Applied Soft Computing 13 (2013) 3008–3020

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

PREACO: A fast ant colony optimization for codebook generation

Chun-Wei Tsai[a], Shih-Pang Tseng[b], Chu-Sing Yang[c], Ming-Chao Chiang[b,*]

[a] Department of Applied Informatics and Multimedia, Chia Nan University of Pharmacy & Science, Tainan 71710, Taiwan, ROC
[b] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, ROC
[c] Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan, ROC

ARTICLE INFO

Article history:
Received 1 January 2012
Received in revised form 5 January 2013
Accepted 16 January 2013
Available online 29 January 2013

Keywords:
Ant colony optimization
Codebook generation problem
Pattern reduction

ABSTRACT

This paper presents an effective and efficient method for speeding up ant colony optimization (ACO) in solving the codebook generation problem. The proposed method is inspired by the fact that many computations during the convergence process of ant-based algorithms are essentially redundant and thus can be eliminated to boost their convergence speed, especially for large and complex problems. To evaluate the performance of the proposed method, we compare it with several state-of-the-art metaheuristic algorithms. Our simulation results indicate that the proposed method can significantly reduce the computation time of ACO-based algorithms evaluated in this paper while at the same time providing results that match or outperform those ACO by itself can provide.

1. Introduction

Vector quantization (VQ) is an important technique for not only image compression but also pattern recognition [1,40], pattern compression [42,43], speech recognition [37], and face detection [61]. Generally, three phases—codebook generation, encoding, and decoding—and a codebook are employed by VQ to encode and decode a signal [30]. Because the codebook generation phase may strongly affect the performance of VQ, it has become an important research topic for many years. Generalized Lloyd algorithm (GLA) [44] is one of the most widely used methods in codebook generation because it is simple and easy to implement. Also known as *k*-means [48], the basic idea of GLA is to find a codebook that minimizes the distortion between the training patterns and the codewords.

To improve the quality of VQ, many heuristic-based algorithms [39,12,25,25,19] have been proposed for the codebook generation problem (CGP) over the past one and a half decades or so. To reduce the computation time of CGP, two kinds of approaches have been taken. The first kind of approach is to employ a structured codebook to speed up the VQ process, such as tree-structured codebook [10,54,68]. The second kind of approach is to reduce the number of codewords compared so as to reduce the search complexity of the partitioning step of GLA, such as codeword reduction [11,16,50] and triangle inequalities [31]. Of course, many other fast algorithms [67] have also been proposed to reduce the computational time of GLA, such as dimensional reduction [18].

1.1. The motivation

Although being able to provide a better codebook, most heuristic-based algorithms for the CGP normally take a much longer computation time than deterministic algorithms. This is the main reason why it is important to reduce the computation time of heuristic-based algorithms when they are applied to the CGP. Another reason is that most fast methods are designed for GLA-based algorithms [67]. Although some researches [31] focus on cutting down the number of similarity comparisons so as to reduce the computation time of CGP, they are not designed for heuristic-based algorithms.

As depicted in Fig. 1, subsolutions of ant colony optimization (ACO) for the CGP[1] do not necessarily reach their *final state* at the same time on the convergence process. This figure eventually shows that more than 90% of the image blocks are assigned to the same codewords after 100 iterations which implies that many of the computations of ACO for the CGP after 100 iterations are essentially redundant. These results also tell us that if subsolutions that have reached the final state can be eliminated on the convergence process, then the computation time of ACO for the CGP can be significantly reduced. Hence, this paper presents an effective and efficient method for speeding up ACO in solving the CGP which works by eliminating computations that are essentially redundant during the convergence process of ACO.

* Corresponding author. Tel.: +886 7 5252000x4321; fax: +886 7 5254301.
E-mail address: mcchiang@cse.nsysu.edu.tw (M.-C. Chiang).

[1] As far as ACO for the CGP is concerned, a subsolution is defined as the assignment of a pattern to a codeword. Otherwise, it is as defined in [21].

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.01.017

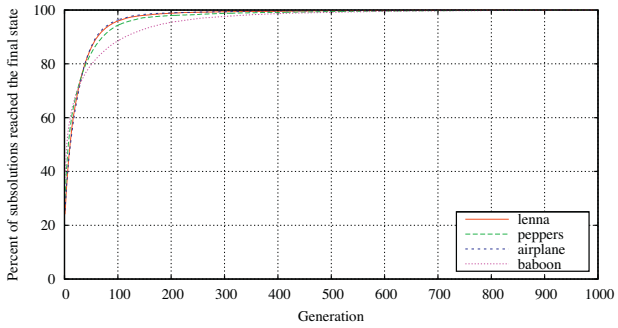

Fig. 1. Example showing the percentage of subsolutions of ant colony system (ACS) [21] for the CGP arrived at the final state during the convergence process when applied to a set of 512 × 512 grayscale images with the codebook size set equal to 256.

1.2. The contribution

Unlike the pattern reduction methods described in [13,64] that are designed for single-solution-based algorithms and consider *patterns that are close to the centroid of each cluster* as patterns to be eliminated, the algorithm described herein is designed for population-based algorithms and considers *patterns that have reached a consensus by voting* as patterns to be eliminated. The proposed algorithm is not just an extension of [65] that is designed for solving the traveling salesman problem (TSP); rather, the proposed algorithm is designed for the CGP, which uses one-iteration *k*-means as the local search operator to improve the final result of ACO for the CGP. The main contributions of the paper can be summarized as follows:

1 This paper first points out that there are many redundant computations on the convergence process of ACO for the CGP.
2 A fast algorithm adapted from [65], called pattern reduction enhanced ant colony optimization (PREACO), is then presented to reduce the computation time of ACO in solving the CGP while at the same time providing a result that either matches or outperforms the result ACO by itself can obtain.
3 This paper shows that the proposed algorithm can be easily plugged into ACO for reducing its computation time in solving a very different combinatorial optimization problem—CGP instead of TSP as described in [65]. These two applications further show that the notion of pattern reduction can be applied to other combinatorial optimization problems, such as bin packing problem and job shop scheduling problem.
4 A detailed analysis of parameter settings is also given to show the impact they may have on the performance of the proposed algorithm.

1.3. The organization

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to the ACO and its variants. Section 3 first defines the CGP and then discusses how to apply ACO to this problem. Section 4 describes in detail the proposed algorithm. The simulation results are discussed in Section 5. Also discussed in this section are the datasets evaluated and the settings of the parameters. We conclude our work in Section 6.

2. Related work

2.1. The ant colony optimization

Over the past two and a half decades or so, metaheuristics [29,55,6,5] have been widely used in solving complex problems, such as the TSP and the CGP. Several new population-based algorithms have been proposed, which consist of ant colony optimization [22,21], particle swarm optimization (PSO) [35], bees algorithm [53], wasps [7], and other animal societies [3]. These algorithms are collectively referred to as the swarm intelligence (SI) [2,36,24]. As one of the top-performing methods of SI, ACO has been successfully applied to solve a wide range of problems [3,23]. The key concept of ACO is to use a colony of cooperative artificial ants to find "good" paths between their colony and a source of food in such a way that these good paths can be considered as good solutions of a discrete optimization problem [23].

Algorithm 1. Outline of the ACO

```
1  Initialization()
2  While the termination criterion is not met
3    s = SolutionConstruction()
4    τ = PheromoneUpdate()
5    LocalSearch(s) /* optional */
6  End
7  Output the result.
```

As shown in Algorithm 1, the very first step ACO takes is to initialize all the parameters. After that, the main procedure on lines 2 through 6 is repeated until the stop condition is met. The main procedure is composed of two required operators *SolutionConstruction*() and *PheromoneUpdate*() and one optional operator *LocalSearch*().

2.1.1. The solution construction operator

As the name suggests, the solution construction operator on line 3 is performed by each ant to construct its routing path (i.e., a solution of the problem in question) based on the pheromone and heuristic information available. More precisely, the routing path of each ant is constructed edge by edge until the routing path is completely established. The probability of selecting edge e_{ij}, i.e., the probability of an ant at subsolution i selecting j as the next subsolution (i.e., subsolution $i+1$), is defined as

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in \mathcal{N}_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} & \text{if } j \in \mathcal{N}_i^k, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{N}_i^k$ denotes the set of candidate subsolutions (i.e., subsolutions that can be selected by ant k at subsolution i); τ_{ij} and η_{ij} denote, respectively, the pheromone value and the heuristic value associated with e_{ij}.

2.1.2. The pheromone update operator

Again, as the name suggests, the pheromone update operator on line 4 is used to update the pheromone values that are associated with the search experience of ACO. That is, the pheromone values are associated with the routing path of an ant, which may in turn increase the probability of other ants choosing the same path (edges). On the other hand, the pheromone evaporation is used to avoid rapid convergence to a local region [23]. The pheromone update operator employed for updating the pheromone value of each edge e_{ij} is defined as

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}^k, \tag{2}$$

$$\Delta\tau_{ij}^k = \frac{1}{L^k}, \tag{3}$$

where L^k denotes the quality of the solution created by ant k; $\rho \in (0, 1]$ denotes the evaporation rate.