Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 55 (2016) 25–28

www.elsevier.com/locate/endm

A preliminary analysis of the Distance Based Critical Node Problem

Roberto Aringhieri, Andrea Grosso, Pierre Hosteins [1]

Dipartimento di Informatica
Università degli Studi di Torino
Turin, Italy

Rosario Scatamacchia [2]

Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy

Abstract

We discuss how to develop efficient heuristics for the distance based critical node problem, that is the problem of deleting a subset of nodes from a graph G in such a way that the distance between each pair of nodes is as large as possible.

Keywords: Critical Node Problem, Graph Fragmentation, Shortest Paths.

1 Introduction

The Critical Node Problem (CNP) has been defined as a type of Interdiction Network Problem which aims at maximally fragmenting an undirected and

[1] Email: roberto.aringhieri@unito.it, grosso@di.unito.it, hosteins@di.unito.it
[2] Email: rosario.scatamacchia@polito.it

http://dx.doi.org/10.1016/j.endm.2016.10.007

unweighted graph $G = (V, E)$ by deleting a subset of its nodes $S \subset V$ ($|S| = K$) according to a specific connectivity measure. This particular problem has raised a certain interest in the recent literature due its potential applicability to a vast number of real situations (see, e.g., [4]). Currently, the state of the art algorithms for solving the CNP are those presented in [1,2,3].

In the classic CNP, the connectivity is related to a pair-wise connectivity concept, that is either a path exists between a pair of nodes, or it does not. In [8], the authors introduces a more refined connectivity concept based on the shortest distance between each pair of nodes: the more distant the nodes, the lower their connectivity value. Therefore, the DB-CNP consists in minimizing the following objective function:

$$F(S) = \sum_{i,j \in V \setminus S \,:\, i \neq j} \frac{1}{d_{spt}(i,j)} \tag{1}$$

where d_{spt} is the value of the shortest path between the node i and the node j belonging to the weighted graph G.

Constructive and Local Search based heuristics usually build an incumbent solution step-by-step, that is, for instance, adding or deleting elements, or swapping a pair of elements respectively belonging and not belonging to a starting solution. As for the classic CNP, the development of efficient heuristic algorithms for the DB-CNP suffers from the non trivial evaluation of the incumbent new solution since we need to update the shortest path between each pair of nodes. In this paper we discuss how to develop efficient heuristics for the DB-CNP.

2 Shortest paths re-computation

The operations traditionally used to obtain an incumbent solution of the CNP consist in adding a node to S (i.e., deleting it from the graph), removing a node from S or swapping a node from S with a node from $V \setminus S$. As moving nodes from or to S can affect the length of shortest paths (SP), we are required to recompute all the SP values in the graph, which is known to have a computational cost of $\mathcal{O}(|V||E| + |V|^2 \log |V|)$ [6]. As such a complexity is usually prohibitive when thousands of incumbent solutions should be evaluated, we need to implement more efficient evaluations of the SP modifications.

It has been noted in computational works regarding all-SP re-computation that usually, if a very small number of edges' weights are modified, the time necessary to recompute only the shortest paths that are affected is actually