M – 1

MODIFICATION OF CROSSOVER OPERATOR ON GA APPLICATION FOR TSP

Darmawan Satyananda
Jurusan Matematika FMIPA Universitas Negeri Malang
darmawan.satyananda.fmipa@um.ac.id

Abstract

Genetic Algorithm (GA) has been widely used in many fields of optimization; one of them is Traveling Salesman Problem (TSP). GA in the TSP is primarily used in cases involving a lot of vertices, which is not possible to enumerate the shortest route. One of stages in GA is crossover operation to generate offspring's chromosome based on parent's.

Example of some crossover operators in GA for TSP are Partially Mapped Crossover (PMX), Order Crossover (OX), Cycle Crossover (CX), and some others. However on constructing the route, they are not considering length of the route to maximize its fitness. The use of random numbers on constructing the route likely produces offspring (a new route) that is not better than its parent. Sequence of nodes in the route affects the length of the route. To minimize uncertainty, then the crossover operation should consider a method to arrange the chromosomes.

This article studied incorporating two methods into crossover stage, in order to ensure the offspring has good fitness. Methods to be combined with algorithms are commonly used in the route searching; those are Nearest Neighbor algorithm, and Sequential Insertion. Operators used are CSI (Crossover combined with Sequential Insertion) and CNN (Crossover combined with Nearest Neighbor), named after the method used.

Those operators are compared with PMX operator on test using benchmark data from TSPLIB on some independent executions. The tests showed that CSI are better than two other and length of its route was relatively equal to optimal length recorded.

Keywords: Genetic Algorithm, Traveling Salesman Problem, Crossover operator

Introduction

A. Background

1. Traveling Salesman Problem

Traveling Salesman Problem (TSP) is one of applications of graph theory. TSP is a problem of finding minimum weight Hamiltonian cycle in a graph. TSP models a salesman who must visit all cities exactly once and return to his hometown, with least distance (or cost, time, and other). Node in the graph symbolizes city, and weight between two nodes symbolizes distance (or cost, time, etc.) between two cities. TSP applications are used to solve some problems, some of them are distribution/transportation, job scheduling, scheduling lectures, PCB drilling, X-ray crystallography, and computer wiring ([Al Rahedi & Atoum, 2009], [Philip, Taofiki, and Kehinde, 2011]).

In algorithmic term, TSP is NP-complete problem, meaning that require completion in polynomial time as number of node increases. The only optimal algorithm for TSP is Brute Force algorithm, but it is not efficient because adding 1 node into the graph will increase computation by 1000%. Up to now there is no known algorithm to solve the TSP that is both optimal and efficient, particularly to large number of nodes.

At *n*-node symmetric graph, there are (n-1)!/2 possible Hamiltonian cycle. By using exhaustive search (brute force), it would have obtained a solution (it's optimal), but this would

require a very large running time (it isn't efficient). For some problems it is possible to design algorithms that are significantly faster than exhaustive search, though still not polynomial time (Woeginger, 2002).

A number of methods that considered "traditional" and exact, has been used to solve TSP, among them is Nearest Neighbor, Cheapest Link, and Christofides. They provide exact optimal solution to the problem. Several other methods are "non-traditional", among of them are Simulated Annealing (SA), Genetic Algorithms (GA), Ant Colony Optimization (ACO), Bee Colony Optimization (BCO), Particle Swarm Optimization (PSO) (Sureja & Chawda, 2012). These are stochastic method that mimics some natural phenomenon, which is designed to problems that cannot be dealt with classical methods (Michalewics, 1996). "Non-traditional" method looks for a "good" solution within a reasonable time, instead of searching for the optimal solution (Ahmed, 2010).

2. Genetic Algorithm (GA)

GA is one of the best heuristic algorithms that have been used widely to solve the TSP instances. GA mimics Darwinian evolutionary principles of natural evolution, in which the individual must be able to adapt to a changing environment. Good quality individuals will survive, and the bad will be extinct. Good individuals are expected to produce good offspring as well. Each individual has chromosomes; parent's chromosomes are passed on to offspring through the process of crossover (marriage), and chromosome of an individual can change due to mutation. Michalewics (1996) and Al Rahedi & Atoum (2009) stated that GA can be applied to problems such as routing optimization, production scheduling, transportation, TSP, control, games, arrangement layout, and database query optimization.

In general, GA consists of some stages (Dr´eo, 2006):

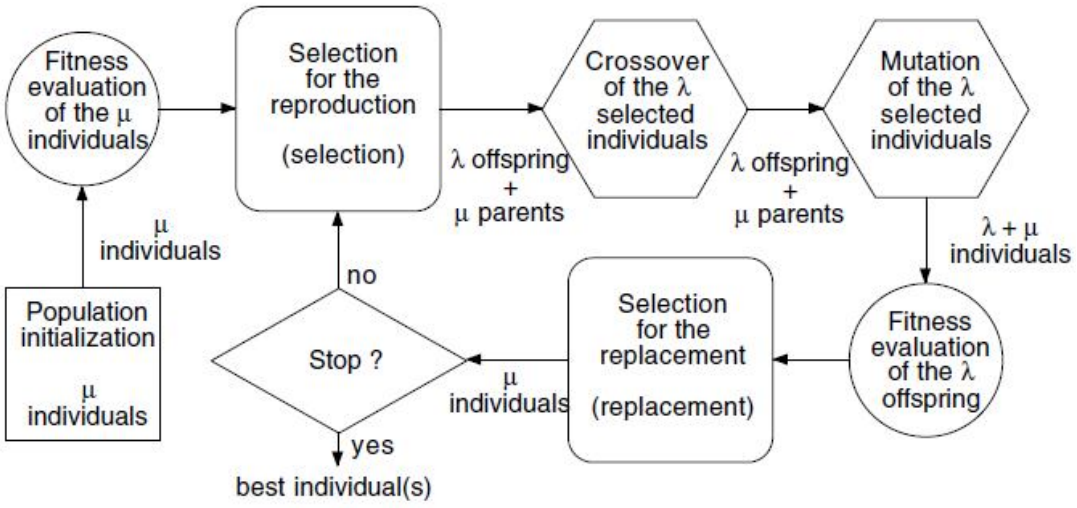

Figure 1. Stages on Genetic Algorithm

Initial population (μ individuals) can be determined randomly; this is generally defined in input parameter. Each individual has a single chromosome (the terms are interchangeably, it is often synonymous between individuals and chromosomes). Each chromosome consists of a number of genes. In the use of GA for solving the TSP case, the gene is a representation of node (vertex) in the graph and chromosome is a representation of route (Hamiltonian cycle). Chromosome size is equal to the number of nodes in the graph. Chromosome {1,4,3,2,5} represents a complete tour {1→4→3→2→5→1}

Fitness evaluation is used to determine the quality of a chromosome. Fitness evaluation function equals to the objective function. For TSP, the fitness evaluation is to calculate the length of the route (Hamiltonian cycle) from a particular starting point. In general GA seeks maximum objective function (the best fitness is the maximum value), whereas in the TSP is to

find the minimum of the objective function. Hence the fitness function is inverted; that is $F(x) = \frac{1}{f(x)}$, where $f(x)$ is the objective function.

GA process is done iteratively. There are three operations that apply on each iteration:

- Selection operation to choose chromosomes to be crossed (subject to crossover operation), based on the fitness of chromosomes (good-fitness chromosome has a great chance of being selected).
- Crossover operation to combine two individual parent's chromosomes, produces a new chromosome for the individual offspring (sexual reproduction)
- Mutation operation to modify gene of a chromosome (asexual reproduction)

End of the mutation operation is calculating fitness of chromosomes resulted from crossover and mutation operation. In the end of iteration, selection is performed to select μ individuals out of λ + μ individuals resulted from mutation operation. Chromosomes resulted from mutation phase is sorted by its fitness ascending, and then only μ chromosomes are taking into account. These are chromosomes with the best fitness and become individuals for initial state of next iteration. This is survival selection which is parents and offspring compete to survive.

Al Rahedi & Atoum (2009) stated that there are 15 kinds of methods for crossover operation and 11 kinds of methods for mutation operations are discussed in some literatures. Not all methods can be applied to a particular problem.

Stages from selection to replacement is a whole generation. Those stages will be repeated several times (to state some generations of population), and somehow it stops on a certain condition: number of generation, goodness of solution to a specific standard, convergence of population, or any other specific stopping criterion (Rexhepi, Maxhuni, and Dika, 2013).

3. Crossover operator

Crossover operation is necessary to produce a good individual. Parents' characteristics are inherited mainly by crossover operator. The operator that preserves good characteristics in the offspring is said to be good operator (Ahmed, 2010). Three crossover operators most commonly used for TSP are PMX (Partially Mapped Crossover), OX (Ordered Crossover), and CX (Cycle Crossover). Number of researches produce other operator such as ERX (Edge recombination operator), GNX (Generalized N -point crossover), TBX (Tie-Break Crossover), and MX (Moon Crossover. Among those, perhaps PMX is the best since which it founds near-optimal solutions to a well-known 33-node problem (Ahmed, 2010).

In general, the crossover operation on TSP ensures no similar genes on one chromosome (since the route only passes a vertex exactly once). Length of the route has not been taken into consideration at the time because it can be done by selecting the best route at the end of the iteration. It means that the route cannot achieve optimal state immediately.

B. Formulation of the problem

According to the fact that crossover operators for solving TSP don't incorporate node arrangement in generating route, the route cannot achieve optimal state immediately. This research will incorporate heuristic for arranging nodes into crossover stage, to ensure GA could generate route optimally. The heuristic will be stated as crossover operator.

C. Goal and Benefit

Goal of this research is to crossover operator that produce the most optimal route. Meanwhile it's benefit is to have GA for TSP which has more optimal solution and faster convergence.

Research Method

A. Crossover operator proposed

In all crossover operations to TSP, each generated chromosome (offspring's chromosome) is valid, but cannot guarantee that produce individual offspring better than their parents. It is proposed intervention on chromosome arrangement. This intervention is done by utilizing the principle of the Nearest Neighbor and Sequential Insertion algorithm. Both only produce one chromosome of offspring for every pair of parent's chromosome subjected to crossover operation. Offspring chromosome is added to the end of the set of chromosomes in the crossover phase.

Nearest Neighbor is one of algorithms that are used to get a minimum weight Hamiltonian cycle on a graph. It has a complexity of O (n^2). According to Rosen (2000), Nearest Neighbor algorithm is:

input:undirected network $G = (V,E)$
output:a traveling salesman tour
$i0$:= any vertex of G {the starting vertex}
$W := V - \{i0\}$
$P := \emptyset$
$v := i0$
while $W \_= \emptyset$
let $k \in W$ be such that $cvk = \min\{ cvj \mid j \in W \}$
add (v, k) to P
$W := W - \{k\}$
$v := k$
add $(k, i0)$ to the path P to produce a tour

In the Nearest Neighbor algorithm, from one starting node searched next node which has closest distance. Selected node serve as a starting point for the next iteration. It will take place until there is no more nodes that may be selected, and in the end the last node is connected with the initial starting node to form the complete tour.

The selected node is added to the last node in the route (which is closest to the previous node). However, according to Gutin & Punned (2002) this method has a drawback, that it does not exploit the geometric structure of the problem. It means that there are possibilities that different position of the nodes (that make different route) would give better result.

Nearest Insertion Algorithm uses node insertion to the existing route, by selecting position that can maximize saving: inserting a node between two other nodes will give shorter distance. Suppose there are two nodes i and j, and a node k will be inserted between them, then total distance $c_{ik} + c_{kj}$ must be smaller than c_{ij}. Since there are many possible pairs as that, then the algorithm will select position that gives least $\min(c_{ik} + c_{kj,} c_{ij})$. Basically this originates from Clark and Wright in 1964. In other literature, this is used in VRP, known as Sequential Insertion in Poot, Kant, and Wagelmans (2002), and as a base of developing Constricting Insertion Heuristic for TSP with Neighborhood in Alatartsev, Augustine, and Ortmeier (2013).

Still according to Rosen (2000), Nearest Insertion algorithm is:

input:undirected network $G = (V,E)$
output:a traveling salesman tour
i := any vertex of G {the starting vertex}
j := subscript such that $cij = \min\{ cir \mid r \in V - \{i\} \}$
$S := \{i, j\}$
$C := \{(i, j), (j, i)\}$
while $S \_= V$
let k be such that $dS(k) = \min\{ dS(r) \mid r \in V - S \}$
$S := S \cup \{k\}$
find an edge $(u, v) \in C$ so $cuk + ckv - cuv = \min\{ cxk + cky - cxy \mid (x, y) \in C \}$
add (u, k) and (k, v) to C, and remove (u, v) from C

This algorithm firstly looks for a node that is closest to a specific starting point to be formed as initial cycle. On each iteration it finds one unselected node that is closest to one of nodes in the cycle. The selected vertex is inserted between pair of nodes in the cycle considering the smallest saving value. Iteration is done until all vertices are selected.

This principle of insertion will be used to arrange gene on chromosome in crossover phase. It will benefit more than merely adding to last node in the route.

Considering the manner in each algorithm, it is designed to incorporate those algorithms to the crossover stage to ensure that the offspring's chromosomes have good fitness. Crossover operator using Nearest Neighbor algorithm named as CNN, Crossover operator using Sequential Insertion operator named as CSI. Those will be compared to original PMX operator. This research was only to compare the quality of operator and solution by different crossover operator, instead of to improve the solution quality.

For a discussion of each case, it is assumed that two chromosomes have been selected as parent (referred as K1 and K2):

a. Crossover operator using Nearest Neighbor algorithm (CNN)

1. Select randomly one of the first gene in each of K1 and K2, to be the first gene of offspring chromosome (referred as gene α)
2. Find next gene after gene α in each of K1 and K2 that is not selected as an offspring chromosome genes (referred as gene β and γ respectively). If there are no possible genes selected (e.g. because α is the last gene in the chromosome or all next genes have been selected), then get a gene that has not been selected in random manner (referred as gene δ).
3. a. When two genes are obtained (gene β and γ), select a gene which has closest distance from gene α (referred as gene δ). Add gene δ into the route right after gene α.
   b. When only one gene obtained (one of genes β, γ, δ) then add the gene into the route after gene α.
4. Set gene that just added into the route as as the initial gene of next iteration (as gene α)
5. Return to step 2 until all genes have been selected
6. Count its fitness

b. Crossover operator using Sequential Insertion algorithm (CSI)

1. Select randomly one of the first genes in each of K1 and K2, to be the first gene of offspring chromosome (referred as gene α), create an initial route α-α
2. Find next gene after gene α in each of K1 and K2 that is not selected as an offspring chromosome genes (referred as gene β and γ respectively). If there are no possible genes selected (e.g. because α is the last gene in the chromosome or all next genes have been selected), then get a gene that has not been selected in random manner (referred as gene δ).
3. a. When two genes are obtained (gene β and γ), select a gene which has closest distance from gene α (referred as gene δ). Insert gene δ into the route in position which gives smallest saving: min ($c\theta\delta + c\delta\lambda - c\theta\lambda$), θ and λ is a pair of successive genes in the route
   b. When only one gene obtained (one of genes β, γ, δ) then insert the gene into the route in position which gives smallest saving: min ($c\theta\delta + c\delta\lambda - c\theta\lambda$), θ and λ is a pair of successive genes in the route
4. Set gene that just added into the route as as the initial gene of next iteration (as gene α)
5. Return to step 2 until all genes have been selected
6. Count its fitness

CSI doesn't use all Nearest Insertion algorithm, it just use insertion principle. The only difference between CNN and CSI is just on step 3.a and 3.b (adding and inserting node to route). Both operators do not depend on the parents' structure; it introduces new edges to the offspring. By using specific treatment, the chances of producing a better offspring are more than PMX operator that rely on random-generated position.

Let us illustrate the algorithms through the example given as cost matrix in Figure 2. Let a pair of selected chromosomes be K1: (1, 4, 2, 6, 5, 3) and K2: (6, 1, 2, 4, 3, 5) with fitness 290 and 190 respectively.

	1	2	3	4	5	6
1	-	23	57	67	12	14
2	23	-	87	12	65	52
3	57	87	-	39	72	41
4	67	12	39	-	54	27
5	12	65	72	54	-	30
6	14	52	41	27	30	-

Figure 2. The cost matrix

a. CNN

α = 1 (random)

Iteration	**α**	**β**	**γ**	**δ**	**Route**	**Fitness**
1	1	4	2	2	1-2	23
2	2	6	4	4	1-2-4	23+12 = 35
3	4	-	3	3	1-2-4-3	35+39 = 74
4	3	-	5	5	1-2-4-3-5	74+72 = 146
5	5	-	-	6	1-2-4-3-5-6	146+30 = 176

Finally, the fitness is 176+14 = 190, as adding edge 6-1 to complete the tour. Step 3 could not find gene β because node 2 had been selected previously, step 4 could not find gene β because node 3 was the last node in K1, and in step 5, gene δ was generated randomly as node 3 in K1 is the last node and node 5 in K2 was previously selected.

b. CSI

α = 1 (random)

Iteration	**α**	**β**	**γ**	**δ**	**Route**	**Fitness**
1	1	4	2	2	1-2-1	46
2	2	6	4	4	1-4-2-1	102
3	4	-	3	3	1-3-4-2-1	131
4	3	-	5	5	1-5-3-4-2-1	158
5	5	-	-	6	1-5-6-3-4-2-1	157

The fitness is 157, much smaller than CNN result or fitness parent's chromosome. Sequence and cases of node selection in this case is similar to CNN; the difference is in route construction process.

B. Procedure of Testing

GA using CNN and CSI to solve TSP is implemented as a computer program, made by Delphi language. Experimental data is taken from TSPLIB dataset, namely gr21, berlin52, and eil21. Each dataset tested for the PMX, CNN, and CSI operator. For each dataset and crossover operators it takes 10 independent executions. The most optimal route and its length for each generation is noted, to count of its final average. Then, the average result of 3 operators compared to the most optimal result recorded in TSPLIB website.

The parameters used in each test:

- Number of generations: 50, 100
- Number of population: 100
- Probability of crossover (pc): 0.6
- Probability of mutation (pm): 0.1
- Survival selection: 0

Result and Discussion

The tests gives following result:

Table 1. Result of testing

Dataset	Optimal	Criteria	50 generations			100 generations		
			PMX	CNN	CSI	PMX	CNN	CSI
gr21	2707	Average	4416,07	3491,34	2710,42	3973,98	3329,20	2718,65
		Minimum	3906,98	3120,56	2707,00	3712,86	3054,82	2707
		Maximum	4814,22	3800,56	2736,66	4265,45	3694,45	2773,37
berlin52	7542	Average	20736,38	14162,47	7954,34	18577,54	12471,62	7999,23
		Minimum	18806,45	13097,68	7730,92	17480,53	11438,46	7825,18
		Maximum	22295,99	15106,77	8145,75	19664,22	13828,14	8153,76
eil76	538	Average	1892,85	1173,74	579,58	1702,10	1010,42	582,24
		Minimum	1811,53	1083,89	570,71	1617,69	927,48	573,77
		Maximum	1969,54	1254,13	587,74	1796,18	1086,20	590,02

Observations were made in 10 independent executions (10 independent runs). Each execution was carried out in 50 and 100 iterations/generations, in each generation the best route is noted. The best route is calculated from the shortest, longest, and its average, as listed in Table 1.

It appears that in general that the crossover operator combined with Sequential Insertion algorithm provides better results than the other operators, while the PMX operator gives the worst results. However, average length of the cycle in all operators is above optimal solution known, with slight difference to optimal on CSI operator.

Figure 3 to 8 show performance graphic of different crossover operator for every dataset and every generations.

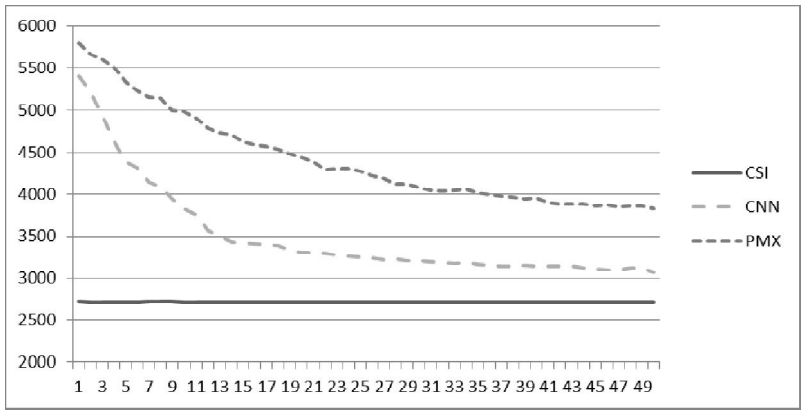

Figure 3. Performance of operators for gr21 case on 50 generations

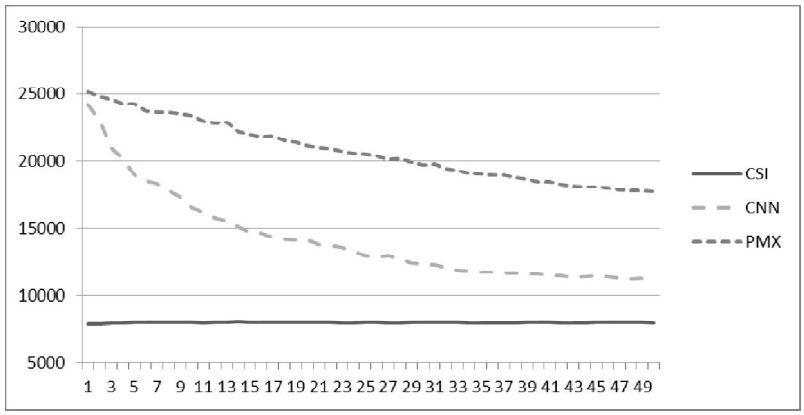

Figure 4. Performance of operators for berlin52 case on 50 generations

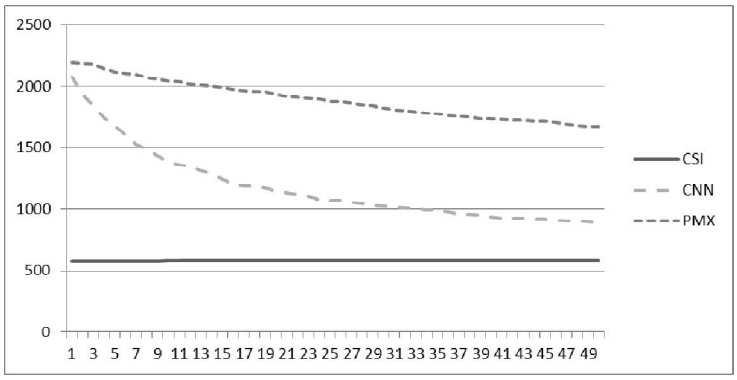

Figure 5. Performance of operators for eil76 case on 50 generations

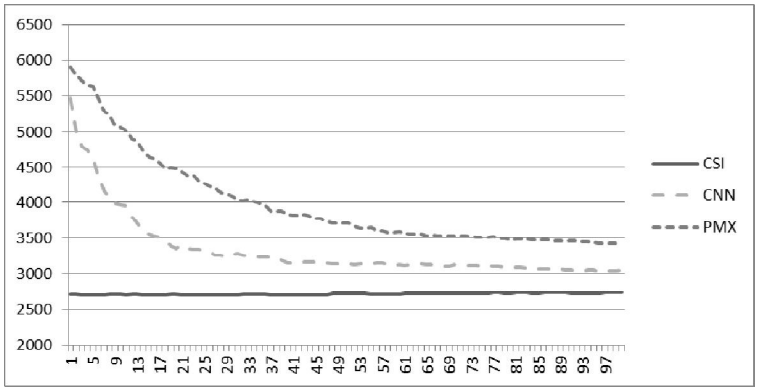

Figure 6. Performance of operators for gr21 case on 100 generations

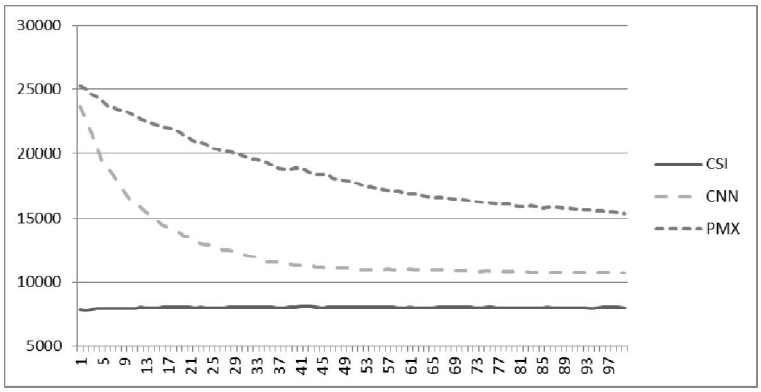

Figure 7. Performance of operators for berlin52 case on 100 generations

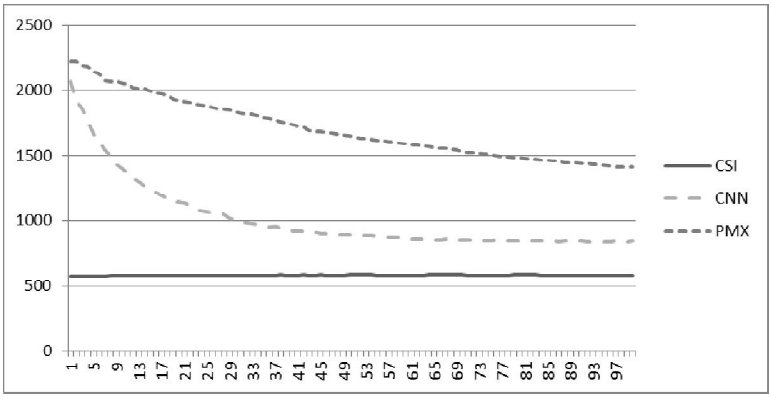

Figure 8. Performance of operators for eil76 case on 100 generations

For all the cases observed, the crossover operator combined with Sequential Insertion (CSI) gives stable results. The two other operators tend to lead to optimal results, with faster rate in the crossover operator combined with Nearest Neighbor (CNN). This shows that PMX works better on longer generations.

CSI gives stable results because since the early generation, position of the insertion has been taken into account to generate the shortest possible route, which is based on the calculation of saving. The insertion of a node in between a pair of nodes should provide a shorter distance, or give maximum saving (maximum saving means least difference between $c_{ik} + c_{kj}$ and c_{ij}).

Result of CNN operator tends to lead to optimal solution on each generation although there is still a significant difference with the results of CSI operator. This is because the node added to the route is the nearest node to the last node on the route, which is not necessarily shorter overall.

The slope of the graph for CNN operator is steeper than the graph for PMX operator in 50 and 100 generations testing; however they both lead to an optimal state. Basically each operator has relatively equal slope on all datasets tested. CNN operator has faster rate to get into convergence state than PMX operator in all testing.

At the testing, found that the CNN and PMX crossover operator lead to optimal but still has significant difference. All of all, for 100 generations average result of CNN and PMX is better than 50 generations; the value is smaller. Meanwhile for CSI the result for 100 generations slightly decrease. This can be inferred from Table 1.

Conclusion and Suggestion

We have proposed two crossover operator for genetic algorithm for the Traveling Salesman Problem (TSP) that use Nearest Neighbor and Sequential Insertion principles, named Crossover operator using Nearest Neighbor algorithm (CNN) and Crossover operator using Sequential Insertion algorithm (CSI) , respectively. Comparative study among them and original PMX operator was performed for three benchmark TSPLIB instances. Some conclusions are:

1. Arrangement ("intervention") in route construction of the crossover stage proved to give better result than in the absence of arrangement.
2. The arrangement in the route construction required to ensure optimal results, as well as minimizing the number of generations needed to achieve optimal result.
3. CSI proven to give better result than CNN and PMX operators in data instances tested. CSI could solve exactly instance gr21, at least in one of 10 runs, while not for the other instances. CNN and PMX could not give exact solution for all instances; furthermore they gave worse solution than the optimal result known.
4. Incorporating heuristic into crossover stage of GA improves overall performance of GA.

Some improvements or future work to be done are:

1. The operator should be tested for more various data instances,
2. Improvement in solution quality
3. Research for optimal parameter, such as number of populations and number of generations. Also, value of crossover probability and mutation probability could be taken into consideration, as Rexhepi, Maxhuni, and Dika (2013) stated that there were a correlation between number of population and probability of mutation, and correlation between number initial population and final result.

Bibliography

Ahmed, Z.H. (2010). Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator. *International Journal of Biometrics & Bioinformatics*, 3(6), 96-105. Retrieved from http://www.cscjournals.org/library/manuscriptinfo.php?mc=IJBB-41

Al Rahedi, N.T, and Atoum, J. (2009). Solving the Traveling Salesman Problem Using New Operators in Genetic Algorithms. *American Journal of Applied Sciences*, 6(8): 1586-1590, 2009.

Alatartsev, S., Augustine, M., and Ortmeier, F. (2013). Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods. *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*. Retrieved from https://cse.cs.ovgu.de/cse/administrator/components/com_jresearch/files/publications/ICAPS2013.pdf

Dr´eo, J. (2006). *Metaheuristics for Hard Optimization*. Berlin: Springer-Verlag

Gutin, G., Punnen, A.P. (2002). *The Traveling Salesman Problem and Its Variations*. Berlin: Springer Science & Business Media. Retrieved from https://books.google.co.id/books?id=JBK_BAAAQBAJ

Michalewics, Z. (1996). *Genetic Algorithms + Data Structure = Evolution Programs (3rd edition)*. Berlin: Springer-Verlag

Philip, A., Taofiki, A.A., Kehinde, O. (2011). A Genetic Algorithm for Solving Travelling Salesman Problem. *International Journal of Advanced Computer Science and Applications*, 2(1), 26-29. doi: 10.14569/IJACSA.2011.020104

Poot, A., Kant, G., and Wagelmans, A. P. M. (2002). A Savings Based Method for Real-Life Vehicle Routing Problems. *The Journal of the Operational Research Society*, 53(1), 57-68. Retrieved from http://www2.eur.nl/WebDOC/doc/econometrie/feweco19991013123552.ps.

Rexhepi, A., Maxhuni, A., and Dika, A. (2013). Analysis of the impact of parameters values on the Genetic Algorithm for TSP. International Journal of Computer Science Issues, 10(1-3), 158-164. Retrieved from http://www.ijcsi.org/contents.php?volume=10&&issue=1

Rosen, K.H, et.al. (2000). *Handbook of Discrete and Combinatorial Mathematics*. Florida: CRC Press

Sureja, N.M. and Chawda, B.V. (2012). Random Travelling Salesman Problem using SA. *International Journal of Emerging Technology and Advanced Engineering*, 2(4), 621-624. Retrieved from http:// www.ijetae.com/files/Volume2Issue4/

Woeginger, G.J. (2003). Exact Algorithms for NP-hard Problems: A Survey, In M. Juenger, G. Reinelt and G. Rinaldi (Eds.), *"Combinatorial Optimization - Eureka! You shrink!"* (pp 185-207). Berlin: Springer.