Journal of the Nigerian Association of Mathematical Physics
Volume26, (*March, 2014*), pp 388 – 393

Memetic Algorithm with Multi-Parent Crossover (MA-MPC) for Multi-Objective Network Design

R. O. Oladele[1] and O. T. Oladele[2]

[1]Department of Computer Science, University of Ilorin
P.M.B. 1515, Ilorin, Nigeria.
[2]Department of Computer Science, Federal University of Technology
P. M. B. 704, Akure, Nigeria.

Abstract

In many Evolutionary Algorithms (EAs), a crossover with two parents is commonly used to produce offsprings. Interestingly, we need not restrict ourselves to two-parent crossover since EA allows us to emulate natural evolution in a more flexible fashion. There are experimental results in the literature which show that multi-parent crossover operators can achieve better performance than traditional two-parent versions. However, most of these experimental results are based on common test functions. Experimental studies involving real-life, NP-hard problems such as network design problem are very rare. This paper presents Memetic Algorithm with Multi-Parent Crossover (MA-MPC) with a view to providing a case study of multi-parent crossover within the framework of MA for network topology design problem. Results show that MA-MPC does not always outperform MA. It depends on the size of the problem and the number parents (be it 3, 5, 7, or any other).

Keywords: Evolutionary Algorithms, Multi-Parent Crossover, NP-hard problems, Memetic Algorithm, network design

1.0 Introduction

Memetic Algorithm (MA) isknown to be one of the highly effective meta-heuristic approaches for solving a large number of constraint satisfaction and optimization problems [1]. One of the key features of a MA is the crossover operator for generating offspring solutions. Generally, meaningful crossover operators enhance healthy diversification in the population and prevent premature convergence of the population.In many Evolutionary Algorithms (EAs), a crossover operation with two parents is commonly used to produce offsprings. However, we need not restrict ourselves to two-parent crossover only as EAs allows us to emulate natural evolution in a more flexible fashion. Several attempts studying the use of more than two parents for crossover in EAs have been reported [1, 2, 3,4]. In fact, there are experimental results in the literature which show that multi-parent crossover operators can achieve better performance than their two-parent versions.Elsayed et al [5] showed that the efficiency of genetic algorithm (GA) can be enhanced by using multi-parent crossover with a randomized operator. Ting and Buning [6] proposed a tabumulti-parent genetic algorithm (TMPGA**)** and observed from their computational experiments that TMPGA achieved faster convergence and better solution quality than the classic two-parent GA. Eiben [7, 8] provides excellent overview of multi-parent crossovers. These researchers and a host of others havedemonstrated the effectiveness of multi-parent crossoverson functionaloptimization problems as well as multi-objective problems. However, computational experiments involving, real-life, NP-hard problems such as network design problem are very rare.

In this paper, we provide a case study of (diagonal) multi-parent crossover operatorwithin memetic algorithm for multi-objective network design problem.

Corresponding author: R. O. Oladele E-mail: roladele@yahoo.com, Tel.: +2347064203812

1.1 Problem Formulation

We consider a multi-objective network design problem of the form:

Minimize $y = f(x) = (f_1(x), f_2(x)$ (1)

Subject to: $LFLOW_{ij} \leq CAP_{ij}$ (2)

$R(x) \geq R_0$ (3)

Where:

$x = (x_1, x_2, \ldots\ldots\ldots, x_n) \in X$ is the decision vector

$y = (f_1, f_2) \in F$ is the objective vector

$f_1(x)$ is the cost function of the configuration x

$f_2(x)$ is the average delay on all the links in the configuration x

$LFLOW_{ij}$ refers to the traffic flowing along link (i,j)

CAP_{ij} is the capacity of link (i,j)

$R(x)$ is the reliability of the configuration x

R_0 is the minimum acceptable reliability (R_0 = 0.95)

The reliability calculation is done via Monte Carlo simulations.

Other network design parameters used are the followings:

N denotes the total number of nodes in the network

D_{ij} denotes the physical distance between every pair of nodes i and j

C_{ij} represents the cost of the link between nodes $i$and j

$C_i$is the cost of network equipment at node i

P_{ij} is selection status of link (i,j) : P_{ij} = 1 if link (i,j) is selected, else P_{ij} = 0

L = maximum distance for which the signal is sustained without amplification (We fix L = 15km)

A = cost of each amplifier unit(#6.00)

Poisson process was used to model the traffic delay

The objective functions; network cost and average delay are approximated by the following formulation.

i Network Cost:

$NetCost = NodeCost + LinkCost + AmpCost$ (4)

Where;

$NodeCost = \sum_i C_i$ (5)

$LinkCost = \sum_i \sum_j C_{ij}$ (6)

$AmpCost = \frac{\sum_i \sum_j D_{ij} \ X \ A}{L}$ (7)

ii Average Delay:

$AvDelay = \frac{\sum_i \sum_j [DELAY_{ij} X LFLOW_{ij}]}{\sum_i \sum_i LFLOW_{ij}}$ (8)

$DELAY_{ij} = \frac{1}{[CAP_{ij} - LFLOW_{ij}]}$ (9)

$DELAY_{ij}$ = 0 if there is no link between nodes i and j

$DELAY_{ij} = \infty$ if the network cannot handle the traffic load with the existing links' capacities and routing policy.

The constraints are:

i. Flow constraints which can be expressed as:

$\boldsymbol{LFLOW_{ij} \leq CAP_{ij}}$ (10)

and

ii. Reliability constraint which can be stated as:

$R(x) \geq \boldsymbol{R_0}$ (11)

Monte Carlo Simulation is used to estimate network reliability. The network is simulated t times, given the design and the links' reliabilities. The method is outlined below.

initialize i = 0, c= 0

Step C0: while i < t Repeat.

Step C1: Randomly generate network

(a): i = i + 1.

Step C2: Check to see if the network forms a spanning tree

(a): if YES, increment c by 1 and go to Step C0

(b): if NO, go to Step C0

Step C3: R(x) = c / t.

Breadth First Search (BFS) is used for routing. The metric used for this purpose is the length of the link. The following assumptions were made in the problem formulation

The location of each network node is given

Each C_{ij} is fixed and known

Each link is bidirectional i.e. a path can be traversed in either direction

There is no redundant link in the network

2.0 Algorithm design and Implementation Details

2.1 MA-MPC

The template of MA-MPC used in this paper is as follows:

1 Initialization: generate a population of N chromosomes
2 Fitness: calculate the fitness of each chromosome
3 Create a new population:
   a. Selection: select m chromosomes from the population ($m \geq 3$)
   b. Crossover: produce m offsprings from the $m$selected chromosomes
   c. Local Search: apply local search to each offspring
   d. Mutation: perform mutation on each offspring.
   e. Local search: apply local search to each offspring.
4 Replace: replace the current population with the new population
5 Evaluation: compute the objective vector of each chromosome
6 Termination: Test if the termination condition is satisfied. If so stop. If not, go to step 2

2.2 Implementation Details

This presents relevant details concerning the implementation of MA-MPC

3.0 Encoding Scheme

The chosen encoding scheme is such that every chromosome codes a possible network, which corresponds to an individual in a set of feasible solutions of the problem. This set of feasible solutions constitutes a population. The chromosome is represented by a constant length integer string representation. The chromosome consists of two parts, the first part contains details of NE's at the nodes and the second part consists of details of the links. For example, if there are H types of nodes, then $\log_2 H$ bits are required to encode a node. Therefore the first part of the chromosome consists of $N . \log_2 H$ bits, where N is the number of nodes in the network. If a link exists between nodes 1 and 2 then the first bit position in the link part is set to 1. Hence the second part of the chromosome consists of (N (N-1))/2 bits.

3.1 Initial Population

The two algorithms start by creating an initial population. There are two ways of generating initial population namely heuristic process and random process. A random process of generating initial population is adopted.

3.2 Fitness Evaluation

Fitness of a chromosome is evaluated based on principle of Pareto ranking. Pareto-rank of each individual is equal to one more than the number of individuals dominating it. All non-dominated individuals are assigned rank one. Network cost and average delay are used to evaluate the rank of an individual chromosome using the principle of Pareto dominance. The fitness of an individual as defined in [9] is given by

$$\boldsymbol{Fitness} = \frac{\mathbf{1}}{\boldsymbol{Rank^2}} \tag{12}$$

3.3 Selection

Roulette Wheel Selection Process is used. In roulette wheel, individuals are selected with a probability that is directly proportional to their fitness values i.e. an individual's selection corresponds to a portion of a roulette wheel. The probabilities of selecting a parent can be seen as spinning a roulette wheel with the size of the segment for each parent being proportional to its fitness. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The fittest individual occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel. The circumference of the roulette wheel is the sum of all fitness values of the individuals. The proportional roulette wheel algorithm procedure is depicted by the algorithm below. Let $\boldsymbol{f_1, f_2, \ldots\ldots f_n}$ be fitness values of individuals 1, 2, ……n. Then the selection probability, $\boldsymbol{P_i}$, for an individual $\boldsymbol{i}$, is given as

$$\boldsymbol{P_i} = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{13}$$

The template of the roulette wheel selection procedure is shown below.

Procedure: Roulette Wheel Selection
while population size < pop_size do
generate pop_size random number r
calculate cumulative fitness, total fitness ($\boldsymbol{P_i}$) and sum of proportional fitness (sum)
Spin the wheel pop_size times
if sum < r then
select the first chromosome
else
select the jth chromosome
Endif
Endwhile
return chromosomes with fitness values proportional to the size of the selected wheel section
End Procedure

3.4 Crossover

This operation operates on two(or more) chromosomes. In particular, for MA-MPC we used 3, 5 and 7 chromosomes. To provide a basis for evaluating MA-MPC we also implemented crossover using 2 chromosomes, that is, traditional MA. The chromosomes are randomly selected based on the probability of crossover which is a randomly generated number ranging between 0 and 10. In this work, two-point crossover technique was implemented. The *crossover probability* (denoted by *p*C) is the probability of the number of offsprings produced in each generation to the population size (denoted by *popSize*). This probability controls the expected number (*p*C × *popSize)* of chromosomes to undergo the crossover operation. A high crossover probability is used here to allow exploration of more of the solution space, and reduces the chances of settling for a false optimum; but if this probability is too high, it results in the wastage of a lot of computation time in exploring unpromising regions of the solution space.

3.5 Mutation

This is the operation of randomly changing some of the bits of the chromosome representing an individual with a view to increasing the exploration of the solution space.

3.6 Local Search

The local search technique used is the hill climbing search algorithm. It is essentially an iteration that continuously proceeds in the direction of increasing quality value. The algorithm is as shown below

While (termination condition is not satisfied) do
New solution ← neighbours(Best solution);
If new solution is better than actual solution then
Best solution ← actual solution
End if
End while

4.0 Computational Results And Discussion

In this section, results of numerical experiments using 3 test problems - 10-node network design problem, 21-node network design problem and 36-node network design problem (See Appendix) are reported. All experiments were performed on a HP 630 NOTEBOOK PC with the following configuration:2.13GHz Processor Speed, 3.0GB RAM and 64 BIT OS and the implementation language is java.

For MA-MPC, a total of 18 simulation runs were carried out for the 3 test problems (6 runs per problem instance, 2 runs per multi-parent instance implementation), noting the pareto-optimal front. For MA (crossover using 2 chromosomes), a total of 18 simulation runs were equally carried out, noting the pareto-optimal front. The algorithms were implemented with the following parameters:

Population size - 100 (250 for 36-node network design problem)
Mutation probability – 0.02
Number of parents 3, 5, 7
Number of Node Type – 4

The worst-ranked results (out of the pareto-optimal front) of MA and MA-MPC (3, 5 and 7 parents) are extracted and then re-ranked among the extracted results as shown in tables 1 to 3. Where two different results tie, the average of the two results is recorded.

For 10-node network, it is evident from Table 1 that the use of multi-parent crossover will always enhance the efficiency of MA since MA has the highest CPU time (410 seconds). It could also be observed that quality of results is either degraded as it is the case with MA-MPC with 3 and 5 parents or not affected at all by multi-parent crossover as it is the case with MA-MPC with 7 parents.

For 21-node network, Table 2 shows that multi-parent crossover will reduce computation time of MA whenever it is used. Results quality is however either reduced or left unaffected by multi-parent crossover.

In the case 36-node network, Table 3 depicts that multi-parent crossover can either improve or impede the efficiency of MA depending on the number of parents involved. Results also reveal that when the efficiency of MA is improved owing to the use of multi-parent crossover, its effectiveness is equally hampered and vice versa. The table also shows that multi-parent crossover will either improve solution quality as it is the case with MA-MPC (with 5 parents) or degrade solution quality as it is the case with MA-MPC (with 3 and 7 parents)

5.0 Conclusion

In this paper, the impact of multi-parent crossover on a Memetic Algorithm applied to solve real-life problem is investigated. In particular, MA-MPC is designed and tested for a multi-objective network design problem whose size ranges from small (10-none network), medium (21-node network) to large (36-node network). The results obtained show that, while multi-parent crossover will certainly improve the efficiency of MA for small and medium networks, it is not the case for large networks. For large networks, multi-parent crossover will either improve or impede the efficiency of MA depending on the number of parents used.

6.0 Acknowledgments

Many thanks to Hammed Sanusi for programming assistance.

Table 1: Table of Results for 10-node network problem

Number of Parents	Cost	AvDelay	Rank	CPU Time
2 (MA)	619.6	0.05	1	410
3	934.9	0.045	3	150
5	758	0.03	2	142
7	689.6	0.03	1	185

Table 2: Table of Results for 21-node network problem

Number of Parents	Cost	AvDelay	Rank	CPU Time
2 (MA)	1167.4	0.04	1	5665
3	1354.3	0.07	4	2899
5	1293.8	0.03	1	5390
7	1350.4	0.03	2	990

Table 3: Table of Results for 36-node network problem

Number of Parents	Cost	AvDelay	Rank	CPU Time
2 (MA)	1167.4	0.05	2	5673
3	1258.9	0.055	3	4744
5	1163.8	0.04	1	6455
7	1350.4	0.06	4	990

APPENDIX
TEST DATA

10-NODE NETWORK

Node Details (Node Type, $\boldsymbol{C_i}$) = { (01,42) ,(0, 78) ,(10,33) ,(00,53) ,(01,42) ,(00,13) ,(10,9) ,(11,23) ,(10,57) ,(10,25) }

Link Details ($\boldsymbol{D_{ij}}$, $\boldsymbol{C_{ij}}$, $\boldsymbol{CAP_{ij}}$, $\boldsymbol{LFLOW_{ij}}$) = { (28,47,60,46) ,(20,43,90,72) ,(28,12,54,28) ,(62,39,61,46) ,(42,23,24,9) ,(42,30,16,14),(36,3,44,16) ,(40,18,75,54) ,(10,36,29,8) ,(44,30,79,53) ,(44,45,54,35) ,(36,18,66,51) ,(32,13,78,25) ,(14,16,96,54) ,(16,13, 84,74) ,(21,28,76,17) ,(22,3,80,71) ,(3,39,55,54) ,(47,12,66,62) ,(26,11,89,56) ,(13,42,77,47) ,(46,22,45,39) ,(28,6,53,16) ,(5,38,89,57) ,(28,40,16, 9) ,(48,49,49,40) ,(18,34,37,9) ,(34,35,11,8),(11,41,39,31) ,(46,20,32,9) ,(11,3,50,35) ,(70,1,54,41) ,(18,6,8,65) ,(35,42,91,66) ,(14,33,10,26) ,(11,33,60,9) ,(43,16,79,49) ,(20,43,88,56) ,(16,13,96,68) ,(6,30,91,67) ,(34,49,16,7) ,(37,21,57,49) ,(20,12,79,62) ,(33,46,81,70) ,(48,25,8,7) }

21-NODE NETWORK
Node Details = { (01,42) ,(01,78) ,(10,33) ,(00,53) ,(01,42) ,(00,13) ,(10,9) ,(11,23) ,(10,57) ,(10,25) ,(01,53) ,(00,55) ,(01,11) ,(00,34) ,(10,33) ,(00,32) ,(01,51) ,(10,38) , (10,15) ,(10,57) }
Link Details = { (29 20 87 74) , (4 15 75 43) , (13 47 50 35) , (41 16 69 52) , (32 25 72 54) , (43 42 89 63) , (31 1 75 21) , (29 38 76 70) , (1 33 39 11) , (16 37 59 56) , (30 44 70 42) , (6 47 89 77) , (10 12 81 18) , (30 6 66 37) ,(26 3 80 54) ,(17 45 98 33) ,(12 10 49 39) ,(32 9 61 31) ,(6 45 35 24) ,(14 42 67 24) ,(19 10 89 31) ,(6 31 73 28) ,(31 19 29 22) ,(12 21 80 3) ,(49 35 60 13) ,(15 30 95 39) ,(15 3 29 9) ,(4 2 82 64) ,(26 27 27 12) ,(24 42 57 2) ,(25 46 68 66) ,(23 18 47 14) ,(5 28 94 65) ,(30 18 44 26) ,(7 18 78 59) ,(20 44 80 38) ,(33 29 30 7) ,(18 10 99 5) ,(25 43 18 2) ,(30 30 94 13) ,(0 26 82 46) ,(22 0 87 52) ,(40 6 63 15) ,(10 41 46 3) ,(25 45 35 10) ,(15 22 35 56) ,(46 28 32 5) ,(13 8 31 47) ,(17 18 29 35) ,(5 24 89 70) ,(36 25 90 76) ,(32 20 94 75) ,(40 34 84 73) ,(7 29 94 53) ,(39 35 33 21) ,(37 42 57 4) ,(43 41 71 60) ,(20 28 85 45) ,(36 17 51 19) ,(22 19 83 48) ,(44 17 28 19),(36 37 40 19) ,(32 36 40 14) ,(4 12 88 78) ,(32 47 88 8) ,(48 19 27 7) ,(26 7 73 60) ,(28 13 32 1) ,(20 21 47 19) ,(41 28 84 54) ,(30 28 78 66) ,(20 38 92 67) ,(21 27 88 27) ,(37 21 63 56) ,(27 22 57 35) ,(3 48 39 38) ,(20 32 62 56) ,(17 33 74 60) ,(41 24 60 14) ,(11 4 93 44) ,(20 44 75 74) ,(49 30 73 52) ,(39 16 64 57) ,(12 40 62 54) ,(33 16 12 70) ,(43 20 83 48) ,(0 16 93 71) ,(23 29 40 8) ,(2 35 81 36) ,(11 38 78 62) ,(7 11 93 63) ,(0 33 94 74) ,(9 48 88 54) ,(9 46 86 69) ,(15 44 87 32) ,(12 18 51 43) ,(16 24 79 43) ,(28 8 68 7) ,(41 49 67 27) ,(29 24 78 60) ,(48 5 63 12) ,(18 22 23 1) ,(22 31 17 7) ,(14 45 58 4) ,(3 45 73 64) ,(16 28 89 71) ,(5 8 59 26) ,(16 49 65 50) ,(6 25 39 9) ,(48 35 76 73) ,(1 30 35 63) ,(19 29 82 34) ,(35 27 57 20) ,(43 10 73 70) ,(17 28 25 15) ,(44 30 14 3) ,(5 20 63 12) ,(19 40 46 59) ,(8 30 82 50) ,(7 9 5 54) ,(7 10 74 66) ,(30 14 18 55) ,(7 5 83 15) ,(30 33 69 64) ,(15 10 64 61) ,(33 11 14 2) ,(18 31 79 75) ,(2 38 66 0) ,(47 0 29 17) ,(20 29 52 48) ,(46 38 10 4) ,(5 45 75 40) ,(39 17 83 66) ,(18 3 94 4) ,(30 25 60 43) ,(31 32 84 71) ,(34 45 74 74) ,(5 19 68 42) ,(27 48 72 69) ,(13 6 45 33) ,(20 17 37 23) ,(41 26 97 30) ,(34 42 54 22) ,(5 42 63 47) ,(39 26 71 47) ,(18 28 20 15) ,(18 4 70 10) ,(16 12 87 32) ,(1 13 97 11) ,(27 39 71 62) ,(41 14 94 59) ,(1 33 63 57) ,(12 2 70 43) ,(37 4 77 51) ,(25 16 85 21) ,(8 17 40 25) ,(6 11 87 42) ,(48 47 97 21) ,(28 39 42 21) ,(18 21 19 15) ,(46 12 99 51) ,(1 27 99 76) ,(8 31 21 11) ,(6 13 91 36) ,(27 17 69 21) ,(18 16 80 38) ,(42 20 92 19) ,(38 33 66 33) ,(47 5 57 27) ,(39 3 45 19) ,(30 4 88 78) ,(39 14 60 30) ,(28 40 92 41) ,(2 48 63 22) ,(17 10 83 42) ,(30 17 71 50) ,(14 20 66 79) ,(10 27 83 65) ,(43 27 89 40) ,(5 2 97 78) ,(8 13 67 57) ,(42 21 68 47) ,(5 20 84 23) ,(41 28 61 51) ,(41 38 70 31) ,(8 48 80 59) ,(9 10 97 25) ,(40 38 86 55) ,(31 20 65 60) ,(39 39 41 40) ,(22 40 95 77) ,(44 16 72 56) ,(21 45 88 53) ,(29 2 74 37) ,(21 45 60 40) ,(4 37 37 7) ,(12 43 48 38) ,(3 43 63 48) ,(46 25 63 43) ,(4 31 53 51) ,(18 9 70 22) ,(47 9 76 70) ,(38 13 75 44) ,(20 49 83 52) ,(21 15 90 25) ,(9 45 66 49) ,(42 40 55 0) ,(1 37 80 27) ,(40 8 88 28) ,(9 27 73 38) }

7.0 References

[1] Lu, Z., Hao, J. K., Glover, F.: A study of memetic search with multi-parent crossover for UBQP. In: Cowling, P., Merz, P. (eds.) EvoCOP 2010. LNCS, vol. 6022, pp.154–165. Springer, 2010

[2] Eiben, A. E, Back, T., Empirical Investigation of Multiparent Recombination Operators in Evolution Strategies. Evolutionary Computation, 5(3), pp. 347-365, 1997.

[3] Palubeckis, G., Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Annals of Operations Research 131, 259–282, 2004.

[4] Ting, C. K., Design and analysis of multi-parent genetic algorithms, PhD Thesis, University of Paderborn, 2005.

[5] Elsayed, S., Sarker, R. and Essam, D., "GA with a New Multi-Parent Crossover for Solving IEEE-CEC2011 Competition Problems", in *Proc. Congress on Evolutionary Computation*, pp. 1034 - 1040, New Orleans, June 2011

[6] Ting, C. K., Buning, H. K., A Mating Strategy for Multi-parent Genetic Algorithms by Integrating Tabu Search. in*Proc. Congress on Evolutionary Computation*, vol. 2, pp. 1259 - 1266, 2003

[7] Eiben, A. E., "Multiparent recombination," Evolutionary Computation 1: Basic Algorithms and Operators, pp. 289-307, Institute of Physics Publishing, 2000.

[8] Eiben, A. E., "Multiparent recombination in evolutionary computing," Advances in EvolutionaryComputing, Natural Computing Series, Springer, 2002.

[9] Banerjee, N, Kumar, R., "Multiobjective network design for realistic traffic models", In *Proceedings of GECCO '07,* pp. 1904-1911, 2007