Neurocomputing 148 (2015) 17–22

Contents lists available at ScienceDirect

Neurocomputing

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

A genetic algorithm for the maximum edge-disjoint paths problem

Chia-Chun Hsu [a,b,*], Hsun-Jung Cho [a]

[a] Department of Transportation and Logistics Management, National Chiao Tung University, Hsinchu 300, Taiwan, ROC
[b] Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA

ARTICLE INFO

Article history:
Received 7 April 2012
Received in revised form
24 September 2012
Accepted 8 October 2012
Available online 7 August 2014

Keywords:
Edge-disjoint paths
MEDP
Genetic algorithm
Ant colony optimization
Disjoint paths

ABSTRACT

Optimization problems concerning edge-disjoint paths have attracted considerable attention for decades. These problems have a lot of applications in the areas of call admission control, real-time communication, VLSI (Very-large-scale integration) layout and reconfiguration, packing, etc. The maximum edge-disjoint paths problem (MEDP) seems to lie in the heart of these problems. Given an undirected graph G and a set of I connection requests, each request consists of a pair of nodes, MEDP is an NP-hard problem which determines the maximum number of accepted requests that can be routed by mutually edge-disjoint (s_i,t_i) paths. We propose a genetic algorithm (GA) to solve the problem. In comparison to the multi-start simple greedy algorithm (MSGA) and the ant colony optimization method (ACO), the proposed GA method performs better in most of the instances in terms of solution quality and time.

1. Introduction

Assigning paths to connection requests is one of the basic operations in the modern communication networks. Each connection request is a pair of physically separated nodes that require a path for information transmission. Given such a set of connection requests, due to the capacity constraints, one wants to assign paths to requests in a way that no two paths share an edge in common. These paths are called the edge disjoint paths (EDPs). The maximum edge-disjoint paths problem (MEDP) maximizes the number of requests that are simultaneously realizable as EDPs. The problem turns out to be one of the classical combinatorial problems in the NP-complete category.

MEDP can be formally stated as follows. Let $G=(V,E)$ be an undirected and connected graph, where $|V|=n$ and $|E|=m$. A sequence of edges $\pi=\{e_1,e_2,\ldots,e_l\}$, where $e_i \in E$, $i=1,\ldots,l$, is called a path of length $l=|\pi|$. Two paths are edge-disjoint if they do not have any edge in common, otherwise we say they interfere. Let $T=\{(s_i,t_i)|i=1,\ldots,\mathrm{I}$ and $s_i \neq t_i \in V\}$ be a list of connection requests. Each request (s_i,t_i) in G is a pair of vertices that asks for a path connecting s_i and t_i. A feasible solution of MEDP is given by a subset $R \subseteq T$, such that each request in R is assigned a path. The assigned paths are pairwise edge-disjoint and denoted by S. The requests in R are called realizable (or accepted) requests. The goal of the maximum edge-disjoint paths problem is to maximize the cardinality of R.

Early works on the edge-disjoint paths problem have focused on the decision problem, which is one of the classical *NP*-complete problems [1]. The investigation of MEDP started in the 1990s and is still ongoing [2–5]. Since that MEDP is an *NP*-hard problem on general graphs, many studies devoted to obtaining good approximation algorithms and exploring more tractable classes of graphs [6]. In the real world, the MEDP has a multitude of applications in the areas of call admission control [7], real-time communication, VLSI layout [2], packing [5], etc. In addition, the routing and wavelength assignment (RWA) problem [8,9] and unsplittable flow problem (UFP) [3,5,6,10] are direct extensions of MEDP. Therefore, the importance of MEDP is significant.

For specific classes of graphs, MEDP can be optimally solved in polynomial time. Such class includes chains, undirected trees, bipartited stars, and undirected (or bidirected) rings, etc. We refer to [6] for more details. For arbitrary graphs, approximation algorithms including simple greedy algorithm, bounded greedy algorithm and shortest-path-first greedy algorithm, were proposed and their approximation ratios were provided [3,5]. However, to the best of our knowledge, there is a lack of efficient algorithms for tackling the MEDP problem on arbitrary graphs. Greedy algorithms and the ant colony optimization (ACO) approach are the only existing methods. In this paper, we present a novel genetic algorithm (GA) for solving the MEDP problem on arbitrary graphs. The GA is a powerful stochastic search method which has been successfully applied to tackle optimization

* Corresponding author. Tel.: +886 921462912.
E-mail addresses: chsu4@ncsu.edu (C.-C. Hsu), hjcho@cc.nctu.edu.tw (H.-J. Cho).

http://dx.doi.org/10.1016/j.neucom.2012.10.046
0925-2312/

problems in engineering and science. It has been broadly used on network optimization problems [11] as well.

The paper is organized as follows. In Section 2, we review one conventional greedy algorithm and an ACO approach for solving MEDP problems. In Section 3 we present our algorithm. The benchmark instances used to test the performance of the GA approach are introduced in Section 4. The computational results and some observations are also provided. In Section 5, we make a conclusion and point out possible directions for future research.

2. Greedy approach and ACO approach for MEDP

A greedy algorithm is a straightforward constructive algorithm that starts from an empty solution and establishes the solution step by step with a greedy strategy. It can often provide a solution in reasonable computational time. The simple greedy algorithm (SGA) for MEDP shown in Algorithm 1 was proposed in [3]. It starts with an empty set S and R, then iteratively assigns a shortest path to the connection request according to the given order. Each time a path is established, all the edges along that path are removed from the graph. The algorithm halts after I iterations.

Algorithm 1 Simple Greedy Algorithm (SGA)

Input: $G=(V,E)$ and $T=\{(s_i,t_i)|i=1,\ldots,I\}$
$S\leftarrow\varnothing$, $R\leftarrow\varnothing$;
for $i=1$ to I
  if $\exists$path from s_i to t_i in G *then*
    $\pi_i\leftarrow a$ shortest path from s_i to t_i in G;
    $S\leftarrow S\cup\pi_i$;
    $E\leftarrow E\backslash\{e|e\in\pi_i\}$;
    $R\leftarrow R\cup\{(s_i,t_i)\}$;
  end if
end for
Output: Realizable requests R and edge-disjoint paths S

The main drawback of SGA is that the solution quality highly depends on the order of the given connection requests. An intuitive way to improve SGA is applying the multi-start simple greedy (MSGA) algorithm [12]. MSGA runs SGA for several times, each time the order of connection requests is randomly permuted. The algorithm then outputs the best solution among all runs.

Other two improved greedy algorithms including the bounded-length greedy algorithm and the shortest-path-first greedy algorithm, were proposed by Kleinberg [3] and Kolliopoulos [5], respectively. The bounded-length greedy algorithm takes an extra parameter D to denote the threshold of route length. A request is accepted only if it can be routed on a path of length at most D. The shortest-path-first greedy algorithm is another modification of SGA. In each iteration, the shortest path of each of the remaining requests is obtained. Then the request that has the path with the shortest length among all the paths is accepted and removed from the request list. This process repeats until no path can be found for any of the remaining requests. Both greedy algorithms have proven to have better performance than SGA [3,5].

However, due to the deterministic decisions that greedy algorithms take during the solution construction procedure, it is sometimes impossible to find the optimal solution. Fig. 1 shows an example (illustrated in [12]) which consists of a network with the connection request $T=\{(v_1,v_7),(v_8,v_{14}),(v_{15},v_{21})\}$. We can observe that the optimal solution should contain all three requests and the paths are shown in boldface. However, since all the greedy algorithms are based on the shortest path algorithm, none of them

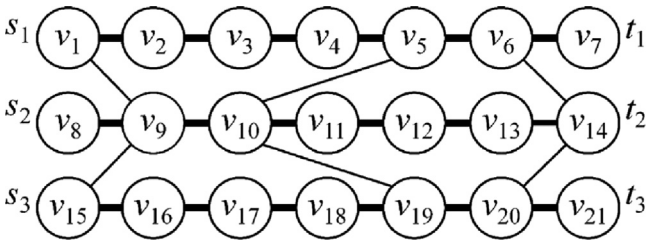

Fig. 1. Instance of the MEDP with $T=\{(v_1,v_7),(v_8,v_{14}),(v_{15},v_{21})\}$. None of the greedy algorithms can find the optimal solution shown in bold font.

can obtain the optimal solution. No matter which connection request we start with, its shortest path always includes one edge that makes it impossible to establish the optimal paths for the other requests. We can observe that the greedy algorithm cannot find a solution of size greater than two.

The application of the ant colony optimization (ACO) to solving the MEDP problem was proposed in [12], which is the only known metaheuristic method so far. The ACO decomposes MEDP into I subproblems $P_i=(G,T_i)$, with $i\in\{1,\ldots,I\}$. Each subproblem itself is trying to find a path for the request $T_i=(s_i,t_i)\in T$ by an ant. During the process of path construction, an ant iteratively moves from one node to another along an edge, the choice of destination can be made either deterministically or stochastically. A constructed ant solution contains I paths which are not necessarily edge-disjoint. An edge-disjoint solution S can be obtained by iteratively remove the path that has the most edges in common with other paths, until the remaining paths are mutually edge-disjoint. A pheromone model τ^i is applied for each subproblem P_i. Each pheromone model τ^i consists of a pheromone value τ^i_e for each edge $e\in E$. We refer readers to [12] for the details of the path construction and pheromone updating procedures.

3. Genetic algorithm for MEDP

A genetic algorithm (GA) mimics the natural evolution processes to gradually improve the solution. Several components are required for solving MEDP by GA: (1) a genetic representation of the solution domain, (2) a way to create an initial population, (3) genetic operators to create new offspring at each generation, and (4) a fitness function for evaluating a solution. The details of these basic components are introduced in this section.

3.1. Decoding and encoding procedure

Since the MEDP problem considers the routes between I terminal pairs, we let an individual (or a solution) contain information of I paths. Each path is encoded by an n-bit string of real values in the interval [0,1]. Each of these values denotes a node's "priority" of being selected into a path during the decoding process. Thereby, an nI-bit string is required to encode an individual. Let $\boldsymbol{u}$ be a $1\times n$ vector of priority values denoting a path from s to t. A $1\times n$ binary vector l is the label of the nodes. Note l is set to $\widehat{0}$ at the beginning. Decoding of $\boldsymbol{u}$ is a path construction procedure. The procedure starts at the source node s, which is also set as the current node w. The candidate nodes for next move, denoted by C, are the unlabeled nodes adjacent to w. If C is not empty, the current node w will move to the node which has the greatest priority value. The label of w is then set to 1 and the path $\boldsymbol{\pi}$ grows. Otherwise, the procedure backtracks. This construction procedure stops when $w=t$.

Algorithm 2 Decoding procedure
Input: $\boldsymbol{u}$ is a priority vector with n elements,
  source s,