Journal of Network and Computer Applications 36 (2013) 504–522

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

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

Swarm optimisation algorithms applied to large balanced communication networks

Eugénia Moreira Bernardino [a], Anabela Moreira Bernardino [a,*], Juan Manuel Sánchez-Pérez [b], Juan Antonio Gómez Pulido [b], Miguel A. Vega Rodríguez [b]

[a] Computer Science and Communication Research Centre, Dept. of Computer Science, School of Technology and Management, Polytechnic Institute of Leiria, Leiria, Portugal
[b] Dept. of Technologies of Computers and Communications, Polytechnic School, University of Extremadura, Cáceres, Spain

ARTICLE INFO

Article history:
Received 30 August 2011
Received in revised form
29 February 2012
Accepted 13 April 2012
Available online 25 April 2012

Keywords:
Computer networks
Terminal Assignment Problem
Swarm Intelligence
Ant Colony Optimisation
Discrete Particle Swarm Optimisation
Artificial Bee Colony

ABSTRACT

In the last years, several combinatorial optimisation problems have arisen in the computer communications networking field. In many cases, for solving these problems it is necessary the use of metaheuristics. An important problem in communication networks is the Terminal Assignment Problem (TAP). Our goal is to minimise the link cost of large balanced communication networks. TAP is a NP-Hard problem. The intractability of this problem is the motivation for the pursuits of Swarm Intelligence (SI) algorithms that produce approximate, rather than exact, solutions. This paper makes a comparison among the effectiveness of three SI algorithms: Ant Colony Optimisation, Discrete Particle Swarm Optimisation and Artificial Bee Colony. We also compare the SI algorithms with several algorithms from literature. Simulation results verify the effectiveness of the proposed algorithms. The results show that SI algorithms provide good solutions in a better running time.

1. Introduction

The Terminal Assignment Problem (TAP) is an important problem in the computer communications networking field. The goal of TAP is to minimise the link cost to form a network by connecting a given set of terminals to a given set of concentrators (Salcedo-Sanz and Yao, 2004; Yao et al., 2004; Khuri and Chiu, 1997). In our work we consider a balanced distribution of terminals among concentrators. Our goal is to minimise the link cost to form a balanced network. In the last decades, many researchers proposed several optimisation algorithms to optimise TAP (see Section 3). Nowadays, we observe an increasing size and consequently an increasing complexity of computer networks and for that reason finding an optimal solution for large instances of TAP continues to be a hard task. TAP is a NP-Hard optimisation problem (Khuri and Chiu, 1997) and for dealing with its difficulty, we propose SI algorithms. The SI algorithms were tested using small instances from literature and 3 large instances.

SI is an artificial intelligence technique involving the study of collective behaviour in decentralised systems. A SI algorithm is initialised with a population (i.e., potential solutions), whose individuals are modified over many iteration steps by imitating the social behaviour of insects or animals, in order to find the optimal solution in the problem solution space. A potential solution "flies" through the search space by modifying itself according to its past experience and its relationship with other individuals in the population and the environment (Kennedy et al., 2001).

Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), Bees Algorithm (BA) and Artificial Bee Colony (ABC) algorithms are some of the most known SI approaches. These algorithms can be used to solve real-world optimisation problems.

Several nature-inspired algorithms have been proposed in the literature to minimise the link cost of balanced networks (see Section 3). Diverse bio-inspired techniques have also been applied to optimise Synchronous Optical Network (SONET) ring loading problems (Bernardino et al., 2011a,b) and SONET design problems (Bernardino et al., 2012). In (Bernardino et al., 2009b; Bernardino et al., 2010e; Bernardino et al., 2011c) the authors considered the use of SI techniques to optimise TAP. Hybrid ACO (HACO) algorithms (Bernardino et al., 2009b; Bernardino et al., 2011c) and BA (Bernardino et al., 2010e) were applied with success to TAP. Currently, we consider a New improved HACO (NHACO) algorithm based in (Bernardino et al., 2011c) (see Section 4.1).

Recently, Wang et al. (2011) proposed a Discrete PSO based on Estimation of Distribution (DPSO-EDA) for solving TAPs and compare the performance of DPSO-EDA with a Hopfield Neural

* Corresponding author. Tel.: +351 967 380 042.
E-mail addresses: eugenia.bernardino@ipleiria.pt (E.M. Bernardino), anabela.bernardino@ipleiria.pt (A.M. Bernardino), sanperez@unex.es (J.M. Sánchez-Pérez), jangomez@unex.es (J.A. Gómez Pulido), mavega@unex.es (M.A. Vega Rodríguez).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.04.005

Network (HNN)-Genetic Algorithm (HNN-GA). A different Discrete PSO (DPSO) algorithm based on Pan et al.'s (2008) DPSO algorithm (see Section 4.2) and an ABC algorithm (see Section 4.3) are also considered in this paper to optimise TAP. In NHACO we incorporate a new mechanism to modify the solutions. In DPSO and ABC we incorporate the LS method proposed by Bernardino et al. (2009b).

Several heuristics have been used to optimise TAP: Local Search (LS) methods, Evolutionary Algorithms (EAs), SI techniques, among others (see Section 3). Note however, that this paper presents the first attempt (to the authors' knowledge) to use the Pan et al.'s DPSO algorithm and the ABC algorithm to optimise TAP in the context of balanced networks. The proposed algorithms are compared with several algorithms from literature. To perform a fair comparison between algorithms we implement several algorithms from literature. We implement those, which can work with an integer representation. For all algorithms studied, we use the same programming language, the same method to encode the solutions and the same fitness function. The results of NHACO, DPSO and ABC are also compared with the results obtained with HNN-GA and DPSO-EDA. We took the results of HNN-GA and DPSO-EDA from Wang et al. (2011).

The SI algorithms have been applied to several combinatorial optimisation problems. Most of them use the binary representation to encode the solutions. Some of them use the continuous representation. In this paper we use an integer representation (see Section 4.4). We prove that discrete SI algorithms can produce good results using this type of representation.

The paper is structured as follows: in Section 2 we describe TAP; in Section 3 we present the previous work; in Section 4 the SI algorithms are described; in Section 5 we present the studied examples; in Section 6 the results are discussed and; in Section 7 we report about the conclusions.

2. Terminal Assignment Problem

TAP tries to determine which terminals will be served by each concentrator (Khuri and Chiu, 1997). To represent this problem we must consider the following aspects: (1) terminals and concentrators have fixed and known locations; (2) the capacity requirements (L) of each terminal are known; (3) each concentrator is limited by the amount of traffic that it can accommodate and; (4) the concentrator capacities (C) and the distances between terminals and concentrators are also known.

In TAP a computer network will connect N terminals, each with L_i demand (weight) to M concentrators, each of C_j capacity. Capacities are given by positive integers and each L_i must be small or equal to min ($C_j,\ldots,C_M$). The terminals $CT_i(x,y)$ and concentrators $CP_j(x,y)$ sites have fixed and known locations placed on a Euclidean grid.

Problem instance:
Terminals — A set N of n distinct terminals $(1,\ldots,n)$;
Weights — A vector L, with the capacity required for each terminal;
Terminal location — A vector CT, with the location (x,y) of each terminal;
Concentrators — A set M of m distinct concentrators $(1,\ldots,m)$;
Capacities — A vector C, with the capacity required for each concentrator;
Concentrators location — A vector CP, with the location (x,y) of each concentrator.

The optimisation goals (see Section 4.5) are to produce feasible solutions, to minimise the distances among concentrators and terminals so as to hold a balanced distribution of terminals and concentrators.

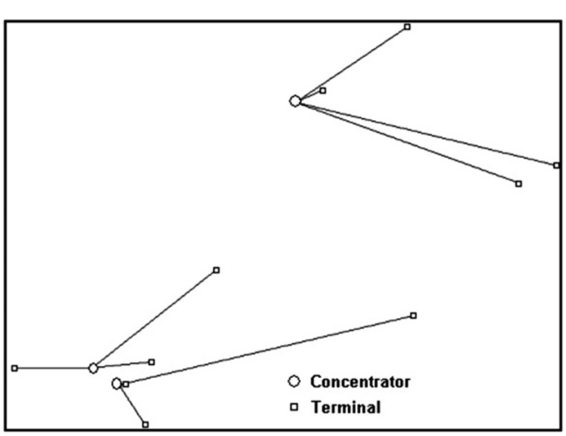
Figure 1 illustrates a possible assignment to a problem with 10 terminals ($N=10$) and 3 concentrators ($M=3$). The figure shows the coordinates of terminals and concentrators based on a 100×100 Euclidean grid and also their capacities.

3. Previous work

Many problems in combinatorial optimisation are NP-Hard. Meta-heuristics are a class of approximate methods designed to optimise complex combinatorial optimisation problems where classical heuristics have failed to be efficient. The existing, successful methods in approximate optimisation fall into two classes: Local Search (LS) and population-based search. Meta-heuristics provide general frameworks for creating new hybrids by means of combining different concepts derived from: classical heuristics, artificial intelligence, biological evolution, neural systems, SI and statistical mechanics. These approaches include Simulated Annealing (SA), Tabu Search (TS), Greedy Randomised Adaptive Search Procedure (GRASP), EAs, SI algorithms, their hybrids and others.

Iterated Local Search (ILS) is a simple and powerful stochastic LS method that creates a sequence of solutions generated by an embedded heuristic (Lourenço et al., 2003). ILS is simple, easy to implement, robust, and highly effective. The essential idea of ILS lies in focusing the search on a smaller subspace, defined by locally optimal solutions for a given optimisation engine.

GRASP is a meta-heuristic belonging to the class of LS techniques (Feo and Resende, 1989). It typically consists of iterations made up from successive constructions of a greedy randomised solution and subsequent iterative improvements of it through a LS.



Terminals

	X	Y	Capacity
1	54	28	5
2	28	75	4
3	84	44	4
4	67	17	2
5	90	41	3
6	68	67	1
7	24	79	3
8	38	59	4
9	27	86	5
10	7	76	4

Concentrators

	X	Y	Capacity
1	19	76	12
2	50	30	14
3	23	79	13

TA Solution

2	1	2	2	2	3	3	1	3	1	Concentrators
1	2	3	4	5	6	7	8	9	10	Terminals

Fig. 1. TAP — example.