Comparing Genetic Algorithm and Guided Local Search Methods by Symmetric TSP Instances

Mehrdad Nojoumian
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
mnojoumi@cs.uwaterloo.ca

Divya K. Nair
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
dknair@cs.uwaterloo.ca

ABSTRACT

This paper aims at comparing Genetic Algorithm (GA) and Guided Local Search (GLS) methods so as to scrutinize their behaviors. Authors apply the GLS program with the Fast Local Search (FLS), developed at University of Essex [24], and implement a genetic algorithm with partially-mapped and order crossovers, reciprocal and inversion mutations, and rank and tournament selections in order to experiment with various Travelling Salesman Problems. The paper then ends up with two prominent conclusions regarding the performance of these meta-heuristic techniques over wide range of symmetric-TSP instances. First, the GLS-FLS strategy on the s-TSP instances yields the most promising performance in terms of the near-optimality and the mean CPU time. Second, the GA results are comparable to GLS-FLS outcomes on the same s-TSP instances. In the other word, the GA is able to generate near optimal solutions with some compromise in the CPU time. 0112(54 357.48 Tm-5(C14(aa]TJ)-1(s)g]TJ o]TJ r)14(eu)-15(bl)TJ no)-1doss D14(au)-15(cr)14(epo)-1so]TJ r0 Tc

2.1 Combinatorial Optimization Problem

A combinatorial optimization problem can be defined as assigning values to a set of decision variables such that a function on these variables (objective function) is minimized when subjected to the specified set of constraints [10]. Hard combinatorial problems like the travelling salesman problem are challenging to be solved and the solving time grows exponentially with the size of the problem. There is no existing algorithm which can solve the TSP problem with a polynomial complexity. In fact, it is a prominent illustration of a class of problems in computational complexity theory which are classified as NP-hard [11].

2.2 Travelling Salesman Problem

Given a set of cities, n, and the distances among the cities, the TSP problem is to find a minimum-length tour such that every city is visited exactly once and returns to the starting point [20]. The formal definition as taken from [20] is as follows: Given a directed graph, Graph = (V, A) where V is the vertex set: {1...n} and A is the arc set: {(i, j): i and j Є V}, a cost factor: $C_{ij} \geq 0$ is associated with every arc. The TSP problem finds a partial digraph, (V, A_1) of G such that $|A_1|$ = n and for every vertex pair, v_1, v_2 Є V, there exist paths from v_1 to v_2 and v_2 to v_1 in G, f Tw oTw -24.5424.5424.54 p1 W R

Zhang and Looks [5] present a new method for the traveling salesman problem which incorporates backbone information into the Linbeen taken from parent[1]. Finally, we fill in the gaps with cities that have not yet been taken, Figure 5.

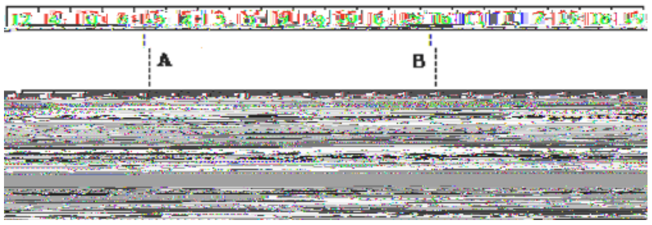

Figure 5. Partially-mapped cross over.

In the second approach, called *Order Cross Over* [17 and 14], we choose points A and B and copy that range from parent[1] to the child similar to the prior method. Subsequently, we fill in the remaining indexes with the unused cities in the order that they appear in parent[2] to [illegible]

>> 0.007 3.TMF8[(t)-6(4(a)6(in)6-5(w)-5(gSy)16(elt)]TeRorr
8TJ(c)11=c 0T4 inc(de).Tc

are penalized depending on the costs, that is, the features with higher costs are penalized more often than the features with lower costs. Since the penalty parameters of high cost features are increased, the solutions possessing these features will be neglected next time the local search procedure is called, and therefore search proceeds to more promising regions of the search space.

5.1 Guided Local Search Procedure

The GLS process is commenced by initializing all the penalty parameters to zero. At first, the local search procedure is called and local search proceeds until the first local minimum is reached. The first time and every other time a local minimum is encountered, the current cost function is modified to a new augmented cost function by incrementing the penalties of those features for which the utility function is a maximum. Then the local search procedure is invoked again by using the modified augmented cost function. Figure 7 presents the basic GLS idea; escaping from a local minimum in the landscape by increasing the objective function value of its solutions. Figure 8 describes the pseudo code for the guided local search process.

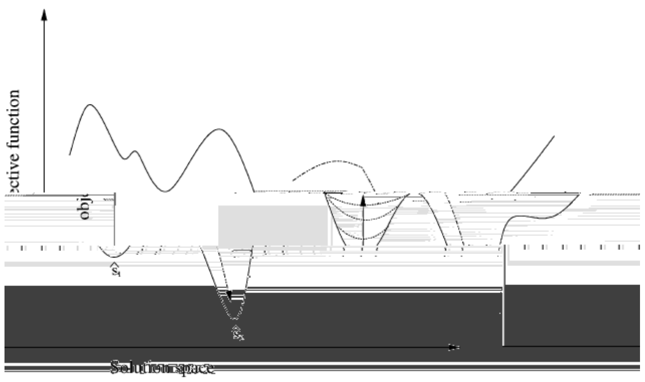

Figure 7. The basic GLS idea.

```
procedure GuidedLocalSeach(S, g, $\lambda$, [$I_1$, ...,$I_M$], [$c_1$,...,$c_M$], M)
begin
    k ← 0;
    $s_0$ ← random or heuristically generated solution in S;
    for i ←1 until M do /* set all penalties to 0 */
        $p_i$ ← 0;
    while StoppingCriterion do
    begin
        h ← g + $\lambda$ * $\sum p_i * I_i$;
        $s_{k+1}$ ← LocalSearch($s_k$, h);
        for i ←1 until M do
            util$_i$ ← $I_i(s_{k+1})$ * $c_i$ / (1+$p_i$);
        for each i such that util$_i$ is maximum do
            $p_i$ ← $p_i$ + 1;
        k ← k+1;
    end
    s* ← best solution found with respect to cost function g;
    return s*;
end
```

Figure 8. GSL pseudo code.

Where, S: search space, g: cost function, h: augmented cost function, λ: regularization parameter, I_i: indicator function for fea0051>o491/16.8 1(n)-1C , executed most of the s-TSP instances for about 10 runs with a time budget of 700 sec/run, and the number of iterations is hard coded in GLS solver to 200K.

The GA algorithm is implemented in Java. Most of the s-TSP instances were executed for 3-6 runs and the maximum number of iterations is set to 15000. For each s-TSP instance, we first generated (4*N, N-number of cities) number of random solutions and then applied the partially mapped crossover, the inverse mutation and the rank selection; since this combination yielded the best outcomes. Table 1 shows the experimental results of GLS-FLS and GLS-greedy LS and Figure 9 represents the graphical comparison between these two results. It can be deduced from the results that, for all the s-TSP instances both GLS-FLS and GLS-greedy LS compute optimal solutions. The performance of both GLS variants is almost the same regarding the optimal solutions, but the results show that GLS-greedy LS consumes more CPU time than GLS-FLS especially from *bier127* to *lin318*. This remarkable time difference exhibited by GLS-greedy LS is due to the inherent nature of the greedy-LS, since the whole neighborhood is searched to find the local minimum during each stage.

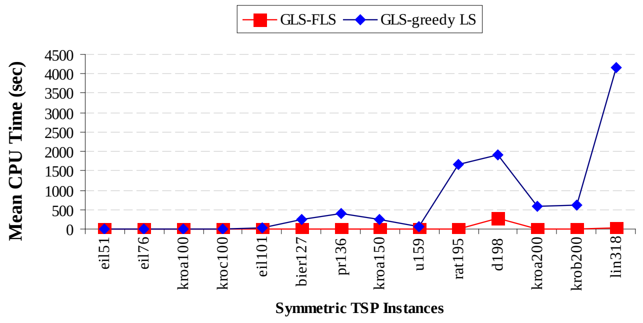

Figure 9. Graph-plot of GLS-FLS and GLS –greedy LS.

Table 1. Comparison of GLS-FLS and GLS –greedy LS.

TSP Instances	GLS-FLS			GLS- greedy LS		
	Mean CPU Time(Sec)	Number of Runs	Excess %	Mean CPU Time(Sec)	Number of Runs	Excess %
eil51	0.57	10	0	1.73	10	0
eil76	0.72	10	0	3.58	10	0
kroa100	1.7	10	0	11.72	10	0
kroc100	0.75	10	0	12.21	10	0
eil101	0.5	10	0	17.62	10	0
bier127	6.6	10	0	236.29	10	0
pr136	12.37	10	0	396.99	9	0.001
kroa150	7.55	10	0	257.58	10	0
u159	4.6	10	0	72.05	10	0
rat195	11.6	10	0	1656	8	0.01
d198	270	10	0	1914.4	10	

Table 2. Comparison of GLS-FLS-2opt and GA.

TSP Instances	GLS-FLS-2opt					GA				
	Mean CPU Time (Sec)	Tour Length	Number of Runs	Iteration	Excess %	Mean CPU Time (Sec)	Tour Length	Number of Runs	Iteration	Excess %
eil51	0.57	426	10	1307	0	3	438	6	2799	0.46
berlin52	0.26	7542	10	563	0	2.78	7542	6	1731	0
eil76	0.72	538	10	3141	0	13	557	6	4754	3.53
kroa100	1.7	21282	10	7485	0	25.2	21466	6	8942	0.86
kroc100	0.75	20749	10	9293	0	33.2	21096	6	6869	1.6
eil101	0.5	629	10	2315	0	27.9	651	6	4999	3.49
bier127	6.6	118282	10	32324	0	90.35	121089	6	9446	2.37

ultimately find the global optimum, thus balancing intensification and diversification.

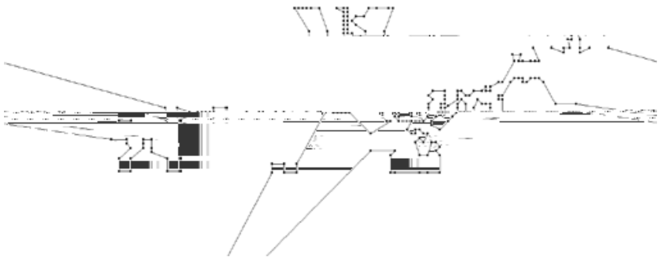

Figure 12. Distribution of cities in d198: drilling problem.

8. ACKNOWLEDGMENTS

We highly appreciate Professor Peter van Beek as well as the anonymous reviewers for their comments.

9. REFERENCES

[1] Voudouris, C. 2000. Guided Local Search Joins the Elite in Discrete Optimization. DIMACS Workshop on Constraint Programming and Large Scale Optimization, 29-40.

[2] Voudouris, C. and Tsang, E. 1998. Guided Local Search. European Journal of Operations Research 113, 80-119.

[3] Voudouris, C. 1997. Guided Local Search for Combinatorial Optimisation Problems, PhD Dissertation, Department of Computer Sciences, University of Essex.

[4] Karova, M., Smarkov, V., and Penev, S. 2005. Genetic Operators: Crossover and Mutation in Solving the TSP Problem. Conference on Computer Systems and Technologies, 6 pages.

[5] Zhang, W. and Looks, M. 2005. A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones, 19th International Joint Conference on Artificial Intelligence. 343-348.

[6] Dong, S., Guo, F., Yuan, J., Wang, R. and Hong, X. 2006. A Novel Tour Construction Heuristic for Traveling Salesman Problem Using LFF Principle. In Proceedings of the Joint Conf. on Information Sciences, Kaohsiung, Taiwan, 4 pages.

[7] Pullan, W. 2003. Adapting the Genetic Algorithm to the Travelling Salesman Problem. Evolutionary Computation 2, 1029-1035.

[8] Gang, P., Limura, L. and Nakayama, S. 2003. A Multiple Heuristic Search Algorithm for Solving Traveling Salesman Problem. Parallel and Distributed Computing, 779-78311(t)-6(eet [(I)-4(nc14(7(m)26(elad [(I-2(ak)ear)3(achTc 0.004 Tw 0.7 Tw 19.(-))-5(th)-5(e)6( )]

[4 m.n,1(i)n7(o.11(c)196.)(a)Po.1Pltip9D1

[4