2004－MPS－50　(1)
2004／6／22

インスタンスの進化に基づくTSPのための発見的解法の改良

藤田　聡　　梅實真一郎

広島大学 大学院工学研究科 情報工学専攻
〒739-8527　東広島市鏡山１丁目４－１

概要　本稿では「与えられたインスタンスを簡単なインスタンスに進化させて解く」という考え方に基づいた、ユークリッド平面上のTSPに対する発見的解法の提案と評価を行う。提案手法は1998年にPapadimitriouとSideriによって提案された手法の改良になっている。提案手法の効果を評価するために行った実験の結果、提案手法を用いることで進化の効率が向上し，従来手法に比べてよりよい解がより高速に求められることが明らかとなった。

キーワード　進化的計算, TSP, エネルギー空間のランドスケープ

An Improved Heuristic for Solving TSP Based on the Evolution of Easy Instances

Satoshi Fujita and Shin'ichiro Umezane

Department of Information Engineering
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima, 739-8527, JAPAN

Abstract　In this paper, we propose a heuristic scheme for solving TSP in the Euclidean plane based on the notion of "evolution of easy instances." The proposed scheme is an improvement of the scheme proposed by Papadimitriou and Sideri in 1998. We conducted several experiments to evaluate the goodness of the schemes. The result of experiments indicates that our scheme really improves the accuracy of solutions compared with the previous one.

Keywords　Evolutionary computation, TSP, Landscape of the energy space.

1　Introduction

Evolutionary computation (EC, for short) is a heuristic scheme for solving inherently hard optimization problems. The basic idea of EC is to imitate livings that could evolve themselves into the given environment. For example, in the genetic algorithm (GA) that is known as a typical EC scheme [1, 5, 6], such an evolution is controlled by the way of selecting individuals that could survive in the next generation, and by the rule of generating individuals from a collection of survived ones [2]. In general EC schemes such as GA, each individual is associated with a solution to the given instance, that is encoded as a "gene" in an appropriate manner. For example, in solving the travelling salesman problem (TSP), such a gene is generally designed so as to represent the visit order of cities and the total length of the corresponding tour.

On the other hand, it is widely believed that livings in the real-world would take different ways of evolution if they were placed in different environments. From the viewpoint of mathematical optimization, such a phenomenon could be regarded as a dynamic change of the given instance as is frequently observed in many online problems such as the cache replacement protocols. In [4], Papadimitriou and Sideri initiated the study of the evolution of easy instances, and proposed a heurstic scheme for solving TSP based on that

idea. The objective of the current paper is to extend the observations made by them, and to develop a new heuristic scheme to evolve easy instances in a more efficient manner. As a concrete problem, we will focus our attention to TSP in the Euclidean space as was did in [4], by the following reasons: 1) we could easily realize continuous modification of the instances, 2) we could effectively visualize the modification of instances that would help to understand the underlying evolution processes, and 3) a fair comparison of the schemes would be possible by using a standard benchmark set such as TSPLIB [7]. The proposed method is evaluated by using several instances drawn from TSPLIB. The result of experiments implies that our scheme really improves the accuracy of solutions compared with the previous scheme, and it efficiently realizes evolution of instances so as to lead to a nearly optimal solution.

The remainder of this paper is organized as follows. In Section 2, after reviewing the Papadimitriou's scheme, a new heuristic is proposed, whose performance is experimentally evaluated in Section 3. Section 4 concludes the paper with directions for future research.

2 Scheme

2.1 Preliminaries

Let V denote a set of vertices and $\mathcal{C}$ be the set of all Hamiltonian cycles over V (note that $|\mathcal{C}| = (|V|-1)!$). Let $\mathcal{F}$ denote the set of all functions from V to the Euclidean plane, and $\Phi_{C,f}$ denote the cost of cycle $C \in \mathcal{C}$ under function $f \in \mathcal{F}$. Under such a model, problem TSP could be formulated as a problem of finding C^* that satisfies

$$\Phi_{C^*,f_0} = \min_{C \in \mathcal{C}} \Phi_{C,f_0} \tag{1}$$

for given initial function f_0 ($\in \mathcal{F}$). In conventional schemes for solving TSP, initial function f_0 is given as an input and does not change during the execution of the algorithm. However, the complexity of TSP depends on the difficulty of the class of given instances. In fact, if $\mathcal{F}'$ ($\subset \mathcal{F}$) is a set of functions that maps vertices in V to the points on a circle in the Euclidean plane, TSP could be solved in a constant time for every instance in $\mathcal{F}'$.

In this section, we describe a heuristic scheme for solving TSP based on the heuristic transformation to "easy" instances [4]. More concretely, it first transforms the given instance f_0 into another instance f' that could be solved in a rather easy way by an evolution process, and constructs the solution to the original instance by regarding the visit order in the solution to instance f' as the solution to f_0.

2.2 Outline

The algorithm first picks up p random initial tours from $\mathcal{C}$. It then locally improves each tour by repeating 2-opt under initial function f_0; let $C_1, C_2, \ldots, C_p$ be p local optima obtained by the local improvements. In what follows, we call a pair (f, C) as an **individual**, where f is a function and C is a tour, and call a collection of p individuals a **pool**. Thus, the current contents of the pool could be represented as $\{(f_0, C_i) : 1 \leq i \leq p\}$.

Next, it repeats the following sequence of operations for a predetermined, say ℓ, times:

1. Select q individuals from the current pool according to a **selection rule**.
2. Generate new individuals from selected ones according to a **generating rule**.
3. Transform generated individuals to local optima by applying 2-opt repeatedly, and let them be a set of individuals in the next generation.

Finally, it outputs a shortest tour constructed in the above procedure as the solution to the given instance, where the tour length is measured by the cost under the initial function f_0.

2.3 Papadimitriou's Method

The selection and generating rules proposed in [4] are described as follows:

Selection Rule: Let $n_{i,j}$ denote the number of individuals in the current pool that contains edge (i, j) in the corresponding tour. The

selection rule selects q individuals in a nonincreasing order of the fitness, where the fitness of (f, C) is defined as

$$\sum_{(i,j)\in C} (n_{i,j})^{\alpha}, \tag{2}$$

where $\alpha > 1$ is an appropriate parameter. This definition of the fitness is based on the intuition such that an individual is well adapted to the environment when it contains as many commonly used edges as possible.

Generating Rule: The first child of a selected individual (f, C) is reserved for a copy of the parent to avoid possible degradation of the accuracy of solutions. The rule for generating the remaining $p/q - 1$ children is described as follows: 1) randomly and independently select vertices in V with the same probability $1/|V|$, and 2) the coordinate of the selected vertex u, i.e., $f(u)$, is moved to the X- and Y- coordinates independently according to a normal distribution with mean zero and standard deviation σ.

2.4 Proposed Method

Selection Rule: In the proposed method, the selection of individuals that could generate children in the next generation, is designed to satisfy the following two requirements: 1) a variety of individuals could be maintained, and 2) individuals with short tours are selected with high probability. The selection rule consists of two phases: i.e., the selection of candidates and the selection of individuals from the nominated candidates.

In the first phase, it constructs a set of candidates for each parent (f, C) of individuals in the following manner:

1. Let k be the number of children of (f, C) with a tour shorter than that of C under the initial function f_0.
2. If $k \leq 1$, an arbitrary individual with a shortest tour is randomly selected as a candidate; otherwise, an arbitrary individual is selected for each length shorter than C (i.e., candidates generated from the same parent must have distinct length).

Table 1: Comparion of selection rules ("A" is a degrading rate [%], and "B" represents the average execution time of a trials [sec]).

	Previous[4]		Proposed	
instance	A	B	A	B
st70	1.534	1.79	0.000	2.26
eil76	2.707	1.84	0.864	2.38
pr76	2.203	2.02	0.134	2.56
kroA100	3.325	4.08	0.000	5.03
kroC100	3.524	4.52	0.015	5.45
kroD100	4.256	4.53	0.567	5.44
eil101	4.836	3.33	1.562	4.13
lin105	0.569	4.79	0.395	5.64
ch130	2.886	9.09	1.117	10.75
ch150	5.820	13.66	0.919	15.69

All individuals in the initial pool are selected as the candidates since they have no parents. Let $\mathcal{P}$ be the set of all candidates.

In the second phase, it picks up q individuals from $\mathcal{P}$ in a nondecreasing order of the tour length, as the parents of individuals in the next generation.

Generating Rule: Individuals in the next generation could be obtained by repeatedly applying 2-opt after moving several points in the instance, in a similar way to the previous scheme [4]. A key idea of the proposed generating rule is to maintain an edge in a tour as much as possible if it is commonly used in many individuals contained in the current pool. It is in contract with the previous scheme that randomly selects vertices and their new coordinate values with a uniform probability. A concrete description of the proposed generating rule is omitted in this extended abstract.

3 Experiments

We conducted a series of experiments to evaluate the goodness of the proposed scheme. In the experiments, we used ten instances extracted from TSPLIB, and fixed parameters used in the schemes as follows: Parameters concerned with the number of individuals are

fixed as $p = 60$ and $q = 15$, and each trial consists of $\ell = 60$ generations. The environment of the experiments is as follows: CPU: Pentium4 2.53GHz, Memory: 1024 MB, and all programs are written in C.

3.1 Selection Rule

At first, we examine the effect of the proposed selection rule by comparing it with the rule proposed in [4]. To make the difference of selection rules clear, in the experiments, we fixed the other part of the scheme as in [4]. We conducted 20 trials for each of the ten instances, and measured the accuracy of the best solution obtained during the 20 trial, as well as the average execution time of a trial.

Table 1 shows the result. The degrading rate shown in the table represents how much the resultant solution increases compared with an optimum solution, e.g., if the resultant solution is 70 and an optimum solution is 50, the degradation ratio is calculated as $(70-50)/50\times 100 = 40$ [%]. As is shown in the table, our selection rule improves the accuracy of solutions generated by the previous selection rule without significantly increasing the average execution time; e.g., the accuracy is improved by 5% in the best case, and we could obtain an optimum solution for two instances, that was zero under the previous rule.

3.2 Generating Rule

Next, we examine the effect of the proposed generating rule. In the experiments, we compare the accuracy of solutions and the average execution time of a trial under the same selection rule proposed in the previous section. The result implies that the proposed rule really improves the accuracy of solutions, and moreover, it obtains an optimum solution for four instances that was two under the previous rule.

3.3 Evolution of the Instances

In this and the next sections, we evaluate how given instances could evolve under the proposed scheme. As the first step, we count the number and the distribution of local optima under initial and final mappings by providing 2000 random tours, where a tour is called local optima if it could not be improved by 2-opt. The result implies that although the number of local optima does not change for several instances, it reduces to a half for several instances.

4 Concluding Remarks

In this paper, we proposed a heuristic scheme for solving TSP based on the notion of evolution of easy instances, that is an improvement of the scheme proposed in [4].

An important future problem is to refine the proposed scheme as a heuristic scheme for solving TSP so as to be competitive to other heuristics such as Lin-Kernighan method [3]. Another future problem is to apply the notion of evolution of easy instances to the other problems.

References

[1] L. Davis, *Handbook of Genetic Algorithms,* Van Nostrand Renhold, 1990.

[2] J. H. Holland, *Adaptation in natural and artificial systems*, The University of Michigan Press, 1975; MIT Press, 1992.

[3] S. Lin, B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.* 21:498-516, 1973.

[4] C. H. Papadimitriou, M. Sideri, "On the evolution of easy instances," manuscript, 1998. URL: `http://www.cs.berkeley.edu/~ christos/`

[5] R. Tanese, "Distributed genetic algorithms," *Proc. 3rd Int'l Conf. on Genetic Alg.*, pp.434–439, 1989.

[6] D. Thierens, D. E. Goldberg, "Elitist recombination: an integrated selection recombination GA," *Proc. 1st IEEE Conf. on Evolutionary Comp.,* pp.508–512, 1994.

[7] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`