Information Sciences 199 (2012) 31–42

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem

Chen Ling [a,b,*], Sun Hai-Ying [a], Wang Shu [a]

[a] Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009, China
[b] State Key Lab of Novel Software Tech, Nanjing University, Nanjing 210093, China

ARTICLE INFO

Article history:
Received 1 February 2009
Received in revised form 5 October 2011
Accepted 26 February 2012
Available online 5 March 2012

Keywords:
Ant colony optimisation
Parallel processing
Convergence
Travelling salesman problem

ABSTRACT

An adaptive parallel ant colony optimisation (PACO) algorithm on massively parallel processors (MPPs) is presented. In the algorithm, we propose a strategy for information exchange between processors that makes each processor choose a partner to communicate with and update their pheromone adaptively. We also propose a method of adaptively adjusting the time interval for the exchange of information according to the diversity of the solutions, to increase the quality of the optimisation results and to avoid early convergence. The analysis and proof of the convergence of the PACO algorithm is presented. Experimental results of the TSP confirm our theoretical conclusions and show that our PACO algorithm has a high convergence speed, high speedup and high efficiency.

1. Introduction

Ant colony optimisation (ACO) is an evolutionary-based optimisation algorithm [15] that was proposed by Dorigo et al. [13,14]. ACO belongs to the class of biologically inspired heuristics [4,23,29,34]. It imitates the cooperative behaviour of ant colonies to solve combinatorial optimisation problems. Using very simple communication mechanisms, a group of real ants is able to find the shortest path between any two points. During their trips, a chemical trail called a pheromone is left on the ground. The role of this trail is to guide the other ants towards their destination. For each ant, the path is chosen according to the quantity of pheromone. Furthermore, this chemical substance has a decreasing action over time, and the quantity left by one ant depends on the amount of food found and the number of ants following this trail. Inspired by this effect of a real ant colony, Dorigo et al. [13,14] used artificial ants in ACO to emulate real ants in the process of seeking food and exchanging information. ACO has achieved widespread success in solving different optimisation problems such as TSP [26], machine learning and reasoning [32], sequential ordering [17], job-shop scheduling [5,35], network design [42], frequency assignment [27], network routing [24], data mining [7,9,18,25], digital IIR filter designing [22], robotics [30] and other combinational optimisation problems [8,20,36,37].

The availability of parallel architectures at low cost has increased the amount of interest in the parallelisation of ACO. Though ACO usually determines a satisfactory solution for a problem within a reasonable amount of time, it becomes more difficult to speed up the algorithm when the complexity of the problem increases. ACO algorithm has intrinsic parallelism, which is very suitable for implementation on large-scale parallel computers.

To parallelise the ant colony algorithm, it is more important to modify the structure of ACO to obtain a better optimisation effect than to re-implement the sequential ACO into a parallelisation schema. The modification of the ant colony structure to

* Corresponding author at: Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009, China. Tel.: +86 514 87978320; fax: +86 514 87887937.
E-mail address: lchen@yzcn.net (L. Chen).

0020-0255/$ - see front matter © 2012 Published by Elsevier Inc.
doi:10.1016/j.ins.2012.02.055

fit the parallel computational model involves three aspects: (1) dividing a single ant colony of a sequential ACO into several mutually independent subordinate colonies; (2) controlling and managing the information exchange between the sub colonies; and (3) determining the time interval for the information exchange between the sub ant colonies. Different methods of colony dividing and information exchange produce different parallel ant colony algorithms. Our goal in parallelisation is to obtain a high speedup and efficiency while the convergence and optimisation quality are maintained or even improved.

Some results on parallel ant colony algorithms have been reported recently [1,3,6,10–12,16,26,28,31,33,39,41]. When developing parallel ant colony algorithms, intuition suggests the adoption of the "island model" approach from parallel genetic algorithms, where the exchange of information plays a major role. Solutions, pheromone matrices, and parameters have been tested as the objects of such exchanges. Ellabib et al. [16] proposed three different strategies, which are based on a weighting scheme. They also developed a search assessment technique based on a team consensus methodology to study the influence of these strategies on the search behaviour. Manfrin et al. [26] studied the impact of communication when parallelising a high-performing ACO algorithm for the travelling salesman problem using message-passing libraries. Specifically, they examined synchronous and asynchronous communications on different interconnection topologies. Antony et al. [1] introduced an asynchronous parallel Max–Min ant colony algorithm that is associated with a local search strategy. Their algorithm was tested on the TSP using the parallel computer Cray-T3E. Randall [31] introduced a synchronous parallel strategy that assigns only one ant on each processor. By modifying the classical ACO, Merkle [28] first proposed a parallel ant colony algorithm on reconfigurable processor arrays. The running time of the algorithm is quasi-linear with the problem size n and the number of ants on a reconfigurable mesh with n^2 processors. Blum and Dorigo [6] advanced a parallel ant colony algorithm on the hyper-cube architecture by modifying the rule of updating the pheromone by limiting the pheromone values within the range of [0,1]. This new approach enhances the ability of the ant colony algorithm to address complicated objective functions theoretically and practically. In [41], Yang and Yu presented a coarse-grain parallel ant colony algorithm for bus network design. They developed a new pheromone-updating strategy called ant-weight, which can adaptively adjust the ants' path-searching activities based on the objective function. Benkner et al. [3] analysed communication strategies in parallel ACO and concluded that the communication of the whole pheromone matrix leads to worse solution quality as well as more runtime, while the exchange of best-so-far and elite solutions produces the most optimum best results with respect to the solution quality.

To parallelise the ant colony algorithm, the most important factors are the pattern and the time interval for the information exchange between the processors. These factors affect not only the convergence speed of the algorithm but also its optimisation performance. In the algorithms of [1,16,26,31,40], the globally best solution is computed and broadcasted to all of the processors in the information exchange. Then, every processor updates the pheromone matrix according to the best solution. This method of information exchange could probably create some similar solutions across different processors, which cause large amounts of pheromones on some trails. These trails could be considered to be the "optimum solution" and thus reduce the search diversity and efficiency of the processors. In addition, in the algorithms of [1,16,26,31], the processors exchange information in a constant time interval. Although paper [16] acknowledged that this constant time interval for information exchange could affect the optimisation speed, the diversity of the solutions and the convergence of the algorithm, the detailed analysis of the effect and a strategy for improvement have not been provided. Because this constant time interval for information exchange does not take the distribution of the solutions into account, it may influence the diversity of the solutions and the convergence speed. Because of the overhead that is caused by synchronisation and communication, the parallel algorithms mentioned above are not efficient, and their speed of convergence and performance could be improved.

In this paper, we present an efficient adaptive parallel ant colony optimisation algorithm (PACO). We also propose a strategy for information exchange between processors, which makes each processor choose its partner to communicate and update the pheromone adaptively. To increase the optimisation ability and to avoid early convergence, we also propose a method of adaptively adjusting the time interval according to the diversity of the solutions. Convergence of the parallel ant colony algorithm is analysed and proved. These techniques are applied to the travelling salesman problem on the massively parallel processor (MPP) system of Shenteng 1800. Experimental results confirm our theoretical conclusions and show that our PACO algorithm has a high convergence speed, high speedup and high efficiency.

The remainder of the paper is organised as follows. In Section 2, we illustrate the architecture of massively parallel processors (MPPs). Section 3 presents the framework of the parallel ant colony algorithm PACO on MPPs. Inter-processor information exchange strategies are proposed in Section 4. In Section 5, we analyse and prove the convergence of PACO. Experimental results are shown and analysed in Section 6. Section 7 concludes the paper.

2. Massively parallel processors

Our parallel ant colony algorithm is based on the computational model of massively parallel processors (MPPs), which adopt the message-passing method. The architecture of a typical MPP systems is shown in Fig. 1.

All MPP systems [21] use physically distributed memory and some distributed I/O. The whole system consists of a number of processing nodes. Each node has a processor/cache (P/C) and a local memory (LM). There is a local interconnection within a node that connects processors, memories, and I/O devices. Each node is connected to the network through network interface circuitry (NIC). An MPP system has several features, as follows: (1) A commercialised microprocessor in each processing node, and one or more microprocessors in each node; (2) Physically distributed memory, namely, each node has its