# New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering

Shubhra Sankar Ray, Sanghamitra Bandyopadhyay, and Sankar K. Pal

Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108
{shubhra_r, sanghami, sankar}@isical.ac.in
http://www.isical.ac.in/∼shubhra_r

**Abstract.** This paper deals with some new operators of genetic algorithms for solving the traveling salesman problem (TSP). These include a new operator called, "nearest fragment operator" based on the concept of nearest neighbor heuristic, and a modified version of order crossover operator. Superiority of these operators has been established on different benchmark data sets for symmetric TSP. Finally, the application of TSP with these operators to gene ordering from microarray data has been demonstrated.

## 1 Introduction

The Traveling Salesman Problem (TSP) has been used as one of the most important test-beds for new combinatorial optimization methods [1]. Its importance stems from the fact there is a plethora of fields in which it finds applications e.g., scheduling, vehicle routing, VLSI layout, microarray gene ordering and DNA fragment assembly. Over decades, researchers have suggested a multitude of heuristic algorithms, including genetic algorithms (GAs) [1, 2, 3]for solving TSP. The classical formulation of TSP is stated as: Given a finite set of cities and the cost of traveling from city i to city j, if a traveling salesman was to visit each city exactly once and then return to the home city, which tour would incur the minimum cost?

Let $1, 2, \cdots, n$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city $i$ to city $j$. The Traveling Salesman Problem (TSP) is the problem of finding the shortest closed route among $n$ cities, having as input the complete distance matrix among all cities. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1} \tag{1}$$

The objective is to find a permutation of the $n$ cities, which has minimum cost.

The TSP, with some minor modifications, can be used to model the microarray gene ordering (MGO) problem. An optimal gene order provides a sequence of genes where the genes those are functionally related and similar are nearer in the ordering [4]. This functional relationship among genes is determined by

S.K. Pal et al. (Eds.): PReMI 2005, LNCS 3776, pp. 605–610, 2005.

gene expression levels from microarray by performing biological experiments [4]. Similarity between genes can be measured with Euclidean distance, Pearson correlation, absolute correlation, Spearman rank correlation, etc.

Two new genetic operators are proposed in this article for solving TSP. Consequently, the application of these operators are demonstrated for solving microarray gene ordering (MGO) problem efficiently.

## 2 Relevance of TSP in Microarray Gene Ordering

An optimal gene order, a minimum sum of distances between pairs of adjacent genes in a linear ordering $1, 2, \cdots, n$, can be formulated as [5]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{2}$$

where $n$ is the number of genes and $C_{i,i+1}$ is the distance between two genes $i$ and $i+1$. In this study, the Euclidean distance is used to specify the distance $C_{i,i+1}$.

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of $k$ experiments. The Euclidean distance between $X$ and $Y$ is

$$C_{x,y} = \sqrt{\{x_1 - y_1\}^2 + \{x_2 - y_2\}^2 + \cdots + \{x_k - y_k\}^2}. \tag{3}$$

One can thus construct a matrix of inter-gene distances, which serves as a knowledge-base for mining gene order using GA. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized. This is analogous to the traveling salesman problem.

## 3 GA with New Operators for TSP

In this section, two new operators of GAs for solving TSP are described. These are nearest fragment (NF) and modified order crossover (MOC). The genetic algorithm designed using these operators is referred to as FRAG_GA. The structure of the proposed FRAG_GA is provided in Fig. 1. Here path representation [1], linear normalized selection and elitism operators are utilized [2]. For TSP, simple inversion mutation (SIM) [1] is employed.

### 3.1 Nearest Fragment (NF) Heuristic

The nearest-neighbor (NN) heuristic for creating initial population, have the advantage that they only contain a few severe mistakes, while there are long segments connecting nodes with short edges. Therefore such tours can serve as good starting tours. In NN the main disadvantage is that, several cities are

```
begin FRAG_GA
    Create initial population with Nearest-Neighbor Heuristic
    while generation_count < k do
        /* k = max. number of generations. */
        begin
            Apply NF heuristic
            Linear normalized selection
            MOC
            Mutation
            Elitism
            Increment generation_count
        end
        Output the best individual found
end FRAG_GA
```

**Fig. 1.** The Pseudo-code for FRAG_GA

not considered during the course of the algorithm and have to be inserted at high costs in the end. This leads to severe mistakes in path construction. To overcome the disadvantages of the NN heuristics, we propose a new heuristic operator, called the Nearest Fragment (NF) operator, which is used in every generation (iteration) of GA with a predefined probability for every chromosome in the population as a subsequent tour improvement method. In this process, each string (chromosome in GA) is randomly sliced in *frag* fragments. The value of *frag* is chosen in terms of the total number of cities ($n$) for a particular TSP instance. For tour construction the first fragment is chosen randomly. From the last city of that fragment, the nearest city that is either a start or an end point of a not yet visited tour fragment is determined from the cost matrix. The fragment containing the nearest city is connected to the selected fragment, with or without inversion depending on whether the nearest city is the last city of a fragment or not respectively. The process is repeated until all fragments have been reconnected.

### 3.2 Modified Order Crossover (MOC)

Order crossover [6] has been observed to be one of the best in terms of quality and speed, and yet is simple to implement for solving TSP using GA [1, 2, 3]. In order crossover the length of a substring is chosen randomly. Thus on an average, the length is equal to $n/2$. This can lead to a marked increase in the computational time, which can be reduced if the length of the substring for performing crossover can be fixed to a small value. However, no study has been reported in the literature for determining an appropriate value of the length of a substring for performing order crossover. Such an attempt is made in this article where it is found that a substring length $'y'$ for MOC provides good results for TSP if $y = \max\{2, \alpha\}$, where $n/9 \leq \alpha \leq n/7$ ($n$ is the total number of cities). Unlike order crossover, where the substring length is randomly chosen, in MOC it is predefined at $y$. For example, for a 10 city problem the value of $\alpha$

is predefined at 1.25, therefore $y = 2$. The rest of the process in MOC is same as order crossover.

## 4 Experimental Results

FRAG_GA is implemented in Matlab 5.2 on Pentium-4 (1.7 GHz). The experiment has two parts. In the first part we have compared FRAG_GA with SWAP_GATSP [3], and OX_SIM (standard GA with order crossover and simple inversion mutation) [1] for solving benchmark TSP instances like Grtschels24, kroA100, d198, ts225, pcb442 and rat783 [7]. In the second part for biological microarray gene ordering, Cell Cycle cdc15, Cell Cycle and Yeast Complexes datasets are chosen [8]. The three data sets consists of about 782, 803 and 979 genes respectively, which are cell cycle regulated in Saccharomyces cerevisiae, with different number of experiments (24, 59 and 79 respectively) [4]. Each dataset is classified into five groups termed G1, S, S/G2, G2/M, and M/G1 by Spellman et. al. [4]. Throughout the experiments the population size is taken to be 10 for smaller problems (<100 cities/genes), while for larger problems ($\geq$100 cities/genes) this is set equal to 20. Crossover probability is fixed at 0.85 and mutation probability is fixed at 0.015 across the generations.

For the nearest fragment (NF) operator, each string (chromosome in GA) is randomly sliced in *frag* fragments, where $frag \cong n/8$. Probability of NF operator was set to be 0.4 for n greater than 100 and 0.5, otherwise. The value of substring length for the modified order crossover operator (MOC) is kept in the range $n/7$ to $n/9$. All these values were obtained after extensive experiments, which are omitted here for the lack of space.

Table 1 summarizes the best results and average results obtained by running the FRAG_GA, SWAP_GATSP and OX_SIM on the aforesaid TSP instances.

**Table 1.** Cost values using FRAG_GA, SWAP_GATSP and OX_SIM for different TSP instances

| | | Best Results | | | Average Results | | |
|---|---|---|---|---|---|---|---|
| Problem | Optimal | FRAG_GA | SWAP_ GATSP | OX_SIM | FRAG_GA | SWAP_ GATSP | OX_SIM |
| Grtschels 24 | 1272 | 1272 (130) | 1272 (500) | 1272 (8,000) | 1272 (1000) | 1272 (2000) | 1322 (15000) |
| KroA 100 | 21282 | 21282 (800) | 21504 (5000) | 22,400 (25,000) | 21,350 (2000) | 21,900 (5000) | 22670 (30000) |
| d198 | 15780 | 15834 (3000) | 15992 (7000) | 16,720 (25,000) | 15964 (3500) | 16,132 (10000) | 18200 (40000) |
| Ts 225 | 126643 | 126730 (3000) | 127012 (7000) | 135800 (25,000) | 126890 (3500) | 128532 (10000) | 138283 (40000) |
| Pcb 442 | 50778 | 51104 (8000) | 52620 (15000) | 53402 (40,000) | 51930 (10000) | 53,820 (20000) | 59740 (65000) |
| Rat 783 | 8806 | 9007 (15000) | 9732 (30000) | 10810 (70,000) | 9442 (20000) | 10110 (40000) | 11520 (100000) |

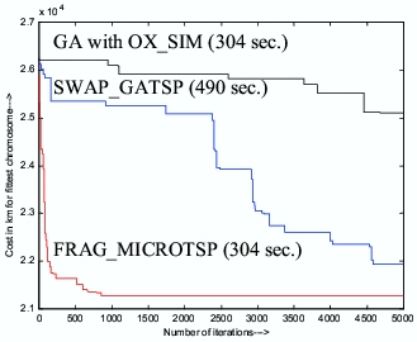


**Fig. 2.** Variation of cost of the best string with number of iteration for kroa100.tsp

For each problem the iteration in which the result is obtained is mentioned in columns 3-8 within parentheses. In SWAP_GATSP and OX_SIM the number of populations is taken to be 10 for 24 and 29 cities, 24 for 48 and 51 cities, 30 for 70 and 76 cities, and 40 for number of cities greater than or equal to 100 [3]. As can be seen from Table 1 FRAG_GA is superior in terms of quality of solution when compared with other existing GAs [1,3].

Fig. 2 shows a comparison of FRAG_GA, SWAP_GATSP and OX_SIM when the fitness value of the fittest string is plotted with iteration. The three programs were run for 5000 iterations for kroa100.tsp with population 20. At any iteration, the FRAG_GA has the lowest tour cost. It took 304 seconds, 490 seconds and 304 seconds by FRAG_GA, SWAP_GATSP and OX_SIM respectively for executing 5000 iterations. Moreover, only FRAG_GA is seen to converge at around 800 iterations at the optimal cost value of 21,282 km. On the other hand, the cost is 21912 km for SWAP_GATSP and 25103 km for OX_SIM even after 5000 iterations.

The performance of FRAG_GA on microarray datasets is evaluated with a biological score (not used as fitness function of GA), defined by [5]

$S(n) = \sum_{i=1}^{n-1} C_{i,,i+1}$ where $C_{i,,i+1} = 1$, if gene $i$ and $i+1$ are in the same group

$= 0$, if gene $i$ and $i+1$ are not in the same group.

Using this, a solution of gene ordering has a higher score when more genes within the same group are aligned next to each other. Table 2 compares the performance of our FRAG_GA with other GA based methods in terms of S value. It is clear that FRAG_GA and NNGA [9] are comparable and they both

**Table 2.** Comparison of FRAG_GA with other algorithms in terms of best score

| Algorithms | Cell cycle cdc15 | Cell cycle | Yeast complexes |
|---|---|---|---|
| FRAG_GA | 537 | 635 | 384 |
| NNGA | 539 | 634 | 384 |
| FCGA | 521 | 627 | —- |

dominate FCGA [5]. Note that FRAG_GA is a conventional GA, while NNGA (hybrid GA) uses exhaustive local search methods [10], which provides the key contribution to optimality (not the GA itself). The main reason behind the good results obtained by FRAG_GA is that, biological solutions of microarray gene ordering lie in more than one sub optimal point (in terms of gene expression distance) rather than one optimal point.

## 5 Conclusion

A new "nearest fragment operator" (NF) and "modified version of order crossover operator" (MOC) of GAs are described along with their implementation for solving both symmetric TSP and microarray gene ordering problem. Appropriate number of fragments and appropriate substring length in terms of the number of cities are determined for NF and MOC respectively, and then applied on TSP and microarray data. It appears that NF operator is able to augment the search space quickly and thus obtains much better results compared to other heuristics. Moreover, MOC requires shorter computation time; thereby balancing the overhead corresponding to the NF operator.

## Acknowledgement

This work is supported by the grant no. $22(0346)/02/EMR-II$ of the Council of Scientific and Industrial Research (CSIR), New Delhi.

## References

1. Larranaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S.: Genetic algorithms for the traveling salesman problem: A review of representations and operators. Artificial Intell. Rev. **13** (1999) 129–170
2. Goldberg, D.E.: Genetic Algorithm in Search, Optimization and Machine Learning. Machine Learning, Addison-Wesley, New York (1989)
3. Ray, S.S., Bandyopadhyay, S., Pal, S.K.: New operators of genetic algorithms for traveling salesman problem. Volume 2., Cambridge, UK, ICPR-04 (2004) 497–500
4. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisia by microarray hybridization. Molecular Biology Cell **9** (1998) 3273–3297
5. Tsai, H.K., Yang, J.M., Kao, C.Y.: Applying Genetic Algorithms To Finding The Optimal Gene Order In Displaying The Microarray Data. GECCO (2002) 610–617
6. Davis, L.: Applying adapting algorithms to epistatic domains. Proc. Int. Joint Conf. Artificial Intelligence (Quebec, canada, 1985)
7. TSPLIB: (http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/)
8. (http://www.psrg.lcs.mit.edu/clustering/ismb01/optimal.html)
9. Lee, S.K., Kim, Y.H., Moon, B.R.: Finding the Optimal Gene Order in Displaying Microarray Data. GECCO (2003) 2215–2226
10. Lin, S., Kernighan, B.W.: An effective heuristic for the traveling salesman problem. Operation Research **21** (1973) 498–516