing and Information Technology, Griffith University, 1995

[3] Goldberg.A.V., http: // www.neci.nj.nec.com/ homepages / avg

[4] H. Shen, "Finding the k Most Vital Edge with respect to Minimum Spanning Tree", School of Computing and Information Technology Griffith University Nathan, QLD 4111, Australia.

[5] K. C. Lin and M. S. Chern. "The Most Vital Edge in the Minimum Spanning Tree Problem", Information Processing Letters, Vol. 45 (1993), pp. 25-31

[6] K. Iwano and N. Katoh, "Efficient Algorithms for Finding the Most Vital Edge of a Minimum Spanning Tree", Information Processing Letters, Vol. 48 (1993), pp. 211-213

[7] L. H. Hsu, R. H. Jan, Y. C. Lee, C. N. Hung, and M.S. Chern, "Finding the Most Vital Edge with respect to Minimum Spanning Tree in Weighted Graphs", Information Processing Letters, Vol. 39 (1991), pp. 277-281

[8] M.O. Ball, B.L. Golden, and R.V. Vohra, "Finding the Most Vital Arcs in a Networks", Operations. Research Letters, 8:73-876, 1989

[9] Ratliff H.D., S.H.Lubore, G.T.Sicilia, Finding the n Most Vital Links in a Flow Network, Management Sci, vol.21, No.5 (1975), pp.531-539

[10] S. Y. Lee, H. Y. Chung, G. U. Seo, Y. K. Kim, "Development of Application Framework for the Simple Evolutionary Algorithm", Korean Journal of Industrial Engineering , Vol. 12, No.4 (1999), pp. 540-550

[11] Y. K. Kim, B. S. Yoon, S. B. Lee, Meta-Heuristic, Youngji Co., 1997