Evolving TSP Heuristics Using Multi Expression Programming

Mihai Oltean and D. Dumitrescu

Department of Computer Science,
Faculty of Mathematics and Computer Science,
Babeş-Bolyai University, Kogălniceanu 1
Cluj-Napoca, 3400, Romania.
`{moltean, ddumitr}@cs.ubbcluj.ro`

Abstract. Multi Expression Programming (MEP) is used for evolving a Traveling Salesman Problem (TSP) heuristic for graphs satisfying triangle inequality. Evolved MEP heuristic is compared with Nearest Neighbor Heuristic (NN) and Minimum Spanning Tree Heuristic (MST) on some difficult problems in TSPLIB. The results emphasizes that evolved MEP heuristic is better than the compared algorithm for the considered test problems.

1 Introduction

Multi Expression Programming [4] is a Genetic Programming (GP) [3] that uses a linear representation of the chromosomes. MEP representation is similar to the way in which *C* and *Pascal* compilers translate mathematical expressions into machine code.

MEP technique is used here for discovering Traveling Salesman Problem (TSP) heuristics for graphs satisfying triangle inequality (TI graphs). This option was chosen due to the existence of a big number of real-world applications implying TI graphs (e.g. plains, trains and vehicles routes). MEP technique is used to learn a path function f that is used for evaluating the reachable nodes. This function serves as a heuristic for detecting the optimum path.

Evolved function f is compared with some well known heuristics. Numerical experiments emphasize that (for considered examples) MEP function outperforms dedicated heuristics.

2 MEP for TSP

TSP problem for TI graphs (i.e. satisfying triangle inequality) is stated as follows.

Consider a set $C = \{c_0, c_1, \dots, c_{N-1}\}$ of cities, and a distance $d(c_i, c_j) \in \mathbb{Z}^+$ for each pair $c_i, c_j \in C$, $d(c_i, c_j) = d(c_j, c_i)$, and for each three cities $c_i, c_j, c_k \in C$, $d(c_i, c_j) \leq d(c_i, c_k) + d(c_k, c_j)$. The tour $\langle c_{\pi(0)}, c_{\pi(1)}, \dots, c_{\pi(N-1)} \rangle$ of all cities in C having minimum length is needed [2]

TSP problem with triangle inequality is an NP-complete problem [2]. No polynomial time algorithm for solving TSP problem is known. Several heuristics for solving TSP problem have been proposed. The most important are Nearest Neighbor and the Minimum Spanning Tree [1,2].

In this section we address the problem of discovering heuristics that can solve TSP rather than solving a particular instance of the problem. MEP technique is used for evolving a path function f that gives a way to choose graph vertices in order to obtain a Hamiltonian cycle. The fitness is assigned to a function f in the current population by applying f on several randomly chosen graphs (training set) and evaluating the results. Evolved path function may be used for solving particular instances of TSP. For each problem the graph nodes are evaluated using the path function f and are added one by one to the already build path. The algorithm for TSP using evolved path function f may be described as follows:

```

S1. Let  $c_{\pi(0)} = c_0$  {the path starts with the node  $c_0$ }
S2.  $k = 1$ ;
S3. while  $k < N - 1$  do
S4. Using function  $f$  select  $c_{\pi(k+1)}$  – the next node of the path
S5. Add  $c_{\pi(k+1)}$  to the already built path.
S6.  $k = k + 1$ ;
S7. endwhile

```

S_4 is the key step of this algorithm. The procedure that selects the next node of the path in an optimal way uses the function f evolved by the MEP technique as described in the next sections.

Path function f has to use (as input) some information about already build path and some information about unvisited nodes. We consider a special terminal set which is independent with respect to the number of graph nodes.

Let us denote by y_1 the last visited node (current node). We have to select the next node to be added to the path. In this respect all unvisited nodes are considered. Let us denote by y_2 the next node to be visited.

For evolving path function f we consider a set T of terminals involving the following elements:

- $d_{y_1-y_2}$ – distance between the graph nodes y_1 and y_2 ,
- $min_g_y_1$ ($min_g_y_2$) – the minimum distance from the nodes y_1 (y_2) to unvisited nodes,
- $sum_g_y_1$ ($sum_g_y_2$) – the sum of all distances between nodes y_1 (y_2) and unvisited nodes,
- $prod_g_y_1$ ($prod_g_y_2$) – the product of all distances between nodes y_1 (y_2) and unvisited nodes,
- $max_g_y_1$ ($max_g_y_2$) – the maximum distance from the nodes y_1 (y_2) to unvisited nodes,
- $length$ – the length of the already built path.

Table 1. The performance of evolved MEP heuristic, NN and MST on some problems in TSPLIB. *Length* is the length of the TSP path obtained with one of the considered heuristics. Each node of the graph has been considered as the first node of the path

Problem	MEP		NN		MST	
	Length	Error (%)	Length	Error (%)	Length	Error (%)
a280	2858.86	10.85	3084.22	19.58976	3475.23	34.75
att48	37188.2	10.93	39236.9	17.04227	43955.8	31.11
bier127	134945	14.08	127954	8.177068	152747	29.13
d198	17780.7	12.67	17575.1	11.37579	17957.6	13.79
d493	43071.3	23.05	41167	17.61328	41846.6	19.55
d657	56965.6	16.46	60398.7	23.48442	63044.2	28.89
fl417	13933.8	17.47	13828.2	16.58545	16113.2	35.85
gil262	2659.17	11.82	2799.49	17.72456	3340.84	40.48
kroB100	24801	12.01	25883	16.90077	28803.5	30.09
kroB200	33267.4	13.01	35592.4	20.91042	40619.9	37.98
pcb442	56948.3	12.15	57856.3	13.9397	73580.1	44.9
pr264	55827.1	13.61	54124.5	10.15468	65486.5	33.27
rat575	7573.6	11.82	7914.2	16.84925	9423.4	39.13
rat783	9982.96	13.36	10836.6	23.05928	11990.5	36.16
rd400	16973.3	11.07	18303.3	19.77816	20962	37.17
u574	43095.6	16.77	44605.1	20.86465	50066	35.66
u724	46545.7	11.06	50731.4	21.04844	60098.9	43.39

Set T of terminals is chosen in such way to be independent of the number of graph nodes. This choice confers flexibility and robustness to the evolved heuristic.

For evolving a MEP function for TSP problem we may consider the following set of function symbols: $F = \{+, -, /, *, \cos, \sin, \min, \max\}$.

The node y_2 that generates the lowest output of evolved function f is chosen to be the next node of the path. Ties are solved arbitrarily. For instance we may consider the node with the lowest index is selected.

Example

Consider the MEP linear structure:

- 1: $d_{y_1-y_2}$
- 2: $\min_{g-y_1}$
- 3: $+ 1, 2$
- 4: $\sum_{g-y_2}$
- 5: $* 2, 4$

This MEP individual encodes the path functions f_1, f_2, f_3, f_4, f_5 given by:

$$\begin{aligned} f_1 &= d_{y_1-y_2}, \\ f_2 &= \min_{g-y_1}, \\ f_3 &= d_{y_1-y_2} + \min_{g-y_1}, \\ f_4 &= \sum_{g-y_2}, \\ f_5 &= \min_{g-y_1} * \sum_{g-y_2}. \end{aligned}$$

The fitness (quality) of a detected path function f is defined as the sum of the TSP path length of graphs in the training set. Thus the fitness is to be minimized.

3 Numerical Experiments

In this experiment we evolve a heuristic for solving TSP problem and we assess its performance by applying it to some real-world instances taken from TSPLIB [5].

Let us denote by G_k the set of class of TI graphs having maximum k nodes.

MEP algorithm considers the class G_{50} (i.e. graphs having 3 to 50 nodes) for training and the class G_{100} for validation. MEP algorithm parameters are *Population Size* = 300; *Number of generations* = 100; *Chromosome length* = 40.

A path function evolved by the MEP algorithm is:

$$f = (sum_g(y2)) * (d_{y1-y2} - (max(d_{y1-y2}, max_g(y1))) + d_{y1-y2}).$$

Heuristic function f that is evolved by MEP technique is directly used for building the optimal path. For assessing the performance of evolved MEP heuristic we use it for solving several difficult problems in TSPLIB [5]. Other two well-known heuristics are used in this comparison: Nearest Neighbor and Minimum Spanning Tree heuristic [1,2]. Results are presented in Table 1.

Table 1 shows that evolved MEP heuristic performs better than NN and MST on all considered problems.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R. R.: Introduction to Algorithms, MIT Press, (1990)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to NP-Completeness, Freeman & Co, San Francisco, CA, (1979)
3. Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, (1992)
4. Oltean, M.: Solving Even-Parity Problems using Multi Expression Programming, In: Ken Chen et. al (eds.): Proceedings of the the 7th Joint Conference on Information Sciences (2003) 315-318
5. Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library, ORSA, Journal of Computing, Vol. 3, no 4, (1991) 376-384