New crossover operators for Real Coded Genetic Algorithm (RCGA)

Gurjot Singh
Indian Institute of Technology Jodhpur
Jodhpur, India
Email: gurjot@iitj.ac.in

Neeraj Gupta and Mahdi Khosravy
University of Information Science & Technology
Ohrid, Macedonia
Email: (neeraj.gupta,mahdi.khosravy)@uist.edu.mk

***Abstract*— This paper aims at achieving global optimal solution of complex problems, such as traveling salesman problem (TSP), using extended version of real coded genetic algorithms (RCGA). Since genetic algorithm (GA) consists of several genetic operators, namely selection procedure, crossover, and mutation operators, that offers the choice to be modified in order to improve the performance for particular implementation, we propose three new crossover techniques for Real Coded Genetic Algorithms, which will improve the quality of solution as well as the rate of convergence to the optimum solution. Methods proposed for crossover operators are inspired by asexual reproduction commonly observed in nature. In this regard, new crossover techniques proposed incorporates the concept of Boltzmann's distribution (BD) for escaping local optima by allowing hill-climbing moves and Metropolis Algorithm (MPA), where, survival of offspring is tested before transit to new generation. Finally, these three methods are compared on various aspects like rate of convergence and quality of final solution among each other and against other randomized algorithms. (*Abstract*)**

Keywords—Genetic algorithm, traveling salesman problem, crossover, Simulated annealing (key words)

I. Introduction

Genetic Algorithms are non-traditional computerized search and optimization techniques which are based on the real life concepts of natural genetics and natural selection. It is well known that Genetic Algorithm (GA) is one of the best optimization technique to solve non-linear non-continuous optimization problems (1). In the past, various computationally expensive problems have been successfully solved using Genetic Algorithms (2). Although the original idea of Genetic Algorithms proposed coding the information in binary values ($0s$ and $1s$), over the last few years, lot of emphasis has been given to Real Coded Genetic Algorithms where the information is coded in real numbers depending upon the problem statement. This is because original binary coded genetic algorithm is not well suited for fine tuned searching in complex search space (3)(4).

Generally genetic algorithms incorporate three mechanisms to fine-tune the search space, namely selection, crossover, and mutation. The selection mechanism determines which individuals are chosen for mating (reproduction) and typically selects fitter individuals over others according to the survival of fittest strategy which says- the better is an individual; the higher is its chance of being parent. (5) Crossover and mutation mechanisms primarily explore the search space, whereas selection reduces the search area within the population by discarding poor solutions. However, worst individuals should not be discarded straight-away and they must have some chances to be selected, since it may lead to better direction afterwards. A good search technique must find a good trade-off between exploration and exploitation in order to find a global optimum. (6)

In this paper, a variant of Real Coded Genetic Algorithm (RCGA) is proposed based on the idea of *asexual reproduction*. In asexual reproduction, offspring(s) is/are generated from a single parent. Nature has many example of this mode of reproduction like liverworts, starfish, etc. (7) Here, characteristics of the offspring(s) are determined only from the transformed heredity of single parent. In context of our algorithm, this escapes the RCGA to strike with similar resulting chromosomes in later stages in population, which reduces efficiency of RCGA for multi-modal objective functions. Asexual reproduction based RCGA generates diverse chromosomes in population until the end of all iterations.

Proposed RCGA with crossover operator mimicking asexual reproduction method is tested to solve Traveling Salesman Problem (TSP), which is an NP-hard combinatorial optimization problem and is used as a benchmark for many optimization methods. The aim of Traveling Salesman Problem is to find the optimum route such that the distance traveled by the salesman is minimum while traversing all the cities/places exactly once. (8)

1) Problem Statement: The variant of Traveling Salesman Problem we considered in this paper is to find the optimum route traveled while traveling from the national capital of India (New Delhi) to each of the 28 state capitals (shown on map of India in figure 1) and then to return back to national capital. It is practical version of TSP, where we have accurate data.

If we have to solve problem I-1 tediously, we have to check all the 8.84×10^{30} possibilities, which will take 280 billion years on a machine with an ablity to perform 1 trillion computations every second. This is indeed computationally very expensive. To solve problem I-1, we use substring swapping method, aspired by asexual reproduction, in which optimum route is evolved by the process of self-crossover. In this approach, offspring is generated without the fusion of gametes (Agamogenesis). In second and third method, the survival of chromosomes is

978-1-4799-58562-3/15/$31.00 ©2015 IEEE

modified using the concept of Boltzmann's Distribution (BD) based Metropolis Algorithm (MA), which is one of the best

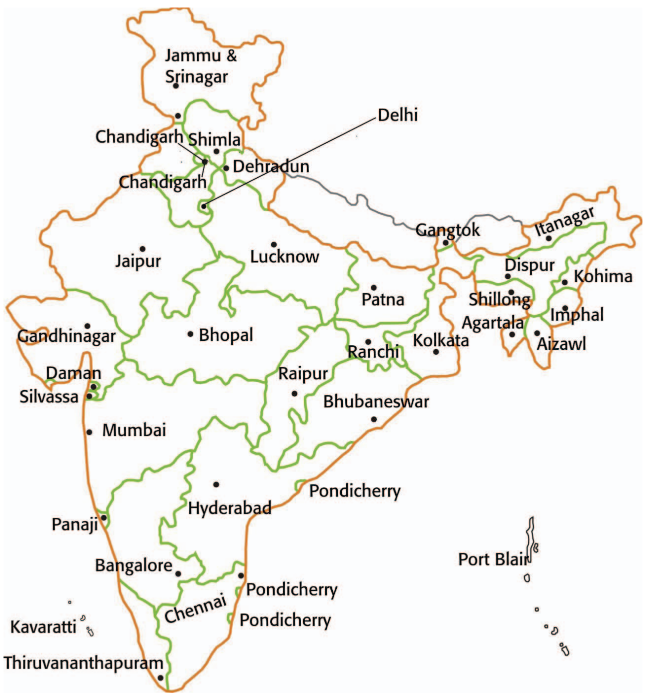

Figure 1. State capitals on the map of India

algorithm for natural selection. According to the algorithm, strong offspring becomes the part of the next generation. Some weak offspring may also survive along with best offspring, when provided some survival probability. This also introduces Hill Climbing (HC) search procedure in RCGA to handle multi-modal objective functions very efficiently.

II. Algorithm

Algorithm starts with the generation of an initial random population (showing random tours) consisting of sequences of real coded distinct integer values. Here, each integer value denotes particular node/city and each chromosome represents a valid route. We define a route as a valid route if it satisfies the following constraint:

1. Starting and ending destinations are the same.
2. Every other city is traversed exactly once while following the route.

Accordingly, we can formulate problem I-1 as a linear programming problem as follows:

$$\min(z) = \min(|x_1 - x_0| + |x_2 - x_1| + \cdots + |x_n - x_{n-1}|)$$

such that,

$$x_0 = x_n \tag{1}$$

$$x_1 \neq x_2 \neq \cdots \neq x_{n-1} \tag{2}$$

The objective function of this linear programming problem is minimize the value of z, which is total distance travelled, while traversing all the cities satisfying the constraints.

First constraint (1) comes from the fact that starting and ending destination have been specified (New Delhi) and is fixed. Initially, conventional *roulette wheel selection* procedure is adopted as in case of binary coded genetic algorithms. (1) For each chromosome, objective function, $f(x)$, calculates the total distance covered while traversing the route suggested by sequence of nodes in the chromosomes. Since in traveling salesman problem, distance traveled has to be minimized, thus it is convenient to write the fitness function as $F(x)$ as

$$F(x) = \frac{1}{1 + f(x)}$$

where $f(x)$ is objective function defined above.

Here, chromosome with lesser traveling distance has larger fitness value and hence more chances of survival for the next generation. Thereafter, conventional crossover procedures (like 1 or 2-point crossover, cut and splice crossover) are used in case of binary coded genetic algorithms. But in this problem or for real coded genetic algorithms in general, above crossover operators cannot be applied since each chromosome represents different route and crossing over using 1 or 2-point crossover may violate second constraint (equation 2) of the problem. These crossover operators are replaced by one of three proposed efficient techniques as given below to escape from all local optimal minimum solution and to increase the rate of convergence. After using crossover technique, uniform mutation (with some mutation probability p_m) is done to create further diversity in the n −dimensional search space.

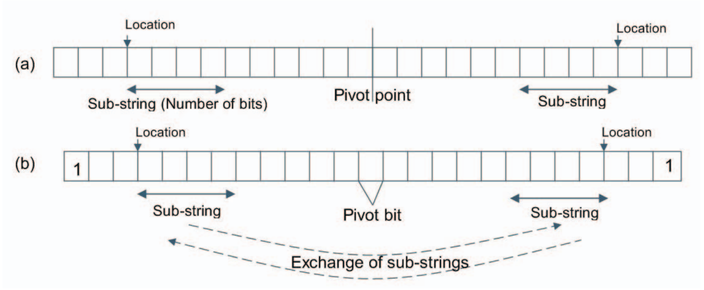

Figure 2. Simple example of substring swapping crossover technique.

A. Simple Substring swapping crossover (SSSC)

This procedure modifies the chromosomes, before it goes into next generation. In this technique, we choose a bit somewhere in the middle as a pivot as shown in Figure 2(a). Then, from the right half of the pivot bit, a substring of bits (representing set of cities) is chosen at random location and is swapped/exchanged with the substring of exact same length in left half part (keeping start and end cities fixed). This swapping satisfies both the constraints (equation 1 and 2), as even after changing the order of cities, no city is visited twice and every city is visited exactly once (except the starting destination). Here, two parameter locations (in the right half, it is from right most bit and vice versa for the left side of the pivot) and length of substring (number of bits) are chosen randomly. Multi-substring swapping can be employed at different locations where neither bit should be overlapped, for introducing more diversity in the sample space. Figure 3 shows the flowchart of the algorithm.

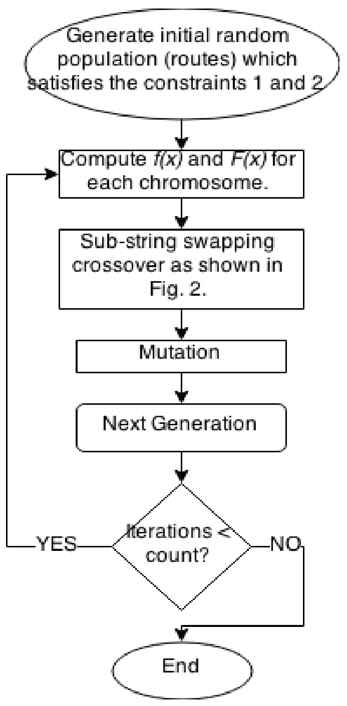
Generate initial random population (routes) which satisfies the constraints 1 and 2
Compute f(x) and F(x) for each chromosome.
Sub-string swapping crossover as shown in Fig. 2.
Mutation
Next Generation
Iterations < count?
YES
NO
End

Figure 3. Flowchart describing Simple Substring Swapping Crossover (SSSC) Algorithm

B. *Elite Substring Swapping Crossover (ESSC)*

Above crossover procedure does not necessarily converge at later stages. This is because there is no filtration of chromosomes happening in previous approach. Therefore, the algorithm is further extended using idea of *strict elimination*, where elimination of either parent or offspring is carried out based on their performance; thus, referred as Elite substring crossover technique. In this approach, computed fitness value $F(x)$ of offspring is compared with parent. If the fitness value of the offspring is higher than its parent, then this offspring is accepted over its parent to transit in the next generation, otherwise it is rejected. As we will see in section III, this algorithm with the help of concept of natural selection, outperforms the earlier method both in terms of quality of solution and the rate of convergence. But many a times, it converges to local minima. Figure 4 shows the flowchart of the algorithm.

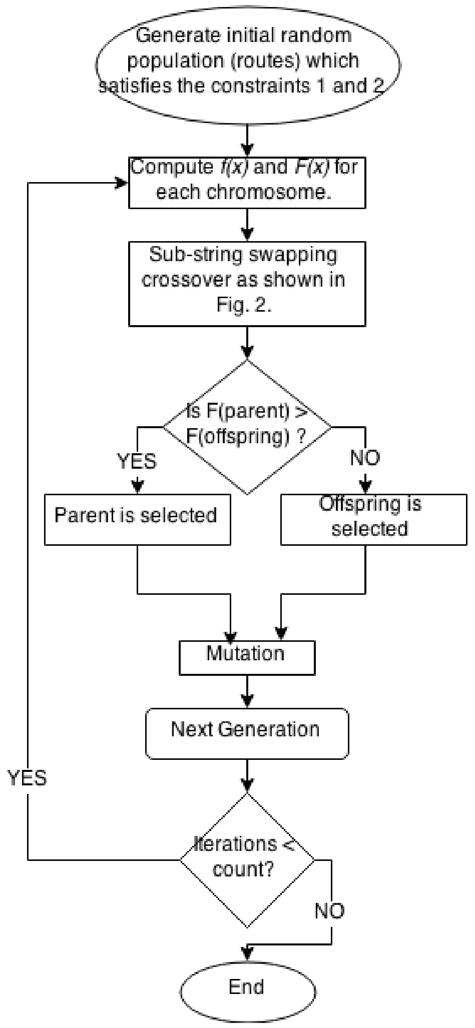
Generate initial random population (routes) which satisfies the constraints 1 and 2
Compute f(x) and F(x) for each chromosome.
Sub-string swapping crossover as shown in Fig. 2.
Is F(parent) > F(offspring) ?
YES
NO
Parent is selected
Offspring is selected
Mutation
Next Generation
YES
Iterations < count?
NO
End

Figure 4. Flowchart describing Elite Substring Swapping Crossover (ESSC) Algorithm.

C. *Hybrid Substring Swapping Crossover (HSSC)*

To improve the quality of solution even further, both in terms of convergence and achieving global optima, simulated annealing approach comes handy. This technique ensures diversity initially and convergence at later stages and is referred to as *Hybrid Substring Swapping Crossover* technique in this paper. In this approach, transit of offspring with higher fitness value follows elite substring swapping approach, but the rejection of worse offspring is decided on the basis of the rejection probability p_n. For this procedure, Metropolis algorithm is adopted. In this approach, normally distributed random probability (p) is compared to the value of probability p_n, given by Boltzmann's distribution.

$$p_n = e^{\frac{-y}{kT}}$$

If $p < e^{\frac{-y}{kT}}$, the offspring gets accepted, otherwise it is rejected. Here T refers to the temperature. Initial temperature is randomly chosen (usually very high temperature results in more diverse search). This temperature becomes half of its previous value when there is no change in the objective function value, until convergence is achieved.

$$T_n = \frac{T_{n-1}}{2}$$

As T becomes smaller and smaller with each iteration, rejection probability becomes higher and higher and eventually leads to Elite Substring Swapping approach. It should also the noted that any other heuristics (constantly decreasing) can be chosen for the given purpose.

Figure 5 shows the flowchart of the algorithm.

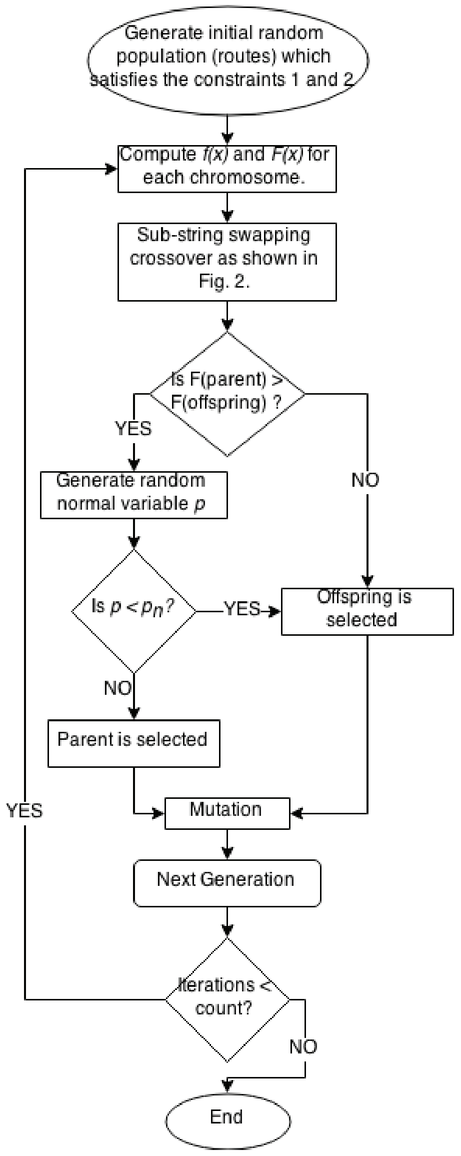
Generate initial random population (routes) which satisfies the constraints 1 and 2
Compute f(x) and F(x) for each chromosome.
Sub-string swapping crossover as shown in Fig. 2.
Is F(parent) > F(offspring) ?
YES
NO
Generate random normal variable p
Is p < pn?
YES
Offspring is selected
NO
Parent is selected
Mutation
Next Generation
YES
Iterations < count?
NO
End

Figure 5. Flowchart describing Hybrid Substring Swapping Crossover (HSSC) Algorithm.

D. Mutation

After crossover step, with some probability (p_m), we mutate the chromosome. We exchange a random bit in the chromosome with another bit at random location in the chromosome. This ensures hill climbing moves to avoid convergence at plateaus or local optima.

III. Results

We compared the results of traveling salesman problem given in I-1 for three modified RCGA procedures (having different crossover methods as described above, in which traveler's starting and ending destination is New Delhi, and he traverses each of Indian states capital exactly once except New Delhi. Fig. 6 to 8 show the plots of Maximum and average fitness value versus number of iterations for different algorithms.

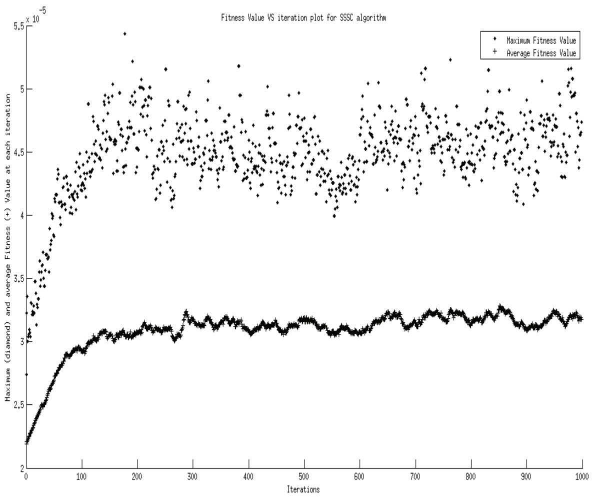

Figure 6. Maximum (diamond) and average fitness value (+) for Simple Substring Swapping Crossover (SSSC) algorithm.

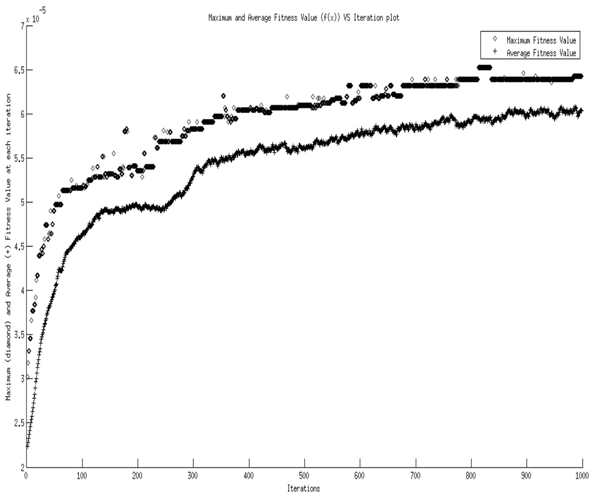

Figure 7. Maximum (diamond) and average fitness value (+) for Elite Substring Swapping Crossover (ESSC) algorithm.

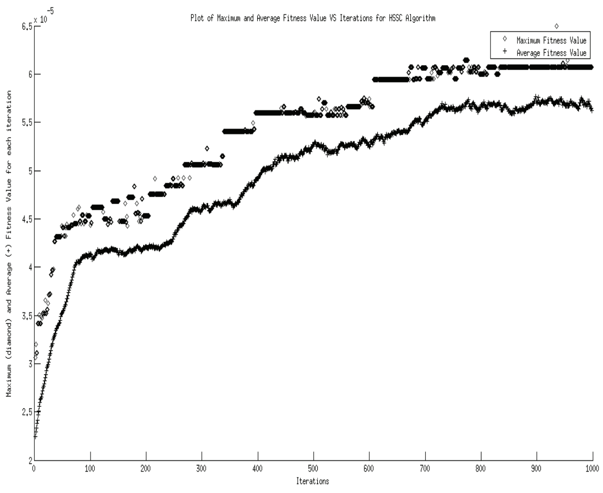

Figure 8. Maximum (diamond) and average fitness value (+) Hybrid Substring Swapping Crossover (HSSC) algorithm.

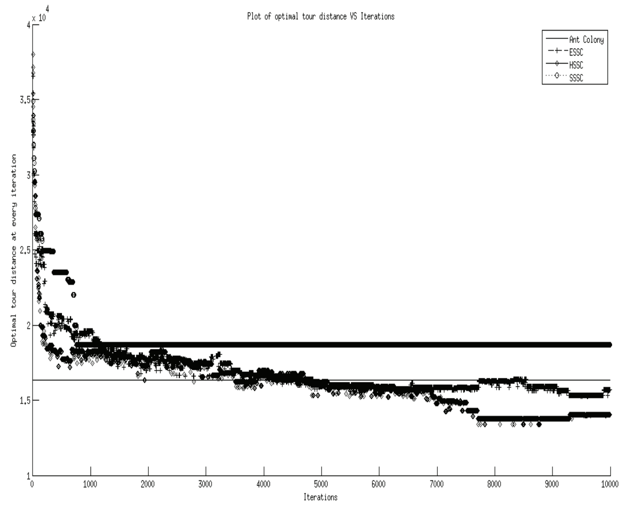

Figure 9. Performance of all the methods on the same problem. (-) shows the output of Ant Colony method. (o) shows performance of SSSC algorithm. (+) shows performance of ESSC algorithm while diamond symbol shows output of HSSC algorithm.

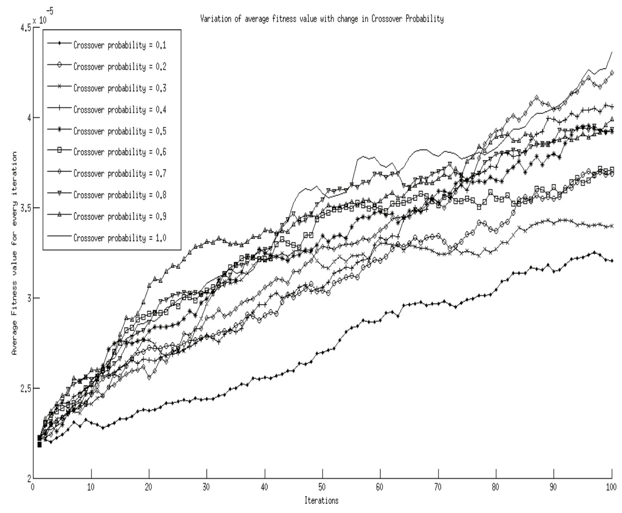

Figure 10. Variation of average fitness value of each iteration with change in crossover probability for HSSC algorithm.

IV. COMPARISON WITH EXISTING ALGORITHMS

In previous section, we have compared our three algorithms on the basis of convergence rate and quality of solution these algorithms offered for the problem I-1. In this section, we will compare the performance of our algorithm with existing approximation and randomized algorithms like ant colony optimization. (9)

A. *Comparison with greedy algorithm (Nearest Neighbor algorithm)*

One other approach to solve traveling salesman problem is to use Greedy Algorithm. Greedy algorithm chooses the locally optimum choice available at each stage in hope of producing globally optimum solution. Nearest Neighbor algorithm is a greedy algorithm that lets the salesman choose nearest unvisited city as the next destination. When we solved the problem I-1, the solution given by NN algorithm outputs 21490 Kilometers which is nearly twice the distance given by our algorithm. Clearly our algorithm outperforms NN algorithm.

B. *Comparison with Ant Colony Otimization*

When we solved the same problem using Ant Colony optimization, the algorithm output the local maxima which was approximately 1.3 times our minimum distance and got stuck in local minima in very early stages of optimization. Figure 9 shows the plot of optimal tours vs iteration count for the ant colony optimization method and three of our algorithms for one instance. Ant Colony method was able to outperform SSSC algorithm in few runs but is almost surely outperformed by ESSC and HSSC algorithms.

V. CONCLUSIONS

From Fig. 6 to 8, it has been noted that HSSC algorithm outperforms both SSSC algorithm and ESSC algorithms over large number of iterations (~104 iterations). HSSC selects better chromosomes (of more diverse nature) to avoid local minimum in multi-modal objective functions. However, ESSC converges much faster than both hybrid and substring swapping algorithm, but, converging towards local minimum many a times.

Increasing crossover probability (in the range of $0.9 - 0.95$) improves the solution with more diversity in n −dimensional search space. Number of chromosomes in the population improves the optimal solution directly. Figure 10 shows the variation of average fitness value of each iteration with change in crossover probability for HSSC algorithm. It can

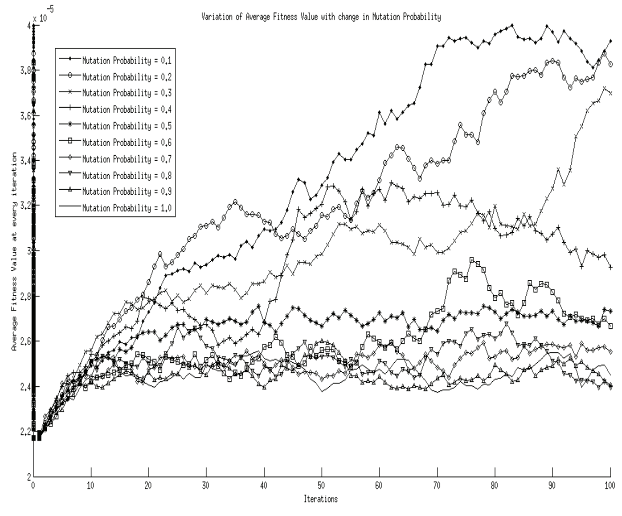

Figure 11. Variation of average fitness value of each iteration with change in mutation probability for HSSC algorithm.

be observed from the figure that as we increase the crossover probability i.e. increasing chances for crossover operation to occur, quality of the solution improves.

However, when we increased the mutation probability, it turned out that the quality of solution decreases with increase in mutation probability. Figure 11 shows the variation of average

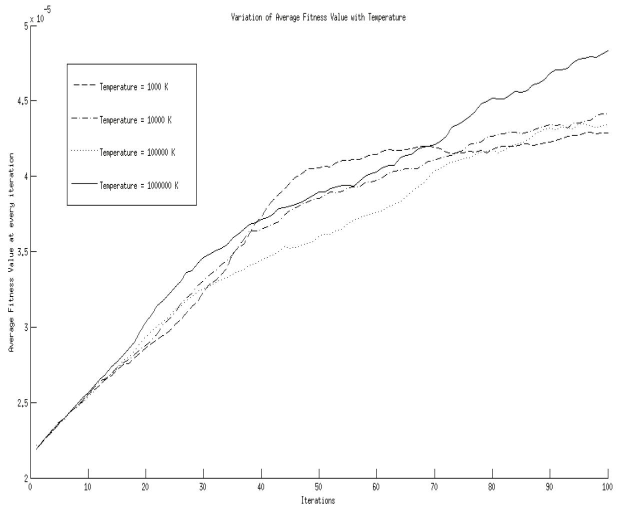

Figure 12. Variation of average fitness value of each iteration with change in Temperature for HSSC algorithm.

fitness value of each iteration with change in mutation probability for HSSC algorithm.

It must also be noted that higher initial temperature in HSSC makes search space more robust initially, thus, reduces the chances of convergence to a local minimum. Figure 12 shows the graphical proof of the above statement. Here, we can see that more robust search (one with higher initial temperature) significantly outperforms the less robust search (one with lower initial temperature). Finally, we have observed that the number of iterations improves performance.

REFERENCES

[1] K. Deb, "Optimization for Engineering Design: Algorithms and Examples", Prentice-Hall, India, 2004.

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[3] L. Davis, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[4] D. Goldberg and S. Voessner, "Proceedings of the Genetic and Evolutionary Computation Conference 99", edited by e. a. In Banzhaf, W., Morgan Kaufmann, San Mateo, California, 1999, pp. 220–228.

[5] M. R. Noraini and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP", 2011.

[6] D. Beasley, D. R. Bully, and R. R. Martinz, "An overview of genetic algorithms: Part 1. Fundamentals."

[7] C. H. Edmondson, "Autotomy and regeneration in Hawaiian starfishes" (The Museum, 1935), No. 8.

[8] R. Johnson and M. G. Pilcher, "The traveling salesman problem", edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B Shmoys, John Wiley Sons, Chichester, 1985, 463 pp (Wiley Subscription Services, Inc., A Wiley Company, 1988), No. 3, pp. 253–254.

[9] M. Young, H. Behravan, "Ant Colony Optimization Techniques on Travelling Salesman Problem", url: http://www.mathworks.com/matlabcentral/fileexchange/35506-ant-colony-optimization-techniques-applied-on-travelling-salesman-problem/content/Codes/randint.m, 2012, [Online; accessed 25-December-2014].