Engineering Applications of Artificial Intelligence 55 (2016) 128–145

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A general Evolutionary Framework for different classes of Critical Node Problems

Roberto Aringhieri [a], Andrea Grosso [a], Pierre Hosteins [a,*], Rosario Scatamacchia [b]

[a] Università degli Studi di Torino, Dipartimento di Informatica, Corso Svizzera, 185-10149 Torino, Italy
[b] Politecnico di Torino, Dipartimento di Automatica e Informatica, Corso Duca degli Abruzzi, 24-10129 Torino, Italy

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form
26 May 2016
Accepted 18 June 2016

Keywords:
Evolutionary algorithm
Critical Node Problem
Graph fragmentation
Greedy rules
Connectivity measures

ABSTRACT

We design a flexible Evolutionary Framework for solving several classes of the Critical Node Problem (CNP), i.e. the maximal fragmentation of a graph through node deletion, given a measure of connectivity. The algorithm uses greedy rules in order to lead the search towards good quality solutions during reproduction and mutation phases. Such rules, which are only partially reported in the literature, are generalised and adapted to the six different formulations of the CNP considered along the paper. The link between solutions of different CNP formulations is investigated, both quantitatively and qualitatively. Furthermore, we provide a comparison with best known results when those are available in literature that confirms the good overall quality of our solutions.

1. Introduction

The Critical Node Problem (CNP) is a class of Interdiction Network Problems (Wollmer, 1964; Wood, 1993) that focuses on maximally fragmenting a graph $G(V, E)$ by deleting a set $S \subset V$ of its nodes (and all incident edges on such nodes). This problem is of interest in a wide range of possible situations, including the identification of key players in a social network (Borgatti, 2006), transportation networks' vulnerability (Jenelius et al., 2006), power grid construction and vulnerability (Salmerón et al., 2004), homeland security (Brown et al., 2006), telecommunications (Alevras et al., 1997) or epidemic control (Zhou et al., 2006) and immunisation strategies (Arulselvan et al., 2009; Cohen et al., 2003; Ventresca, 2012). A possible application to computational biology, through the example of protein–protein interaction networks, has been suggested in Boginski et al. (2009).

Each domain of application usually defines a specific version of the problem through the use of a particular connectivity measure. Moreover, solving real graphs with up to thousands of nodes often calls for the use of an efficient heuristic algorithm. The contribution of the approach advocated here is twofold: on one hand, it provides a global and flexible framework that allows us to deal with different fragmentation measures. On the other hand, it can find good quality solutions with limited costs in terms of algorithmic implementation and computational effort. To the best of the authors' knowledge, this is the first attempt to develop a general tool for tackling different classes of the CNP.

We will represent a solution by the set of its deleted nodes S. The degree of fragmentation of the induced graph $G[V\backslash S]$ needs to be measured by a given connectivity metric. We will consider only undirected graphs and we denote the set of maximal connected components as $\mathcal{H}$ and the cardinality of the said components as $|h|$ for $h \in \mathcal{H}$.

Many connectivity measures can be devised according to the type of application desired. We will concentrate on the measures that take into account the number of remaining connected components and their cardinality after the deletion of set S, which is generally enough to determine which nodes are still able to interact in the remaining network. These measures are defined as (i) pair-wise connectivity, i.e. the number of pair of nodes connected by a path inside the graph, (ii) the size of the largest connected component and (iii) the number of connected components. The value of these three measures for a solution set S will be expressed, respectively, through the following mathematical functions:

$$f(S) = |\{i, j \in V\backslash S: i \text{ and } j \text{ connected by a path in } G[V\backslash S]\}|, \quad (1)$$

$$C(S) = \max\{|h|, h \in \mathcal{H}(G[V\backslash S])\}, \quad (2)$$

$$H(S) = |\mathcal{H}(G[V\backslash S])|. \quad (3)$$

Pair-wise connectivity $f(S)$ can alternatively be expressed in

* Corresponding author.
E-mail addresses: roberto.aringhieri@unito.it (R. Aringhieri), andrea.grosso@unito.it (A. Grosso), hosteins@di.unito.it (P. Hosteins), rosario.scatamacchia@polito.it (R. Scatamacchia).

http://dx.doi.org/10.1016/j.engappai.2016.06.010
0952-1976/

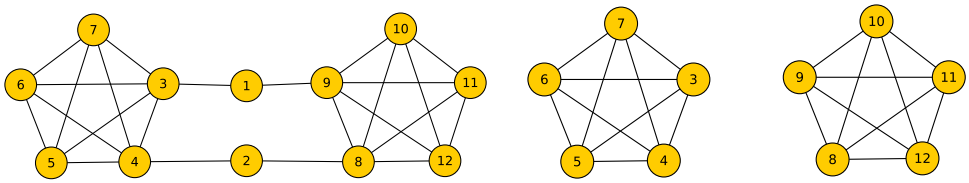

Fig. 1. Example of a small graph (on the left) fragmented into two connected components (on the right) after the removal of nodes 1 and 2.

terms of the cardinality of the maximal connected components: $f(S) = \sum_{h \in \mathcal{H}} \frac{|h|(|h|-1)}{2}$. Even though these measures are all different and can lead to very different optimal solutions, as explicitly demonstrated in Shen and Smith (2012), they are not generally unrelated. For example the ideal situation for minimising the pair-wise connectivity is to obtain the largest number of connected components $H(S)$ with the smallest possible variance in their cardinality. This implies a minimisation of the size of the largest component. In practice, this means that disrupting pair-wise connectivity $f(S)$ is a tradeoff between minimising the cardinality of the largest component $C(S)$ and maximising the number of connected components $H(S)$. As the nodes are removed or disabled, we do not count them as single components. An example of the fragmentation of a small graph is provided in Fig. 1: after the removal of two nodes (numbers 1 and 2), the graph is split into two connected components of five nodes each. This solution corresponds to the optimal solution when trying to either minimise $f(S)$ and $C(S)$ or maximise $H(S)$ by removing at most two nodes from the graph, with corresponding values: $f(\{1,2\}) = 20$, $C(\{1,2\}) = 5$ and $H(\{1,2\}) = 2$.

Given a connectivity measure, a CNP solution is defined by the set of deleted nodes and the value of the connectivity metric for the resulting graph. Depending on the problem at hand, the selection of the nodes can be performed using two complementary approaches:

- the budget constrained formulation: minimise/maximise the connectivity under a budget limitation over $S(|S| \leq K)$;
- the connectivity constrained formulation: minimise the number of nodes deleted $(|S|)$ in order to meet a threshold connectivity value.

For the sake of clarity, we will refer to the problems with the different connectivity measures $f(S)$, $C(S)$ and $H(S)$ as CNP1, CNP2 and CNP3, respectively. For each problem, we consider the two variants of the CNP that arise taking into account both the budget ("*a*") and connectivity ("*b*") constrained formulations, that is

- CNP1*a*: minimise $f(S)$ (pair-wise connectivity) subject to $|S| \leq K$.
- CNP1*b*: minimise $|S|$ such that $f(S) \leq P$.
- CNP2*a*: minimise $C(S)$ (cardinality of the largest connected component of $G[V \setminus S]$) subject to $|S| \leq K$.
- CNP2*b*: minimise $|S|$ such that $C(S) \leq L$ (L denotes the cardinality parameter in accordance with notations in Boginski et al., 2009; Arulselvan et al., 2011; Veremyev et al., 2014a).
- CNP3*a*: maximise $H(S)$ (number of connected components of $G[V \setminus S]$) subject to $|S| \leq K$.
- CNP3*b*: minimise $|S|$ such that $H(S) \geq N$.

In this paper we will consider the 6 different types of the CNP problem accordingly to the above taxonomy. Handling each of these formulations through the use of a single algorithmic framework is not straightforward. For instance, the VNS algorithm provided in Aringhieri et al. (2016b) for CNP1*a*, which provides good results compared to other heuristics for that problem, is hard to generalise even to the CNP1*b*. One main reason is the fact that finding feasible solutions for "*b*" types of the CNP is potentially very difficult, posing a relevant challenge for implementing the classical shaking procedures in a VNS framework and in general for the exploration of the solution space. Another important difficulty concerns the application of local search approaches. In order to improve the objective value of an instance of CNP1*b*, a local search procedure should involve a swap of a node from $V \setminus S$ with at least two nodes from S, which would increase the complexity of a move by a factor $K/2$ compared to the "*a*" version (more details about local search procedures for the CNP are provided in Section 3.5). Furthermore, the development of efficient neighbourhoods is also challenging, as discussed in Aringhieri et al. (2016b).

We will demonstrate how our Evolutionary Framework (EF) can tackle any of the six problems above by using tailored reproduction and mutation operators capable of repairing the solutions through appropriate greedy rules (preliminary results of such a framework can be found in Aringhieri et al., 2016a). Such rules can effectively guide the search through the solution space, in particular when they are properly combined as pointed out by the previous work of Addis et al. (2016).

Based on the considerations above, the aim of this work is to provide a simple and easy to implement algorithmic framework that can tackle many different versions of the CNP by embedding suitable and efficient greedy rules.

Table 1 reports the main heuristic algorithms in the literature for the different types of the CNP considered. CNP1*a* has gained more attention, while there exists a gap in the literature for the other five versions. We further extend the analysis of the CNP to these versions and propose a set of benchmark results which may constitute an interesting basis for comparison for future algorithms.

The paper is organised as follows. Section 2 introduces the greedy rules adopted as well as some greedy algorithms that will be used for comparison. Section 3 describes a general evolutionary algorithm for the different types of the CNP as defined above, embedding the greedy rules defined in Section 2 within the tailored reproduction and mutation operators. Section 4 discusses the results of the evolutionary algorithm over a set of benchmark instances and investigates the correlation between solutions of the different types of the CNP. Finally Section 5 provides conclusions

Table 1
Heuristic algorithms from the literature for the six types of the CNP considered in this work.

	Type "*a*"	Type "*b*"
CNP1	Greedy algorithms (Arulselvan et al., 2009; Ventresca and Aleman, 2015; Addis et al., 2016) Simulated annealing and PBIL (Ventresca, 2012) VNS and ILS (Aringhieri et al., 2016b)	Approximation algorithm (Dinh et al., 2010)
CNP2	–	Greedy and genetic algorithm (Boginski et al., 2009; Arulselvan et al., 2011)
CNP3	–	–