Improved Genetic and Simulating Annealing Algorithms to Solve the Traveling Salesman Problem Using Constraint Programming

Mamoon Alameen
School of Engineering,
The Australian College of Kuwait, Kuwait
m.radiy@ack.edu.kw

Mohammed Abdul-Niby
School of Engineering
The Australian College of Kuwait, Kuwait
m.nibi@ack.edu.kw

Ayad Salhieh
School of Engineering
The Australian College of Kuwait, Kuwait
a.salhieh@ack.edu.kw

Ali Radhi
Bahbahani Projects
Kuwait city, Kuwait
Ali.radhi@gmail.com

Abstract— **The Traveling Salesman Problem (TSP) is an integer programming problem that falls into the category of NP-Hard problems. As the problem become larger, there is no guarantee that optimal tours will be found within reasonable computation time. Heuristics techniques, like genetic algorithm and simulating annealing, can solve TSP instances with different levels of accuracy. Choosing which algorithm to use in order to get a best solution is still considered as a hard choice. This paper suggests domain reduction as a tool to be combined with any meta-heuristic so that the obtained results will be almost the same. The hybrid approach of combining domain reduction with any meta-heuristic encountered the challenge of choosing an algorithm that matches the TSP instance in order to get the best results.**

Keywords-Traveling Salesman Problem; Genetic Algorithm; Simulating Annealing; Domain Reduction

I. INTRODUCTION

The traveling salesman problem (TSP) is a problem of “finding the shortest possible route given a list of cities and the distances between each pair of cities, such that the route visits each city exactly once and returns to the origin” [1]. The increasing numbers of vehicles and the fluctuation of oil prices as well as the need to reduce costs, ignited the search to optimize the travel of sales personnel and the daily delivery and transport plans. Accordingly, the vehicle routing problem (VRP) and the TSP are important problems in the field of customer services and in the distribution network. They played a significant role in reducing the costs and time and hence improving the service. This paper considers a symmetric TSP, where a number of cities (or customers) is given as well as the distances between each pair of these cities, and the problem is to find the shortest possible routes that visits each city exactly once and return to the origin.

This problem was raised, with trial for a mathematical formulation, in the early 19th century in the UK [2]. A general form of the TSP mathematically tackled for the first time by Karl Menger at 1930. Karl Menger was able to define the problem and to find out that the nearest neighbor heuristic would not provide the optimal solution. Since then, the TSP is being a very popular subject and many efforts are carried out to determine the optimal solution for certain variants. In the last years, the TSP problem approach has been extended and modified to provide special solutions for other fields in life, such as for DNA fragmentation or for finding the optimal insert sequence of SMD components and optimal soldering points sequence.

Over the past decades, TSP instances had been encountered using too many types of approaches. Techniques like exact methods, classical and meta-heuristics had been applied to solve TSP instances with various dimensions. It is a fact that choosing between exact and heuristics approach to solve TSP is an easy choice as it is governed by the accuracy vs time concept. Choosing between classic and meta-heuristics is also an easy choice for the same reason. On the other hand, choosing the right meta-heuristics to solve the TSP could be very challenging as some of them are working excellently on one problem and fail to provide a good solution on others [3].

This paper surveys the effect of domain reduction on the final results of using genetic and simulating annealing algorithms to solve TSP. The objective is to minimize the domain of the problem in order to minimize the search iterations for the algorithms and getting close (if not similar) results.

II. MATHEMATICAL FORMULATION

This paper consider the following mathematical formulation for the TSP [4]:

For i = 0, ..., n, let u_i be an artificial variable, and finally take c_{ij} to be the distance from city i to city j.

$$\text{Min} \sum_{i=0}^{n} \sum_{j\neq i, j=0}^{n} c_{ij} x_{ij} \qquad (1)$$

$$u_i \in \mathrm{Z} \qquad i, j = 0, ..., n \qquad (2)$$

$$\sum_{i=0,i\neq j}^{n} x_{ij} = 1 \qquad i,j = 0,...,n \qquad (3)$$

$$\sum_{j=0,j\neq i}^{n} x_{ij} = 1 \qquad i,j = 0,...,n \qquad (4)$$

$$u_i - u_j + nx_{ij} \leq n-1 \qquad 1 \leq i \neq j \leq n \qquad (5)$$

$$x_{ij} = \begin{cases} 1 \\ 0 \end{cases} \qquad (6)$$

Where (1) is the objective function. (2) implies that u_i are integers, (3) is to get sure that each city visited at most once, (4) serves the fact that all cities must be visited, (5) is to get sure that there is only one tour for the problem and (6) implies that the value of *x* is 1 if the path goes from city *i* to city *j* and 0 otherwise.

III. META-HEURISTICS

The quality of the solution obtained by any of the metaheuristic algorithms is usually much better than the ones obtained by the classical algorithms because the metaheuristic algorithms explore all the solution space deeply. However, metaheuristics take more time than the classical heuristics. The following elaborates two popular metaheuristics:

A. Simulating Annealing (SA) [5]

SA is a stochastic relaxation technique that has its origin in statistical mechanics. Formerly, the process of crystallizing a solid by heating it to a high temperature and gradually cooling it down motivated the development of SA.

Assuming $\Delta = f(x) - f(x_t)$, where $f(x)$ is the best value for the objective function found so far, and $f(x_t)$ is the value of the objective function at iteration t. The solution will be accepted as a new current solution if $\Delta \leq 0$. If $\Delta > 0$, any moves with a probability of $e^{-\Delta t}$ increase of the objective function are accepted, where *T* is the temperature and its value varies from large to close to zero. The values of *T* are controlled by a cooling schedule that specifies the temperature values at each stage. In the literature it was proposed that a solution x is drawn randomly in $N(x_t)$ at *t* iterations. If $f(x) \leq f(x_t)$, then x_{t+1} is set equal to *x*; otherwise

$$x_{t+1} = \begin{cases} x & with\ probability\ p_t \\ x_t & with\ probability\ 1-p_t \end{cases}$$

where p_t is a decreasing function of $_t$ and of $f(x) - f(x_t)$.

With the pre-specified values for π_1, π_2 and k_i the SA stops when:

- The value f^* has not decreased by π_1% for at least k_1 consecutive cycles of *T* iterations.
- The number of accepted moves has been less than π_2% of *T* for k_2 consecutive cycles of *T* iterations.
- K_i of *T* iterations have been executed

The application of SA to solve TSP is to take an initial solution to the problem and consider it as the best solution. A parallel search for removing and adding cities from the routes follows. The adding and removing is a random process within the above mentioned boundaries, updating the best solution as the total distance is reduced.

B. Genetic Algorithm (GA) [6]

Coming from a biological background for simulation of the evolution using the gens, the algorithm represents a solution as a population of chromosomes:

$$X^1 = \{X_1^1, \cdots, X_N^1\}$$

N here is the number of vertices (or cities). To proceed, the following three steps are carried out:

- Two "parent" chromosomes from X^1 are selected.
- These parent chromosomes are used to produce offsprings that forms the next generation.
- Each offspring is then mutated randomly utilizing a small probability.

The above three steps will be repeated *K* times for each iteration *t*=1,…,*T*, where $k \leq N/2$ and T is the number of generations. The next step will be applied:

X^{t+1} from X^t. This is achieved by removing the *2k* worst solution in X^t (the ones with the furthest distances) and replacing them with *2k* new offsprings. Additionally, to apply the GA to solve TSP, two points have to be considered:

- Initial population constructor. Which means initial solution to the problem has to be provided
- Determine fitness, crossover and mutation operators. Which means that a criterion for improving the solution (new iteration) has to be specified.

The GA will be repeated for a pre-specified number of iterations.

C. Constraint Programming (CP) [7]

Constraint Programming techniques have been developed since early 1990s. They have two common features:

- constraint propagation
- distribution (labeling) connected with search.

Constraint propagation will lead to Constraint Programming (CP). It would automatically remove from the domain of variables all values that do not fulfill constraints. Let us consider these examples:

- Let X-Y=3 or X<Y. These 2 given constraints would provide information about the values of the variables X and Y, but it is in a poorly usable form. CP will work to simplify such information. If we have, beside X-

Y=3, another information that X+Y=7, then the solution would be: X=5 and Y=2. This simplification will be carried out by a special algorithm, the constraint solver, a fixed part of the CP.

- If we have two variables x and y, with x∈{1..5} and y ∈{1..6}. We introduce here a constraint with x>y+1, then the constraint propagation will reduce the domains to the following values: x∈{3, 4, 5} and y∈{1, 2, 3} because values {1, 2} from x domain do not fulfill the constraint x>y+1 and the y values {4, 5, 6} also conflict with the given constraint.
- In the last example, if we add another constraint x+y=6, then none of the values can be removed.

Usually we don't have the joy of such simple constraints. They are often connected with each other. Therefore, constraint propagation would not remove all values that are in conflict with all constraints and its performance is measured as a trade-off between number of removed values and execution time.

Actually, constraint propagation does not lead to the solution (example above). This explains the need to always add a distribution connected with the search. Distribution is based on incorporation of an additional constraint, often it is a constraint about equality of one variable to one value. A major task of the distribution is to find or choose a proper variable and a suitable value. As soon as this is achieved, a consistency is checked and there will be three possibilities:

- a solution is found
- variables domains are narrowed, but there is no unique solution, so distribution is conducted with another variable
- the additional constraint is inconsistent with other constraints, so the backtrack is made and from the chosen variable domain a chosen value is removed.

This is an iterative process and it is called "search". Search is responsible for stopping after finding the first solution or some number of solutions or all solutions. Search forms a search tree, where each node is a state of variables.

IV. Computations

This paper applies SA and GA to 12 benchmark problem from the TSP library [8]. The size of each problem increase by almost a factor of 2 in order to survey the effect of the reduction. TSP instances that considered in this paper are shown in Table I.

The distance between the cities were calculated and placed in to a symmetric distance (cost) matrix then:

$$R^{*}=R-\frac{s}{100}R \qquad (7)$$

$$s=\begin{cases} 0,10,20,30\ldots \text{ if the size of the problem} \le 250 \\ 0,15,30,45\ldots \text{ if the } 250 \le \text{size of the problem} \le 750 \\ 0,20,40,60\ldots \text{ if the } 750 \le \text{size of the problem} \le 1500 \\ \text{otherwise} \end{cases} \qquad (8)$$

R is the maximum distance in the distance matrix. All the distance values above R^{*} should be neglected. The above domain reduction technique suggests that the arcs between far cities will not be considered. The solution starts by taking all distances, then based on the size of the problem the domain will be reduced by 10% from the maximum distance then 20% and so on until no feasible solution can be obtained (in case the size of the problem less than or equal to 250). The solution considered in this case will be the one with the least distance or cost. Table II provides the results of applying genetic algorithm and simulating annealing to solve TSP combined with domain reduction.

TABLE I. TSP INSTANCES AND DIMENSION

Problem Number	Size	Description
1	29	bays29
2	48	gr48
3	101	eil101
4	202	gr202
5	493	d493
6	1002	pr1002
7	2103	d2103
8	3038	pcb3038
9	7397	pla7397
10	13509	usa13509
11	33810	pla33810
12	85900	pla85900

TABLE II. THE RESULTS OF THE IMPROVED SA AND GA

Problem No.	SA	GA	Optimal Solution	Improved SA	Improved GA
1	2020	2020	2020	2020	2020
2	5046	5046	5046	5046	5046
3	668	670	629	629	629
4	42518	45922	40160	40992	41374
5	54229	73417	35002	37898	379524
6	285534	348604	259045	261409	261409
7	103250	127831	80450	91693	91332
8	169782	183421	137694	148972	148972
9	31886458	41894001	23260728	23853401	23853401
10	35109814	35710228	19982859	22014415	22014415
11	71323137	75115103	66048945	66829102	66829102
12	200219623	265892412	142382641	147920114	147920114

Graphically the obtain results of Table 2 can be illustrated as shown in Figures 1 to 6.

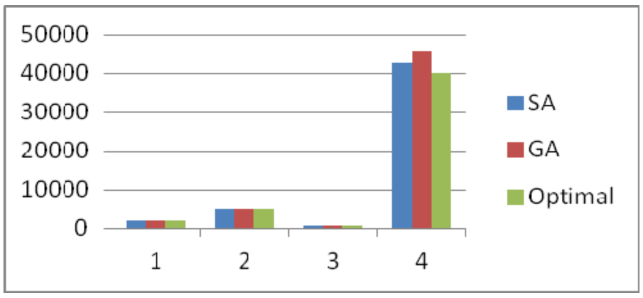

Fig. 1. TSP Instances 1-4 Solved by GA and SA

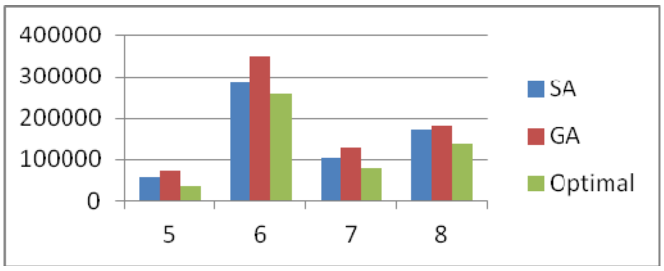

Fig. 2. TSP Instances 5-8 Solved by GA and SA

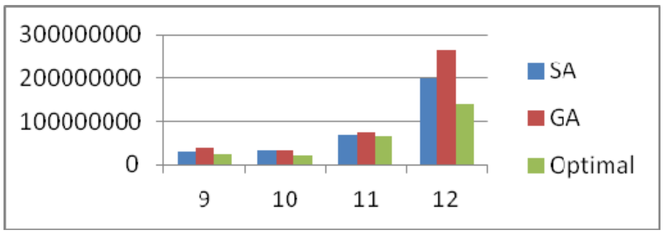

Fig. 3. TSP Instances 9-12 Solved by GA and SA

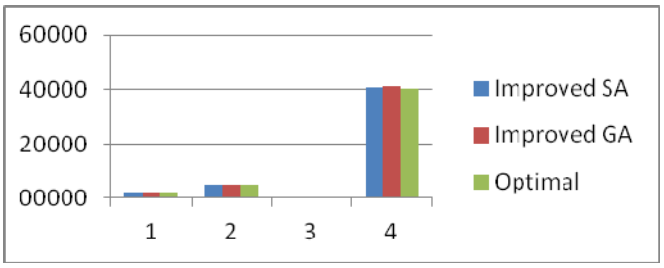

Fig. 4. TSP Instances 1-4 Solved by Improved GA and SA

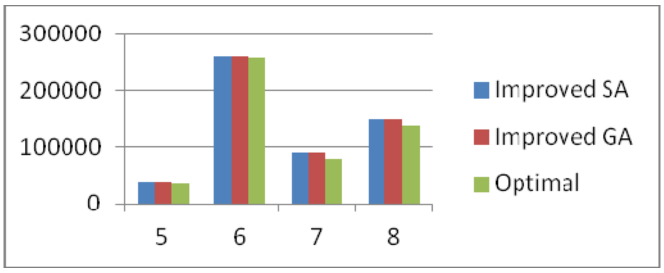

Fig. 5. TSP Instances 5-8 Solved by Improved GA and SA

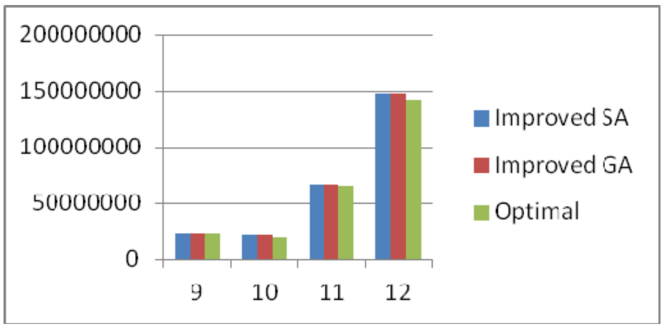

Fig. 6. TSP Instances 9-12 Solved by Improved GA and SA

V. CONCLUSION AND FUTURE WORK

As Table II suggests, small dimension problems can be solved (optimally) sometimes no matter the algorithm applied. For medium to large sized problems the solution obtained by applying genetic algorithm is very close (if not the same) for the solution obtained by applying simulating annealing. In brief combining domain reduction with genetic algorithm and simulating annealing provides the following important advantages:

- improves the solution for one or both algorithms
- using either genetic or simulating annealing provides similar results once domain reduction is combined with the selected algorithm.
- the domain reduction approached improved the accuracy of SA and GA and also minimized the searching process iterations for the large size instances.

In order to acquire the full benefit of using domain reduction with meta-heuristics, more well-known algorithms should be considered. Algorithms like Tabu search and Ant colony should be applied to solve TSP combined with domain reduction in the future. Also, a general and more logical domain reduction approach should be taken in order to minimize the computation time.

REFERENCES

[1] G. Gutin, A. Yeo, A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP", Discrete Applied Mathematics, Vol. 117, No. 1–3, pp. 81–86, 2002

[2] C. H. Papadimitriou, "The Euclidean traveling salesman problem is NP-complete", Theoretical Computer Science, Vol. 4, No. 3, pp. 237–244, 1977

[3] A. Corberán, M. Oswald, I. Plana, G. Reinelt, J. M. Sanchis, "New results on the Windy Postman Problem", Mathematical Programming, Vol. 132, No. 1-2, pp. 309-332, 2012

[4] R. Martí, G. Reinelt, The Linear Ordering Problem. Exact and Heuristic Methods in Combinatorial Optimization, Springer, Heidelberg, 2011

[5] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, "Optimization by Simulated Annealing", Science, Vol. 220, No. 4598, pp. 671–680, 1983

[6] L. M. Schmitt, "Theory of Genetic Algorithms", Theoretical Computer Science, Vol. 259, No. 1–61, 2001

[7] F. Benhamou, N. Jussien, B. O' Sullivan, Trends in constraint programming, John Wiley and Sons, 2007

[8] Universität Heidelberg, TSPLIB, 2013 TSP data