VNS solutions for the critical node problem

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/155351 since 2015-12-11T17:17:53Z

Published version:

DOI:10.1016/j.endm.2014.11.006

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)
This is an author version of the contribution published on:

R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia.
VNS solutions for the critical node problem.
In The 3rd International Conference on Variable Neighborhood Search (VNS’14), volume 47 of Electronic Notes in Discrete Mathematics, pages 37-44, February 2015
DOI: 10.1016/j.endm.2014.11.006

The definitive version is available at:

VNS solutions for the Critical Node Problem

Roberto Aringhieri, Andrea Grosso, Pierre Hosteins\textsuperscript{1,2}

\textit{Dipartimento di Informatica}
\textit{Università degli Studi di Torino}
\textit{Turin, Italy}

Rosario Scatamacchia\textsuperscript{1,3}

\textit{Dipartimento di Automatica e Informatica}
\textit{Politecnico di Torino}
\textit{Turin, Italy}

\underline{Abstract}

We present a VNS algorithm for the Critical Node Problem, i.e., the maximal fragmentation of a graph through the deletion of $k$ nodes. Two computational efficient neighbourhoods are proposed proving also their equivalence to the straightforward exchange of two nodes. The results of the proposed VNS algorithms outperform those currently available in literature.

\textit{Keywords:} Critical Node Problem, graph fragmentation.

\textsuperscript{1} Work supported by a Google Focused Grant on Mathematical Programming, project “Exact and Heuristic Algorithms for Detecting Critical Nodes in Graphs”.

\textsuperscript{2} Email: roberto.aringhieri@unito.it, grosso@di.unito.it, hosteins@di.unito.it

\textsuperscript{3} Email: rosario.scatamacchia@polito.it
1 Introduction

Given an undirected graph \( G(V, E) \) and an integer \( k \), the Critical Node Problem (CNP) consists in finding a subset of \( k \) nodes \( S \subseteq V \), such that the number of node pairs still connected in the induced subgraph \( G[V \setminus S] \)

\[
f(S) = |\{u, v \in V \setminus S: u \text{ and } v \text{ are connected by a path in } G[V \setminus S]\}|
\]

is as small as possible.

To the authors’ knowledge, the problem can be traced back to the so-called network interdiction problems studied by Wollmer [14] and later Wood [15] where anyway emphasis is placed on arc deletion instead of nodes. The interest about node deletions seems to have spread more recently: Borgatti [3] states the CNP in the context of detecting so-called “key players” in a relational network; Arulselvan et al. [1] and Ventresca [13] emphasize contagion control via vaccination of a limited number of individuals, when the nodes of the graph are potentially infected individuals and the edges represent contacts occurring between them. Similar problems are studied for assessing robustness of communication networks by Dinh et al. [5,11].

The CNP is known to be NP-complete [1], polynomially solvable on trees [4] and other “simple” kinds of graphs [2,10]. For a broad literature review, including problems with different metrics about graph fragmentation, we refer to comprehensive references given in other works, for example [10,9]. From a practical point of view, the CNP on general graphs has been tackled by Arulselvan et al. [1] by proposing a MILP model and a heuristic approach based on a greedy heuristic coupled with a successive local search phase. Sophisticated metaheuristics – namely population-based incremental learning and simulated annealing – are studied and experimentally compared by Ventresca [13].

In this paper, we present a solution framework for the CNP based on the Variable Neighbourhood Search (VNS) methodology [7,8]. The main contribution of this paper concerns the development of two computational efficient neighbourhoods which are also proved to be equivalent to the straightforward neighbourhood exchanging of two nodes \( u \) and \( v \) in such a way that \( u \in S \) and \( v \in V \setminus S \). This allows the development of a VNS solution framework for the CNP which largely improves the best known solutions provided in literature. The paper is organized as follows. Section 2 reports the two efficient neighbourhood explorations. Section 3 depicts the proposed VNS approach for the solution of CNP. Section 4 discusses the computational results of the proposed algorithm on the state of the art benchmark instances. Finally, Section 5 closes the paper.
2 Efficient neighbourhood exploration

A straightforward neighbourhood, say $N_0$, tries to improve a given solution $S$ by exchanging a node $u \in S$ with another node $v \in V \setminus S$ (2-node-exchange). Its main drawback lies in the computational complexity of the move evaluation which requires to perform a graph search whose complexity is $O(|V| + |E|)$. In the worst case, that is when $k = |S| = \frac{|V|}{2}$, the whole neighbourhood exploration is $O(|V|^3)$ because of the number of exchanges to be evaluated is quadratic in the number of nodes in $V$. We propose two more efficient neighbourhoods based on the idea of reducing the number of exchanges to be evaluated to overcome this computational issue.

The former, say $N_1$, considers all nodes in $u \in S$ and evaluates, for each node, the exchange with the node $v \in V \setminus S$ that disconnects the graph as much as possible, i.e., such that $v = \arg \max \{ f((S \cup \{v\}) \setminus \{u\}) - f(S) \}$. The latter, say $N_2$, builds a move exchanging each node $v \in V \setminus S$ with the node $u$ which yields the minimum increase in the objective function, i.e., $u = \arg \min \{ f((S \cup \{v\}) \setminus \{u'\}) - f(S) \}$. Note that, both $N_1$ and $N_2$ break ties at random.

The number of exchanges in $N_1$ and $N_2$ are, respectively, $k$ and $|V| - k$. Therefore, the complexity of a complete neighbourhood exploration is $O(|V|^2)$ instead of $O(|V|^3)$ of 2-node-exchange. This is due to the fact that $u'$ and $u''$ can be computed in $O(|V| + |E|)$ through a modified depth-first search [12].

Finally, we would like to remark that $N_0$, $N_1$ and $N_2$ yield the same best move $(u^*, v^*)$, unless breaking ties lead to the choice of different pairs of nodes, as proved below.

**Theorem 2.1** Neighbourhood exploration methods $N_0$, $N_1$ and $N_2$ all yield the same result $(u^*, v^*)$ when there is only one best move (no ties).

**Proof.** Let us suppose there exists one and only one best possible move $(u^*, v^*)$. By construction $N_0$ is sure to identify it. Let us now consider $N_1$: if we select a node $u \neq u^*$ from set $S$, any resulting move $(u, v)$ the algorithm finds is by construction inferior to $(u^*, v^*)$. If in fact we choose $u = u^*$, let us suppose the algorithm extracts a pair $(u^*, v)$ where $v \neq v^*$: since the algorithm evaluates the impact of all possible nodes $v \in (V \setminus S) \cup \{u\}$, it means we have found a pair $(u^*, v)$ with higher impact than $(u^*, v^*)$, which is impossible by construction.

Thus $N_1$ extracts the best possible pair $(u^*, v^*)$. Given that for a given $v \in V \setminus S$, the rule based on $N_2$ evaluates the impact of all nodes $u \in S \cup \{v\}$, a similar reasoning proves that $N_2$ will also select $(u^*, v^*)$. \qed
3 A Variable Neighbourhood Search for CNP

In this section we present a general VNS solution framework to deal with CNP. In the following we refer to the basic VNS scheme discussed in [8] (cf., algorithm 7). From a notational point of view, we use $h$ instead of $k$ to denote the $k$th neighbourhood of a solution $x$, and $S$ to denote a solution.

The main ingredients of our algorithm are: the procedure to compute an initial solution, the improvement and the shake procedures. The pseudocode of our algorithm is depicted in Algorithm 1.

The initial solution can be any set $S$ of deleted $k$ nodes. In our current implementation, we select such nodes by applying the greedy algorithm proposed in [1]. The improvement procedure is a Local Search (line 3) based on the first improvement exploration of the neighbourhood $N_1$ or, alternatively, $N_2$. Let $\phi_u$ be the occurrence or frequency in which $u$ belongs to a solution $S$. The value $\phi_u$ is updated ($\phi_u \leftarrow \phi_u + 1$) in two cases: if $u$ is added to the solution $S$ during the neighbourhood exploration (line 3), and if $u$ belongs to a solution improving the current best solution (line 5). The shaking procedure replaces the $h$ most frequent nodes in $S$ with the $h$ least frequent nodes in $V \setminus S$.

Algorithm 1 A Variable Neighbourhood Search for CNP

**CNP-VNS** ($S, t_{max}, h_{max}$)

repeat
  $h \leftarrow 2$;
  repeat
    $S' \leftarrow \text{Shake}(S, h)$;
    $S'' \leftarrow \text{FirstImprovement}(S', N_1)$;
    if ($f(S'') < f(S)$) then
      $S \leftarrow S''$;
      $h \leftarrow 2$;
    else
      $h \leftarrow h + 1$;
    end;
  until $h = h_{max}$;
  $t \leftarrow \text{cpuTime}()$;
until $t \geq t_{max}$;
return $S$

In the following, we consider 4 different versions derived from Algorithm 1 varying the neighbourhood in the Local Search ($N_1$ or $N_2$) and increasing $h$
(\(h = 2, \ldots, h_{\text{max}}\)) or decreasing \(h \ (h = h_{\text{max}}, \ldots, 2)\). We denote them as VNS-I-\(N_1\), VNS-I-\(N_2\), VNS-D-\(N_1\), VNS-D-\(N_2\) where I and D stand for increasing and decreasing \(h\), respectively.

4 Computational analysis

In this section we report the computational analysis of the 4 versions of Algorithm 1, with two different update mechanisms of the frequencies of the selected nodes. They were programmed in standard C++ and compiled with gcc 4.1.2. All tests were performed on an HP ProLiant DL585 G6 server with two 2.1 GHz AMD Opteron 8425HE processors and 16 GB of RAM. We use the graphs presented in [13] as benchmark instances and compare our results with the best known results (coming from [6]), provided in our tables in the column “BK”. Each graph has a specific topology based on Erdos-Renyi, Barbasi-Albert, Watts-Strogatz and Forest Fire models (see [13] for more details). In the column “graph” we indicate the type of graph by two letters (e.g. BA stands for Barabasi-Albert, etc...), followed by its number of nodes. New best known results are displayed in bold font.

The results are clearly in favour of the VNS algorithms, with a few very remarkable improvements. The gap between the VNS and the best known result is often around 50% and sometimes much higher, except for BA graphs where it can be only a few percent lower.

In Table 2, we report the results of our algorithms when adopting a different policy – less pervasive – for the updating of frequencies: actually, we update \(\phi_u\) only when \(u\) belongs to a solution improving the current best solution (line 5 of Algorithm 1). Thus, while we previously updated the frequency of a node each time it was found in a solution, we now only update it when it is found in a local optimum. Comparing the results with those in Table 1, it is evident that there is not a clear dominance of the updated method, in the sense that the conclusion will differ according to the family of graphs considered. We also observe that the number of best known solutions are almost the same, that is 22 in Table 1 and 24 Table 2.

Running times can be quite long when the graphs reach 1000 nodes so we choose to terminate the algorithm after 10000 seconds. Even though this seems substantial, they are comparable or inferior to meta-heuristics from [13], however the newer algorithms in [6] usually run in a matter of seconds as greedy algorithms usually do. Note that VNS based on \(N_1\) are faster (in terms of running time) than those based on \(N_2\) when \(k < \frac{|V|}{2}\). This is true for all our test instances.
A version with best improvement instead of first improvement gives similarly good results (although it finds a lower number of best known results), however the running times are higher for almost all instances: on average from 45% to 244% depending on the algorithm. Table 3 summarizes the best known solution computed by the 4 versions of the algorithm both in the case of first and best improvements (results from the best improvement versions are displayed in italic).

### 5 Conclusions

We devised a solution framework based on VNS methodology for the Critical Node Problem. Our solution framework exploits the two efficient neighbourhoods discussed in Section 2. The results proposed in Table 1 and Table 2 largely outperform those reported in literature, especially for harder benchmark instances. Running times can be high but are justified by the improve-

<table>
<thead>
<tr>
<th>graph</th>
<th>$K$</th>
<th>BK</th>
<th>VNS-D-$N_1$</th>
<th>VNS-D-$N_2$</th>
<th>VNS-I-$N_1$</th>
<th>VNS-I-$N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA500</td>
<td>50</td>
<td>203</td>
<td>195</td>
<td>195</td>
<td>195</td>
<td>195</td>
</tr>
<tr>
<td>BA1000</td>
<td>75</td>
<td>580</td>
<td>559</td>
<td>559</td>
<td>559</td>
<td>559</td>
</tr>
<tr>
<td>BA2500</td>
<td>100</td>
<td>4254</td>
<td>3722</td>
<td>3704</td>
<td>3722</td>
<td>3704</td>
</tr>
<tr>
<td>BA5000</td>
<td>150</td>
<td>11886</td>
<td>10196</td>
<td>10218</td>
<td>10196</td>
<td>10196</td>
</tr>
<tr>
<td>ER235</td>
<td>50</td>
<td>1141</td>
<td>306</td>
<td>306</td>
<td>301</td>
<td>298</td>
</tr>
<tr>
<td>ER466</td>
<td>80</td>
<td>19952</td>
<td>1562</td>
<td>1611</td>
<td>1561</td>
<td>1725</td>
</tr>
<tr>
<td>ER941</td>
<td>140</td>
<td>114166</td>
<td>5470</td>
<td>6106</td>
<td>8106</td>
<td>5198</td>
</tr>
<tr>
<td>ER2344</td>
<td>200</td>
<td>1666656</td>
<td>1112994</td>
<td>1091185</td>
<td>1118785</td>
<td>1094239</td>
</tr>
<tr>
<td>WS250</td>
<td>70</td>
<td>13786</td>
<td>7175</td>
<td>11196</td>
<td>10237</td>
<td>12457</td>
</tr>
<tr>
<td>WS500</td>
<td>125</td>
<td>53779</td>
<td>2148</td>
<td>2209</td>
<td>2230</td>
<td>2209</td>
</tr>
<tr>
<td>WS1000</td>
<td>200</td>
<td>308596</td>
<td>198494</td>
<td>139653</td>
<td>268500</td>
<td>179531</td>
</tr>
<tr>
<td>WS1500</td>
<td>265</td>
<td>653015</td>
<td>16210</td>
<td>16549</td>
<td>14623</td>
<td>14619</td>
</tr>
<tr>
<td>FF250</td>
<td>50</td>
<td>302</td>
<td>194</td>
<td>198</td>
<td>198</td>
<td>198</td>
</tr>
<tr>
<td>FF500</td>
<td>110</td>
<td>344</td>
<td>257</td>
<td>258</td>
<td>257</td>
<td>257</td>
</tr>
<tr>
<td>FF1000</td>
<td>150</td>
<td>1880</td>
<td>1270</td>
<td>1274</td>
<td>1263</td>
<td>1265</td>
</tr>
<tr>
<td>FF2000</td>
<td>200</td>
<td>7432</td>
<td>4578</td>
<td>4584</td>
<td>4583</td>
<td>4549</td>
</tr>
</tbody>
</table>

Table 1

Results of the 4 algorithms (BK stands for Best Known result from previous works, while $K$ is the number of nodes deleted from the graph).
Table 2
Results of the 4 algorithms (update $\phi_u$ only on line 5 of Algorithm 1).

<table>
<thead>
<tr>
<th>graph</th>
<th>$K$</th>
<th>BK</th>
<th>VNS-D-$N_1$</th>
<th>VNS-D-$N_2$</th>
<th>VNS-I-$N_1$</th>
<th>VNS-I-$N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA500</td>
<td>50</td>
<td>203</td>
<td>195</td>
<td>195</td>
<td>195</td>
<td>195</td>
</tr>
<tr>
<td>BA1000</td>
<td>75</td>
<td>580</td>
<td>559</td>
<td>559</td>
<td>559</td>
<td>559</td>
</tr>
<tr>
<td>BA2500</td>
<td>100</td>
<td>4254</td>
<td>3722</td>
<td>3704</td>
<td>3722</td>
<td>3704</td>
</tr>
<tr>
<td>BA5000</td>
<td>150</td>
<td>11886</td>
<td>10196</td>
<td>10218</td>
<td>10196</td>
<td>10218</td>
</tr>
<tr>
<td>ER235</td>
<td>50</td>
<td>1141</td>
<td>303</td>
<td>335</td>
<td>301</td>
<td>302</td>
</tr>
<tr>
<td>ER466</td>
<td>80</td>
<td>19952</td>
<td>1542</td>
<td>1727</td>
<td>1567</td>
<td>1751</td>
</tr>
<tr>
<td>ER941</td>
<td>140</td>
<td>141166</td>
<td>5503</td>
<td>6289</td>
<td>5658</td>
<td>5628</td>
</tr>
<tr>
<td>ER2344</td>
<td>200</td>
<td>1606656</td>
<td>1067397</td>
<td>1097573</td>
<td>1052406</td>
<td>1034333</td>
</tr>
<tr>
<td>WS250</td>
<td>70</td>
<td>13786</td>
<td>8833</td>
<td>6610</td>
<td>10413</td>
<td>7186</td>
</tr>
<tr>
<td>WS500</td>
<td>125</td>
<td>53779</td>
<td>2170</td>
<td>2199</td>
<td>2152</td>
<td>2213</td>
</tr>
<tr>
<td>WS1000</td>
<td>200</td>
<td>308596</td>
<td>200225</td>
<td>256239</td>
<td>255061</td>
<td>154813</td>
</tr>
<tr>
<td>WS1500</td>
<td>265</td>
<td>653015</td>
<td>17198</td>
<td>26225</td>
<td>14719</td>
<td>15692</td>
</tr>
<tr>
<td>FF250</td>
<td>50</td>
<td>302</td>
<td>194</td>
<td>199</td>
<td>194</td>
<td>199</td>
</tr>
<tr>
<td>FF500</td>
<td>110</td>
<td>344</td>
<td>257</td>
<td>258</td>
<td>257</td>
<td>258</td>
</tr>
<tr>
<td>FF1000</td>
<td>150</td>
<td>1880</td>
<td>1270</td>
<td>1274</td>
<td>1270</td>
<td>1273</td>
</tr>
<tr>
<td>FF2000</td>
<td>200</td>
<td>7432</td>
<td>4576</td>
<td>4584</td>
<td>4577</td>
<td>4550</td>
</tr>
</tbody>
</table>

Table 3
New Best Known

Ongoing work is devoted to devise auxiliary data structure or different neighbourhoods, which could improve the average case complexity, possibly trading a loss in the quality of the solution for a reduction of the computational time.

References


