Fast Algorithms to Enumerate All Common Intervals of Two Permutations and Their Applications

東京工業大学 宇野 毅明 (Takeaki Uno)[♠]
京都大学 柳浦 睦憲 (Mutsunori Yagiura)[♣]

Abstract

Given two permutations of n elements, a pair of intervals, one from each permutation, consisting of the same set of elements is called a *common interval.* Some genetic algorithms based on such common intervals have been proposed for sequencing problems, and have exhibited good prospects. In this paper, we introduce three types of fast algorithms to enumerate all common intervals. One of them is an algorithm with $O(n+K)$ worst case running time (denoted RC; abbreviation for reduce candidate), where K ($\leq \binom{n}{2}$) is the number of common intervals.

Then, given two trees with labels on their leaves, we consider the enumeration of all *common subtrees*, i.e., pairs of subtrees having the same set of leaf labels. This problem has an application in constructing evolutionary trees. By using the algorithm RC, we can derive a fast randomized algorithm with $O(n \log^2 n)$ expected running time if we are given two binary trees of depth $\log_2 n$, where n is the number of leaves. The expected running time becomes $O(n)$ if the same two binary trees of depth $\log_2 n$ are given as the input. The latter special case is a trivial instance; however, this case is intuitively considered to be tough for this algorithm, and hence, it is expected that the proposed algorithm runs in $O(n)$ expected time for most of the practical instances, although the worst case running time is $O(n^2)$ for any instance.

Keywords: common intervals of permutations, common subtrees, evolutionary tree, genetic algorithm, linear time algorithm, random permutations, randomized algorithm, Monge property, subtour exchange crossover.

[♠]Department of Systems Science, Tokyo Institute of Technology, 2-12-1 Oh-okayama, Meguro-ku, Tokyo 152, Japan. E-mail: `uno@is.titech.ac.jp`

[♣]Corresponding author: Department of Applied Mathematics and Physics, Graduate School of Engineering, Kyoto University, Kyoto 606-01, Japan. Phone: +81-75-753-5514, E-mail: `yagiura@kuamp.kyoto-u.ac.jp`

1 Common Intervals

Two permutations σ_A and σ_B of set $N = \{1, \ldots, n\}$ are given as the input, where $\sigma_A(i) = j$ (or $\sigma_A^{-1}(j) = i$) denotes that j is the i-th element of σ_A (σ_B is similarly defined). Let $[x, y]$ denote the index set $\{x, x+1, \ldots, y\}$. We call a pair of intervals $([x_A, y_A], [x_B, y_B])$ $(1 \leq x_A < y_A \leq n, 1 \leq x_B < y_B \leq n)$ a *common interval* if it satisfies

$$\{\sigma_A(i) \mid i \in [x_A, y_A]\} = \{\sigma_B(i) \mid i \in [x_B, y_B]\}.$$

The length of a common interval $([x_A, y_A], [x_B, y_B])$ is defined to be $y_A - x_A + 1$.

Some genetic algorithms based on common intervals have been proposed for sequencing problems (e.g., traveling salesman problem, single machine scheduling problem, etc.) and have exhibited good prospect [1, 6, 7, 12].

In this paper, we consider enumeration of all common intervals of length 2 to n. Three algorithms are proposed, which are improved versions of a simple $O(n^2)$ time algorithm proposed in [11]:

1. A simple $O(n^2)$ time algorithm (called LHP), whose expected running time becomes $O(n)$ for two randomly generated permutations.
2. A practically fast $O(n^2)$ time algorithm (called MNG) using the reverse Monge property.
3. An $O(n+K)$ time algorithm (called RC), where K ($\leq \binom{n}{2}$) is the number of outputs.

It is also shown that the expected number of common intervals of length 2 to $n-2$ for two random permutations is $2+O(n^{-1})$. This implies that the expected number of common intervals of length 2 to n is $O(1)$, since the number of common intervals of length $n-1$ or n is at most 3. This result gives a reason for the phenomenon that the expected time complexity $O(n)$ of the algorithm LHP is independent of K. We also give an example for which both LHP and MNG require $\Omega(n^2)$ time, although $K = O(n)$.

Among the three algorithms proposed in this paper, RC is most desirable from the theoretical point of view, but is quite complicated compared to LHP and MNG. Therefore, it is possible that RC is slower than the other two algorithms in some cases. For this reason, computational experiments for various types of problems with up to $n = 10^6$ are conducted. The results indicate that

1. LHP and MNG are much faster than RC for two randomly generated permutations (e.g., LHP is about 13 times faster than RC).
2. MNG is rather slower than LHP for random inputs; however, there are cases that LHP requires $\Omega(n^2)$ time, but MNG runs in $o(n^2)$ time and is faster than both LHP and RC.

The details of the algorithms and a recommendation about the use of the three algorithms is in [10].

These results are applicable to the similar problem defined on two cyclic permutations [11].

2 Common Subtrees

Two rooted trees T_A and T_B are given as the input, each of which has n leaves labeled with $1, 2, \ldots, n$. A subtree $T_A(u)$ is defined to be the subgraph of T_A induced by u and all descendants of u. Let $L_A(u)$ be the set of labels of the leaves in $T_A(u)$. $T_B(u)$ and $L_B(u)$ are similarly defined. We call a pair of subtrees $(T_A(u), T_B(v))$ a *common subtree* if it satisfies

$$L_A(u) = L_B(v),$$

where u and v are neither a root nor leaf. We assume that every inner vertex of T_A or T_B has at least two children, so that the number of inner vertices is $O(n)$.

Genetic algorithms based on common subtrees are proposed for VLSI design [5]. Common subtree also has an application in evolutionary trees for species sets, which are used in biology. Unfortunately, there are many proposals for constructing evolutionary trees, which are then compared to form consensus. The number of common subtrees is one of the basic measures for consensus [4, 8, 9], among others [2].

The proposed algorithm is based on the following observation. Let σ_{T_A} be the permutation of leaf labels of T_A defined by the order where they are scanned by depth-first search, in which the left to right order of choosing the children of each inner vertex is determined arbitrarily. Let $l_A(u)$ (resp., $r_A(u)$) be the label of the left (resp., right) most leaf of $T_A(u)$. σ_{T_B}, $l_B(u)$ and $r_B(u)$ are similarly defined. Then $(T_A(u), T_B(v))$ is a common subtree if and only if $([\sigma_{T_A}^{-1}(l_A(u)), \sigma_{T_A}^{-1}(r_A(u))], [\sigma_{T_B}^{-1}(l_B(v)), \sigma_{T_B}^{-1}(r_B(v))])$ is a common interval of two permutations σ_{T_A} and σ_{T_B}. Note that there are common intervals of σ_{T_A} and σ_{T_B} that do not correspond to any subtrees.

The basic framework of the algorithm is as follows.

1. Apply depth-first search to T_A and T_B, choosing randomly the order of the children at each inner vertex. Denote the two permutations of the leaf labels of T_A and T_B by σ_{T_A} and σ_{T_B}, respectively.

2. Enumerate all common intervals of σ_{T_A} and σ_{T_B} one by one, and if the two intervals corresponding to each common interval define subtrees of T_A and T_B, respectively, then output the corresponding pair of subtrees.

Step 1 can be executed in $O(n)$ time. We can check in Step 2 if an interval of σ_{T_A} (resp., σ_{T_B}) defines a subtree of T_A (resp., T_B) in $O(1)$ worst case time by using the data structure called perfect hash [3], which can be constructed in $O(n)$ expected time and in $O(n^2)$ worst case time.

Let $K(\sigma_{T_A}, \sigma_{T_B})$ be the number of common intervals of two permutations σ_{T_A} and σ_{T_B}. $K(\sigma_{T_A}, \sigma_{T_B})$ may be the dominating factor of the running time of our algorithm. Note that $K(\sigma_{T_A}, \sigma_{T_B}) = \binom{n}{2}$ in the worst case, although the number of common subtrees is $O(n)$. It is also noted that the result about the expected number of common intervals for two random permutations stated in Section 1 is not applicable in this case, since the probability space is different. We show that the expected value of $K(\sigma_{T_A}, \sigma_{T_B})$ is $O(n \log^2 n)$ if the given two trees are binary and the depth of them is $\log_2 n$. We also show that the expected value of $K(\sigma_{T_A}, \sigma_{T_B})$ is $O(n)$ if the same two binary trees of depth $\log_2 n$ are given as the input. The latter special case is a trivial instance as the common subtree enumeration problem. However, in this case, $K(\sigma_{T_A}, \sigma_{T_B}) = \binom{n}{2}$ if we do not randomize the children order of each inner vertex, which is the largest possible value of $K(\sigma_{T_A}, \sigma_{T_B})$. Hence, this is considered to be a tough instance for our algorithm. Therefore, we believe that the expected value of $K(\sigma_{T_A}, \sigma_{T_B})$ is small (e.g., $O(n)$) for most of the practical instances, although theoretical results are limited to the above special cases.

Actually, it is observed by computational experiments on some types of randomly generated trees with up to $n = 10^6$ that the average value of $K(\sigma_{T_A}, \sigma_{T_B})$ is $O(n)$ for all the tested instances, in which trees of depth $\Omega(n)$ are included.

If we use the algorithm RC to enumerate common intervals in Step 2, the expected running time of the above algorithm is $O(n + K(\sigma_{T_A}, \sigma_{T_B}))$, which is $O(n \log^2 n)$ if the given two trees are binary and the depth of them is $\log_2 n$, and $O(n)$ if the same two binary trees of depth $\log_2 n$ are given as the input. The worst case running time is $O(n^2)$ for any input, since $K(\sigma_{T_A}, \sigma_{T_B})$ is $O(n^2)$ in general.

Similar algorithms are applicable to similar problems defined on two *unrooted* trees, in which two connected components defined by deleting an edge are considered as subtrees.

References

[1] R. M. Brady, Optimization Strategies Gleaned from Biological Evolution, *Nature*, **317** (1985), 804–806.

[2] M. Farach and M. Thorup, Sparse Dynamic Programming for Evolutionary-Tree Comparison, *SIAM J. Comput.*, **26** (1997), 210–230.

[3] M.L. Fredman, J. Komlós and E. Szemerédi, Storing a Sparse Table with $O(1)$ Worst Case Access Time, *J. ACM*, **31** (1984), 538–544.

[4] M.D. Hendy, C.H.C. Little and D. Penny, Comparing Trees with Pendant Vertices Labeled, *SIAM J. Appl. Math.*, **44** (1984), 1054–1065.

[5] S. Koakutsu, Private Communication.

[6] S. Kobayashi, I. Ono and M. Yamamura, An Efficient Genetic Algorithm for Job Shop Scheduling Problems, *Proc. 6th ICGA*, 1995, pp. 506–511.

[7] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer, Evolution Algorithms in Combinatorial Optimization, *Parallel Computing*, **7** (1988), 65–85.

[8] D. Penny and M.D. Hendy, The Use of Tree Comparison Metrics, *Systematic Zoology*, **34** (1985), 75–82.

[9] F.J. Rohlf, Consensus Indices for Comparing Classifications, *Mathematical Biosciences*, **59** (1982), 131–144.

[10] T. Uno and M. Yagiura, Fast Algorithms to Enumerate All Common Intervals of Two Permutations, Technical Report #96015, Department of Applied Mathematics and Physics, Graduate School of Engineering, Kyoto University, (1996).

[11] M. Yagiura, H. Nagamochi and T. Ibaraki, Two Comments on the Subtour Exchange Crossover Operator (in Japanese), *J. Japanese Society for Artificial Intelligence*, **10** (1995), 464–467.

[12] M. Yamamura, T. Ono and S. Kobayashi, Character-Preserving Genetic Algorithms for Traveling Salesman Problem (in Japanese), *J. Japanese Society for Artificial Intelligence*, **7** (1992), 117–127.