Expert Systems with Applications 42 (2015) 390–410

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Ant Colony Extended: Experiments on the Travelling Salesman Problem

Jose B. Escario *, Juan F. Jimenez, Jose M. Giron-Sierra

Arquitectura de Computadores y Automatica, Universidad Complutense de Madrid, Av. Complutense s/n, 28040 Madrid, Spain

ARTICLE INFO

Article history:
Available online 12 August 2014

Keywords:
Ant Colony Optimisation
Swarm intelligence
Self-organisation
Artificial intelligence
Multi-agent system

ABSTRACT

Ant Colony Extended (ACE) is a novel algorithm belonging to the general Ant Colony Optimisation (ACO) framework. Two specific features of ACE are: the division of tasks between two kinds of ants, namely patrollers and foragers, and the implementation of a regulation policy to control the number of each kind of ant during the searching process. In addition, ACE does not employ the construction graph usually employed by classical ACO algorithms. Instead, the search is performed using a state space exploration approach. This paper studies the performance of ACE in the context of the Travelling Salesman Problem (TSP), a classical combinatorial optimisation problem. The results are compared with the results of two well known ACO algorithms: ACS and MMAS. ACE shows better performance than ACS and MMAS in almost every TSP tested instance.

1. Introduction

Recently, the Ant Colony Extended (ACE) has been introduced in Escario, Jimenez, and Giron-Sierra (2012) using, as an application example, the optimisation of ship manoeuvring. With respect to ACO, the new algorithm includes two significant contributions: The use of a state space representation instead of the conventional graph representation, and a partition of the ant population into explorers and foragers ants. It has been shown in Escario et al. (2012) that ACE can tackle problems related with dynamic systems. The target of the present article is to present a detailed description of the algorithm, and to confirm that ACE is comparable with well established ant algorithms when applied to a standard benchmark. Our results show that in most TSP tested cases, ACE outperforms the chosen reference algorithms.

Ant Colony Optimisation (ACO) metaheuristic was originally developed to solve combinatorial optimisation problems (Blum, 2005; Dorigo & Stützle, 2004, 2010). The algorithm is inspired by the foraging activity of ants in the nature and, more specifically, by the pheromone trail used by some ants species for marking paths from food sources to the nest (Goss, Aron, Deneubourg, & Pasteels, 1989).

In ACO, artificial ants build solutions by performing randomised walks in a completely connected graph $G_c = (C, L)$ known as *construction graph*. The nodes of the graph, C, are components for constructing solutions of the combinatorial optimisation problem. The edges of the graph, L, fully connect the nodes. Using the construction graph, the original combinatorial problem can be reduced to a search of the minimum paths in this graph.

Ants move through neighbour nodes applying a stochastic decision policy that makes use of pheromone trails and heuristic information. The pheromone trails accumulate the collective knowledge of the quality of those solutions the ants have found since the start of the algorithm. The heuristic information contains also knowledge about the quality of the solutions, yet heuristic information is problem dependent and it is provided by some source other than the ants. A complete description of ACO metaheuristic can be found in Dorigo and Stützle (2004).

The first ACO algorithm developed was Ant System (AS) (Dorigo, Maniezzo, & Colorni, 1996). It was tested using the Travelling Salesman Problem (TSP) (Johnson & McGeoch, 1997). This classical NP-hard problem has been extensively used as a benchmark for testing optimization algorithms. Ant System proved that the methodology was promising, but it also showed some drawbacks: its performance tends to decrease as the size of the TSP instance increases (Dorigo & Stützle, 2010).

After AS, there were attempts to improve its performance that result in two classical ACO algorithms: Ant Colony System (ACS) (Dorigo & Gambardella, 1997) and Max–Min Ant System (MMAS) (Stützle & Hoos, 2000). These two algorithms have been selected to perform the comparison in the present article.

As already mentioned, ACE includes new features with respect to standard ACO algorithms. These are the use of a state space representation (Allen & Herbert, 1972; Russell & Norvig, 2010), and the partition into to kinds of ants.

* Corresponding author.
E-mail addresses: jbescario@filos.ucm.es (J.B. Escario), juan.jimenez@fis.ucm.es (J.F. Jimenez), gironsi@fis.ucm.es (J.M. Giron-Sierra).

http://dx.doi.org/10.1016/j.eswa.2014.07.054
0957-4174/

```
1: Initialization
2: while termination condition not met do
3:     for ant ∈ Population do
4:         Search step (ant)
5:         if search finished then
6:             if g(ant) < μ then                    ▷ successful search
7:                 Update mean (ant)
8:                 Pheromone update (ant)
9:                 Success (ant)                     ▷ Pop. Dyn. success case
10:            else                                  ▷ unsuccessful search
11:                Failure (ant)                     ▷ Pop. Dyn. failure case
12:            end if
13:            Remove (ant)
14:        end if
15:    end for
16:    Recruitment ()                                ▷ Pop. Dyn. recruitment
17:    Daemon actions                                ▷ optional
18: end while
```

Fig. 1. ACE pseudo-code description: main loop.

The change of the representation opens new possibilities of application, like it was shown with the ship manoeuvring optimisation. Essentially, the idea is to adopt a more flexible representation compared with the conventional graph, while keeping the possibility of tackling problems typically treated with graphs. Actually, in the present article, we demonstrate that ACE can deal with the TSP problem, which is normally represented with graphs. With graphs one has to deal the complete topology of the problem; with the state space you only work with the explored part of the topology. Therefore, ACE is less time consuming.

As said before, ACO combines pheromone trails *and* heuristic information. In ACE instead, ants use pheromone *or* heuristic information. In order to promote exploration, which is convenient to avoid stagnation (Dorigo & Stützle, 2004), ACE uses explorer ants (Patrollers) that can opt for heuristics or for pheromone. At the same time ACE uses also exploitative ants (Foragers) that only employ pheromone. In this way, we take new inspiration from the studies of biological researchers (Bonabeau, Theraulaz, & Deneubourg, 1996; Bonabeau, Dorigo, & Theraulaz, 1999; Camazine et al., 2003; Gordon, 2000, 2002, 2007, 2010). There are, among others, two remarkable features of some ant species: The existence of different roles associated to different tasks and the self-regulation of the organization by direct communication among ants.

The details of the self-organization will be explained in further sections. The Patroller ants tend to increase the dispersion of the search. The foragers ants tend to reduce it. Therefore, the balance of the search is done through the balance of a population, using a similar self-organisation dynamics than real ants. The advantage of this dynamics is that it reduces the number of parameters needed by the algorithm to regulate the search. In fact, ACE uses lower number of parameters than the typical ones present in ACO algorithms. Besides, ACE parameters are rather different than the common ones used in ACO algorithms.

Key	Value
q_0	$[(u_0, \tau_0), (u_1, \tau_1), \ldots]$
q_1	$[(u_1, \tau_1), (u_2, \tau_2), \ldots]$

Fig. 2. ACE pheromone table structure. The symbol q represents a state, u an action and τ a probability value.

The present article includes a short description of the population dynamics. A more extended account can be found in Escario, Jimenez, and Giron-Sierra (2013, chap. 3).

According with Dorigo and Stützle (2004), the main reason to use TSP as a benchmark is its simplicity, allowing to focus the study in the performance of the algorithms. The use of TSP for testing ACE is aimed to this same purpose: To perform a parameters study, and a comparison of ACE performance with two well established ACO algorithms: ACS and MMAS. In addition, the paper presents the pseudo-codes of ACE procedures, with the aim to facilitate the use of ACE to interested readers.

It is opportune to note, that ACS and MMAS are recognised as very good algorithms. In fact, they are commonly used to develop more specialised version of ACO algorithms to solve other problems different from TSP. An actual review of the modifications and applications of ACO can be found in Chandra Mohan and Baskaran (2012).

Another recent trend consists in the combination of ACO algorithms with other optimisations algorithms. There is a large number of these hybrid algorithms, which try to explode the best features of the combined original algorithms. In the context of the TSP problem, when dealing with large instances ACO is usually hybridised with a suitable local search algorithm (Stutzle, 1997). ACO has also been combined, among others, with Genetic algorithms to solve the production scheduling problem (Hecker, Stanke, Becker, & Hitzmann, 2014), with Fuzzy logic to for ACO parameters dynamic adaptation (Valdez, Melin, & Castillo, 2014), with Data Envelopment Analysis for Knowledge sharing Assessment (Kuah, Wong, & Tiwari, 2013), with Tabu Search for K-minimum spanning tree problems (Katagiri, Hayashida, Nishizaki, & Guo, 2012) and Vehicle Routing Problem (Yu, Yang, & Yao, 2011),