A preliminary analysis of the Distance Based Critical Node Problem

R. Aringhieri, A. Grosso, P. Hosteins and R. Scatamacchia.

A preliminary analysis of the Distance Based Critical Node Problem

Electronic Notes in Discrete Mathematics , 55C: 25–28, 2016. Available online 17 November 2016.

DOI: 10.1016/j.endm.2016.10.007

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

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

much less than the theoretical worst case complexity. Since for the CNP we only modify the edges belonging to the backward and the forward start of one node at a time, such empirical results are encouraging for implementing efficient heuristics.

Moreover, some particular cases of interest to us can be demonstrated to require a lower worst case complexity than the general all-SP re-computation. For example, reintroducing a node $u \in S$ inside the graph amounts to consider that each SP can now go through u if it is profitable enough. Using the SP properties, we can show that computing the SP starting and ending at u can be done in $\mathcal{O}(D(G)(|V| - |S|))$ where $D(G)$ is the largest number of edges incident on any node in G. Then using those new SP lengths we can update all shortest paths in a maximum number of operations equal to $\mathcal{O}(|V|^2)$, which is inferior to the general case of edge weights modification [6].

Some dominance rules should also be devised, the simplest example being a node $v \in V \setminus S$ which does not belong to any shortest path in graph $G[V \setminus S]$: evidently such a node can never be an appropriate candidate for deletion since such a move would not lower the objective function. Similarly, the impact of removing a node belonging to a certain connected component would not change if our moves in the solution space only modifies other connected components.

3 Extension to directed graphs and weighted pair-wise connectivity

Since the SP definition is not limited to undirected graphs, the DB-CNP can be also applied to directed graphs, which opens the perspective of applying the critical node analysis to such situations that can be modelled by directed graphs only, contrary to the versions of the CNP previously considered in the literature [4].

We also note that the CNP based on weighted pair-wise connectivity is much more difficult to tackle with the existing heuristic algorithms since they tend to rely on the fact that not weighted pair-wise connectivity can be computed solely using the connected components cardinality, a fact which is no longer true when weights are introduced between pairs of nodes. However, the heuristic framework developed for a DB-CNP, which tracks the SP values, allows us to evaluate solution moves for weighted pair-wise connectivity as a non infinite length means that the nodes are connected. Thus we see that algorithmic efforts in order to solve the DB-CNP can be beneficial for other formulations of the CNP as well.

4 Betweenness centrality

Betweenness centrality [7] can play also a fundamental role to devise efficient heuristics. Centrality would evaluate how important is a node for the connection of every pair of nodes. Betweenness centrality of the node j is the number of shortest paths from all vertices to all others that pass through the node j, and it can be computed using the Brandes' algorithm [5]. The basic idea is therefore to rank the nodes with respect to their betweenness value, and to consider first those having highest value in our heuristics. Note that heuristics for the classic CNP benefit of using such a rank as reported in [1,3].

References

[1] Addis, B., R. Aringhieri, A. Grosso and P. Hosteins, *Hybrid Constructive Heuristics for the Critical Node Problem*, Annals of Operations Research **238** (2016), pp. 637–649.

[2] Aringhieri, R., A. Grosso and P. Hosteins, *A genetic algorithm for a class of Critical Node Problems*, Electronic Notes in Discrete Mathematics **52** (2016), pp. 359–366.

[3] Aringhieri, R., A. Grosso, P. Hosteins and R. Scatamacchia, *Local Search Metaheuristics for the Critical Node Problem*, Networks **67** (2016), pp. 209–221.

[4] Arulselvan, A., C. W. Commander, L. Elefteriadou and P. M. Pardalos, *Detecting critical nodes in sparse graphs*, Computers & Operations Research **36** (2009), pp. 2193–2200.

[5] Brandes, U., *A faster algorithm for betweenness centrality*, Journal of Mathematical Sociology **25** (2001), pp. 163–177.

[6] Demetrescu, C. and G. F. Italiano, *Experimental analysis of dynamic all pairs shortest path algorithms*, ACM Trans. Algorithms **2** (2006), pp. 578–601.

[7] Freeman, L. C., *A Set of Measures of Centrality Based on Betweenness*, Sociometry **40** (1977), pp. 35–41.

[8] Veremyev, A., O. A. Prokopyev and E. L. Pasiliao, *Critical nodes for distance-based connectivity and related problems in graphs*, Networks **66** (2015), pp. 170–195.