Application of Monte-Carlo Tree Search to Traveling-Salesman Problem

Masato Shimomura Yasuhiro Takashima

Faculty of Environmental Engineering
University of Kitakyushu
Kitakyushu, Fukuoka 808-0135
{masato.shimomura@is.env.,takasima@}kitakyu-u.ac.jp

Abstract— This paper shows an application of Monte-Carlo Tree Search (MCT) to Traveling-Salesman Problem (TSP). Compared with the simulated annealing, which is one of the general probabilistic optimization methods, MCT has very high ability of optimization with problem-aware implementation. Its efficiency is confirmed, empirically.

I. Introduction

Recently, the performance of computers and the size of the problems are increased. But, there is a gap, that is, the speed for the problems is larger than that for the computers. Thus, the importance of the efficient method becomes larger and larger. Especially, the problems in NP-hard [1] needs to the efficient method, strongly.

For the optimization method, there are 3 types, 1) strict method, 2) approximate method, and 3) heuristic method. The strict method is the method which outputs the optimum solution exactly. It typically is formulated as the SAT problem [2] or the ILP problem [3, 4]. These frameworks have noted for this decade, since the performance of their solvers are much improved. But, the exponential increase of the runtime can not be avoided due to the character of the method. The approximate method has the guarantee of the solution performance. [1] introduces several approximate method. One of the drawbacks of the approximate algorithm is that it may not exists for some problem. The heuristic method does not have any guarantee of the solution but has reasonable runtime algorithm. In this class, simulated annealing and genetic algorithm are enumerated. These methods are general. Thus, they are applicable to any problem. But, especially, when the constraint is strict, the search efficiency degrades.

Recently, Monte-Carlo tree search is noted, especially from the game-tree search area. Furthermore, its application to the optimization problem is proposed [5]. This paper follows the proposition from [5] and introduces an application of Monte-Carlo tree search to Traveling-salesman problem, one of the combination problem. Compared with SA, it obtains 40% performance improvement with same runtime and the same performance with 685 times faster. Thus, we conclude the Monte-Carlo tree search is efficient.

The rest of this paper is as follows: Section II introduction the Monte-Carlo tree search; Section III describes how to apply the Monte-Carlo tree to Traveling-salesman problem; Section IV reports experimental results; and SectionV concludes this paper.

II. Monte-Carlo Tree Search

Monte-Carlo method is a method which calculates statistical values, for example, an expected value, with a random sampling. For a recent decade, Monte-Carlo Tree Search (MCT) is watched, where it utilizes Monte-Carlo method to the tree search. It is a beginning to apply MCT to the game tree search. Furthermore, there are several propositions to employ MCT to the solution of the optimization problems. Especially, the utilization of MCT to CAD is also considered.

Fig. 1 shows a framework of MCT. In the MCT, each node and each terminal corresponds to a sub-solution and a solution, respectively. In Fig. 1, a rectangle node which is a terminal corresponds to a solution, and the other nodes correspond to sub-solutions. Each edge between two nodes corresponds to an expansion from a sub-solution to a larger solution. In this case, the original sub-solution is parent and the constructed solution is child. In Fig. 1, for the parent node p1, node c1, c2, and c3 are children. For the child node, two classes exist, *extended* and *unextended*. They correspond to the sub-solutions which have already visited and have not visited yet, respectively, during the previous search. In Fig. 1, the nodes c1 and c2 are extended and the node c3 is unextended. For the terminal, its evaluation can be calculated uniquely.

Fig.2 shows a pseudo code of MCT. In the figure, V_0 is a root node.

In the code shown in Fig.2, node V_0 is a root node and its stopping criteria is until satisfying the calculation time or the iteration number which are given. This flow is applied to the example shown in Fig.1.

At the beginning of the execution of TREE_POLICY, the search starts at the node V_0. At the search of the node V_0, all children of V_0 are extended. Thus, select the child p which has the best evaluation among them. At the node p, there exist unextended child, thus select a node among the unextended children randomly. As a result, the node c3 is selected and V_l is set to c3 (shown in Fig.3).

After the selection of the node c3, construct the solution corresponding to the terminal Ve by DEFAULT_POLICY and cal-

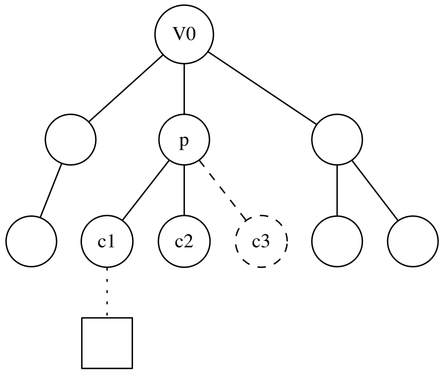

Figure 1: Framework of Monte-Carlo Tree Search

```
function MCT_SEARCH($V_0$) {
    while (stopping criteria is not satisfied) {
        $V_l \leftarrow$ TREE_POLICY ($V_0$)
        $\Delta \leftarrow$ DEFAULT_POLICY ($V_l$)
        BACKUP ($V_l, \Delta$)
    }
    return BEST_CHILD ($V_0$)
}
```

Figure 2: Pseudo code of MCT

culate the evaluation value of Ve (shown in Fig.4).

Finally, in BACKUP, the information including the evaluation is returned hierarchically and update the information of the ancestors (Fig.5).

In the remains of this section, the above steps are introduced briefly. In the following description, the optimization objective is the maximization.

A. TREE_POLICY

This step is formulated by the Multi-Armed Bandit Problem [6] which selects the best item statistically among K items. Its pseudo code is shown in Fig.6.

This Multi-Armed Bandit Problem can be solved with high probability if a large number of trials are executed. However, the selection must be done with the limited number of trials. In this case, Upper Confidential Bound for Trees (UCT) is proposed. This method select the node which has the highest evaluation by Eq.1.

$$\mathrm{UCT} = \overline{x_j} + 2C_p\sqrt{\frac{2\ln n}{n_j}}, \tag{1}$$

where $\overline{x_j}$ is the average value of node j among the previous trials; n is the total number of trails; n_j is the number of trials

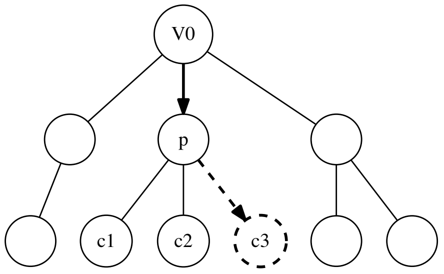

Figure 3: Example of TREE_POLICY

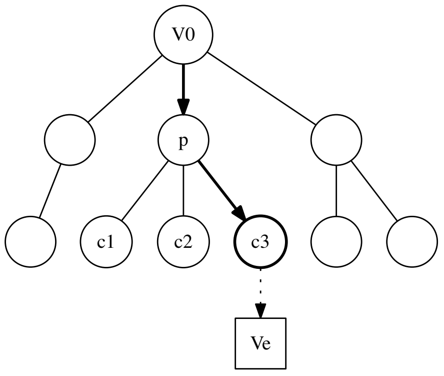

Figure 4: Example of DEFAULT_POLICY

which select node j; C_p is the weight. This equation tends to return the large value when the average of the previous evaluation is large. But, when the number of trials is relative small, the sub-solution corresponding to the node has a potential where it can extend to a good solution. Thus, the second term reflects such variation.

B. DEFAULT_POLICY

This step consists of constructing solution randomly from the sub-solution corresponding to the selected node by TREE_POLICY and evaluating the solution. The pseudo code is shown in Fig.7.

C. BACKUP

This step traverses the tree from the selected node to root hierarchically with updating the information of each node by the solution and its evaluation from DEFAULT_POLICY. The update modifies the average $\overline{x_j}$ and the trial number n_j for each node j.

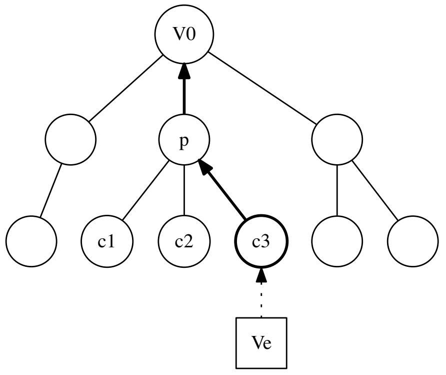

Figure 5: Example of BACKUP

```
function TREE_POLICY ($V$) {
    while ($V$ is not terminal) {
        if (all children of $V$ are extended) {
            select the child of $V$ with the best evaluation
            of UCT(Eq.(1)) and set it to $V$
        } else {
            select the child $V'$ of $V$ which is unextended
            return $V'$
        }
    }
    return $V$.
}
```

Figure 6: Pseudo code of TREE_POLITY

III. Application of MCT to TSP

This section describes how to apply MCT to Traveling Salesman Problem (TSP). The TSP is defined as Def.1

In this paper, TSP is a path-version between two cites. TSP is known as NP-hard problem [1]. Thus, it is difficult to solve it in the polynomial time. To solve it, there are several methods proposed. Especially, [1] introduces an approximated algorithm with the minimum spanning tree (MST).

This paper tries to apply MCT to TSP. In the rest of this section, this application is described briefly.

A. Solution Representation

The solution of TSP is represented by the sequence of the cities which corresponds to the order of the route. Thus, a sub-solution corresponding to the node of MCT is represented as the sub-sequence.

```
function DEFAULT_POLICY ($V$) {
    while ($V$ is not terminal) {
        Select the child $V'$ of $V$ randomly
        set $V \leftarrow V'$
    }
    return (evaluation of $V$)
}
```

Figure 7: Pseudo code of DEFAULT_POLICY

Def. 1 (Traveling Salseman Problem (TSP))
Input: *Set of cities* $C = \{c_i\}$ *and distance function* $d(c_i, c_j)$
Output: *Minimum route length*
Constraint: *The route visits each city exactly once*

B. TREE_POLICY

To define TREE_POLICY, the evaluation function UCT must be defined. Since TSP is the minimized problem, Eq.1 is modified as Eq.2, where the sign of the second term is changed to negative.

$$\mathrm{UCT} = \overline{x_j} - 2C_p\sqrt{\frac{2\ln n}{n_j}} \qquad (2)$$

In TREE_POLICY, the best child with UCT is the minimum node by Eq.2. For the selection of C_p, no concrete formulation exists. This paper considers two methods, 1) utilization of MST value, and 2) utilization of sampling values. For the utilization of MST value, we calculate the MST value for given Graph and decide C_p by it. On the other hand, for utilization of sampling values, we focus on the framework of MCT. In the MCT process, the UCT value is not used during the extension of root node. Therefore, when all children are extended, standard deviation of the route length among the children is calculated and C_p is decided according to it.

C. DEFAULT_POLICY

In DEFAULT_POLICY, several random methods to construct a solution can be selected. In this paper, we consider two methods, 1) uniform distribution, and 2) roulette selection. This step constructs the route from the sub-route corresponding to the node selected by TREE_POLICY. We note the selection of the next city among all unvisited cities from the current last city. For the uniform distribution, each unvisited city has an equal probability of selection. On the other hand, for the roulette selection, we calculate the reciprocal of the distance from the last city to each unvisited city and employ the ratio over the total of the reciprocal as probability.

As comparing them, the method with the uniform distribution can be calculated easier than that with roulette selection.

Thus, the execution time should be shorter. On the other hand, for the minimum route, the distance between the consequence two cities may be desirable. Thus, the resultant route may be smaller.

D. BACKUP

In this step, the best route is remained for each node.

IV. Experiments

To confirm the performance of MCT, we implement MCT on the computer. The computational environment is as follows: Processor is Intel Core i5 3.2GHz; Memory is 32GB 1600MHz DDR3; OS is OSX 10.11.4. we compare the results among Simulated annealing (SA), MCT with uniform distribution and C_p by MST (MCT_UM), MCT with roulette selection and C_p by MST (MCT_RM), and MCT with roulette selection and C_p by standard deviation (MCT_RS). For SA, the initial temperature, the final temperature, the iteration number on each temperature, and the cooling ratio are 10000K, 10K, 1000, and 0.99, respectively. For the movement of SA, we employ that two cities are selected randomly and exchange their orders. For the stopping criteria of MCT, we use the running time of SA.

For the benchmarks, we select 77 data from [7]. The distribution of the number of cites is between 48 and 15112.

A. Comparison with the random method on DEFAULT_POLICY

We confirm the comparison of the results with the random method on DEFAULT_POLICY. In this experiment, set $C_p = 2 * (MST)$. The experimental result is shown in Fig.8, where the horizontal and vertical axis correspond to the logarithm of the number of cities and the logarithm of the normalized route length by MST. In this figure, SA, MCT_Uni2, and MCT_RLT2 correspond to the results from SA, MCT_UM, and MCT_RM, respectively.

From the comparison, the utilization of uniform distribution is a little worse than SA. On the other hand, the utilization of roulette selection is much better than SA. That is, the utilization of uniform distribution outputs 8% longer route length, while that of roulette selection outputs 37% shorter route length. Thus, we confirm the method how to construct a solution much is important.

We also confirm the efficiency of the roulette selection by the runtime in which the MCT can achieve better solution than SA. As a result, only 0.15% runtime is needed on the average. Thus, MCT is about 685 times faster than SA.

B. Comparison with C_p

Next, we confirm the affect of C_p. In this experiment, DEFAULT_POLICY employs roulette selection. The experimental results are shown in Fig.9, where the horizontal and vertical axis also correspond to the logarithm of the number of cities and the logarithm of the normalized route length by MST. In this figure, k*MST and k*SD correspond to the result with $C_p = k * (MST)$ and $C_p = k * (StandardDeviation)$, respectively.

From the result, no significant difference exist among them. For example, the average improvements from SA are 0.628 for 0*MST, 0.632 for 2*MST, 0.633 for 4*MST, 0.629 for 0*SD, 0.633 for 2*SD, and 0.634 for 4*MST, respectively. The difference among them is about 1%. Especially, 0*MST and 0*SD correspond to no consideration of the second term in Eq.2. Thus, it means that almost same results are obtained with or without this term. The reason may be that no affect by the consideration of C_p may occur since the efficiency of the utilization of roulette selection is good enough. However, the decision of C_p with sampling may be general. Thus, it is easy to apply the decision to other problems.

V. Conclusion

This paper proposes the application of Monte-Carlo tree search to Traveling-Salesman problem. Compared with simulated annealing, it was confirmed empirically that MCT outputs 1) 40% length route with the same runtime, and 2) same length route with 685 times faster runtime. From the experiments, we also confirmed that the random selection in DEFAULT_POLICY is much important to the result. As a consequent, the efficiency to apply MCT to the optimization problem is confirmed.

The Future works are as follows: (1) application of MCT to EDA problem; (2) proper decision of UCT. For the issue (1), the architecture of DEFAULT_POLICY is important to obtain the high optimization ability. Thus, we need to select the proper method to each problem. For the issue (2), TSP does not matter the construction of UCT. But, in general, C_p may be important to the performance of result. We expect the standard deviation based method is promising. We plan to show its efficiency.

References

[1] M. R. Garey and D. S. Johnson, "COMPUTERS AND INTRACTABILITY", 1979.

[2] MiniSat, `http://minisat.se`.

[3] IBM CPLEX Optimizer, `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

[4] Gurobi Optimizer, `http://www.gurobi.com`.

[5] Yusuke Matsunaga, "On applications of Monte-Carlo tree search algorithm for CAD problems", IEICE Tech. Rep., VLD2015-46, pp. 51–55, 2015.

[6] P.Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem", Machine Learning, Vol. 47, No. 2, pp.235–256, 2002.

[7] Symmetric traveling salesman problem, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`.

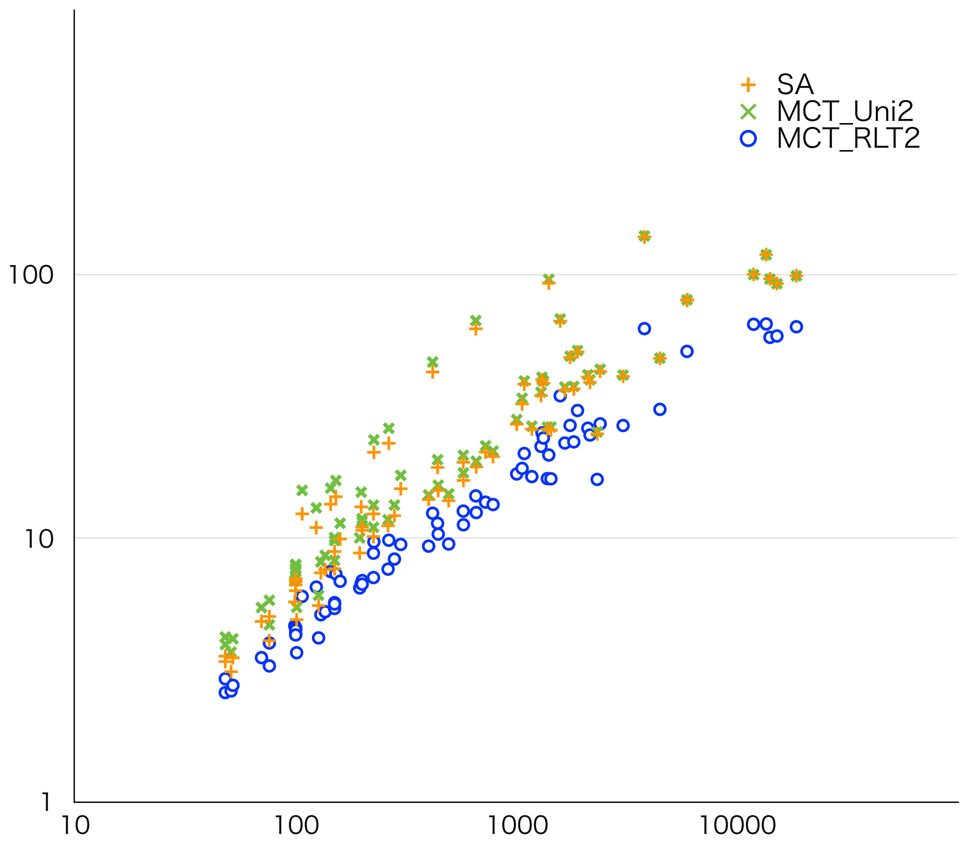

Figure 8: Comparison of the route length between the random method on DEFAULT_POLICY

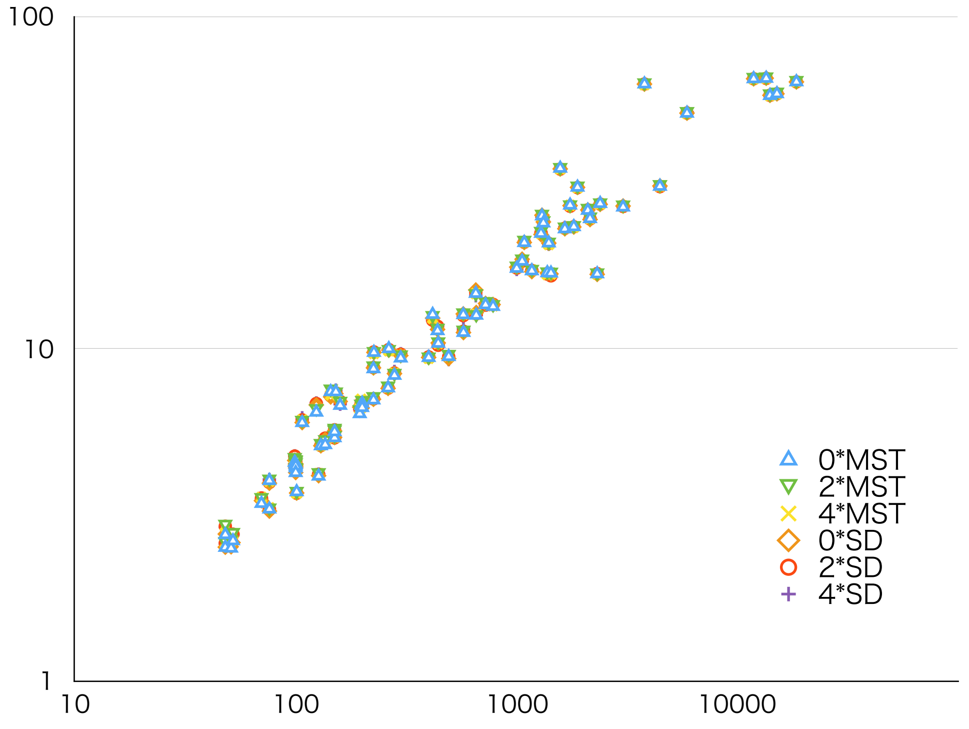

Figure 9: Comparison with C_p