Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 52 (2016) 359–366 www.elsevier.com/locate/endm

A Genetic Algorithm for a class of Critical Node Problems

Roberto Aringhieri, Andrea Grosso, Pierre Hosteins [1,2]

Dipartimento di Informatica
Università degli Studi di Torino
Turin, Italy

Abstract

In this paper, we deal with two different variants of the Critical Node Problem, designing a flexible genetic algorithm for tackling them both. The results are compared with the best known results available in the literature.

Keywords: Critical Node Problem, Cardinality-Constrained, graph fragmentation, genetic algorithm

1 Introduction

In this paper we consider two versions of the *Critical Node Problem (CNP)*. The most canonical on is formulated as follows: given an undirected graph $G(V, E)$ and an integer K, determine a subset of K nodes $S \subseteq V$, such that the number of node pairs still connected in the induced subgraph $G[V \setminus S]$

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

[1] Work supported by a Google Focused Grant on Mathematical Programming, project "Exact and Heuristic Algorithms for Detecting Critical Nodes in Graphs".

[2] Email: roberto.aringhieri@unito.it, grosso@di.unito.it, hosteins@di.unito.it

http://dx.doi.org/10.1016/j.endm.2016.03.047

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 [15] and later Wood [16]. Although these seminal papers focused on arc deletion, more recently the attention is more on node deletion. The CNP has many applications, such as the robustness of communication networks [7,8], detection of so-called "key players" in a relational network [6] or contagion control via vaccination [4,13]. Different optimization methods were proposed for solving the CNP: heuristics [6,4,9,1], Local Search and other metaheuristics [4,13,3], approximation [14] and exact [12] algorithms. The CNP is known to be NP-complete [4] in the general case. For a broad CNP literature review, the reader can refer to [10,11].

Other versions of the problem were studied, such as the Cardinality-Constrained CNP (CC-CNP) [5], which can be described as follows: determine the minimal subset of nodes $S \subseteq V$, such that the largest connected component in the induced subgraph $G(V \setminus S)$ is smaller than a given parameter L. While the objective function now becomes trivial to compute, the cardinality constraint itself is not trivial to satisfy, contrary to the Canonical CNP discussed above. This makes moves in the solution space much more delicate, e.g., for a VNS or ILS algorithm such as those proposed in [2] for the Canonical CNP and it calls for more generally applicable algorithms. [10] and [11] also discuss other CNP formulations where one minimizes the size of the largest component or maximizes the number of connected components by deleting K nodes.

In this paper, we propose a genetic algorithm that is more easily adaptable to the different possible connectivity measures of the CNP, exploiting useful greedy rules for reducing and extending the set S and already described in [4,1]: basically a node n is chosen in S according to $n = \arg\min\{f(S \setminus \{n\}) - f(S)\}$ or in $V \setminus S$ as $n = \arg\max\{f(S) - f(S \cup \{n\})\}$, optimising the impact on the function $f(S)$ (these rules are adaptable to the CC-CNP, as we will develop in Section 2.2). The paper is organized as follows. Section 2 depicts the details of the proposed genetic algorithm for solving the CNP and CC-CNP. Section 3 reports the preliminary computational results of our algorithm, comparing with those reported in [2] and discuss the current extension of the present work.

2 A genetic algorithm for the CNP

2.1 Canonical CNP

As stressed in the introduction, there already exist several heuristic methods for solving the CNP with different kinds of connectivity measure. Those that seem to give the best results at the moment, for the formulation based on Eq. (1) (that we call "Canonical CNP"), are based on efficient local searches that evaluates the impact of 2-nodes exchanges between the set S of deleted nodes and $V \setminus S$ [2].
However the diversification phase is actually crucial to the CNP: even though articulation points may (and do) appear during the local search, with a sizeable impact on the objective function, sometimes a certain amount of nodes must be deleted together in order to disconnect the graph further. Such "co-articulation points" are usually very hard to spot.

Since Genetic Algorithms (GA) are known for their ability to diversify the search, we propose to devise such a GA specifically for the CNP. It should be noted that a GA has already been explored in the context of the CNP [5], more specifically in the context of the Cardinality Constrained CNP, which connectivity measure differs from the one presented in (1). However, its design was quite different from our approach and with no particular use of the problem structure or other existing algorithms. The general pseudo-code of our GA is given in Algorithm 1.

Algorithm 1 *A Genetic Algorithm for the CNP*

CNP-GA (S^*, $\mathcal{P}$, $t_{\max}$, $\mathcal{N}$)
$t \longleftarrow 0$;
Initialise($\mathcal{N}, \mathcal{P}, S^*, \gamma, \pi$);
repeat
  $\mathcal{P}' :=$**New_Generation**($\mathcal{N}, \mathcal{P}, \gamma$);
  Mutate($\mathcal{P}', \pi$);
  $\mathcal{P} :=$**Ordering**($\mathcal{P}, \mathcal{P}', \gamma, S^*$);
  Update(γ, π);
  $t \longleftarrow$ **cpuTime**();
until $t \geq t_{\max}$;
$S^* :=$**Local_search**(S^*);
return S^*

The notation must be understood as follows: $\mathcal{P}$ and $\mathcal{P}'$ are populations of $\mathcal{N}$ individual solutions to the CNP. Individual solutions are represented by the set S of deleted nodes, thus $\mathcal{P}^{(\prime)} = \{S_i^{(\prime)}, i \in \{1, ..., \mathcal{N}\}\}$. A parameter γ is used in the fitness function used to evaluate the solutions and a parameter π is used for the probability of mutation of each newly created solution. The initial solutions of $\mathcal{P}$ are made of K nodes chosen at random in the graph. In order to evaluate the solutions we introduce a fitness function:

$$F(S, \gamma, S^*) = \gamma f(S) + (1 - \gamma)\Sigma(S, S^*).$$

The function $\Sigma(S, S^*)$ computes the number of nodes in S that are also present in the best known solution S^*. Using $F(S, \gamma, S^*)$ instead of connectivity function $f(S)$ (1) allows to maintain a certain diversity among solutions by boosting those that differ from the best one while maintaining a competitive disconnectivity. Parameter γ is modified at each generation as: $\gamma = 1.4 * \langle\Sigma(S, S^*)\rangle_{\mathcal{P}}/(\langle\Sigma(S, S^*)\rangle_{\mathcal{P}} + f(S^*))$, with $\langle\Sigma(S, S^*)\rangle_{\mathcal{P}}$ the average of the function $\Sigma(S, S^*)$ over the population $\mathcal{P}$. π is then initialised at $\pi = 5$.

The reproduction phase is comprised in the **New_Generation** procedure. $\mathcal{N}$ children solutions S_i' are created from the union of two parents S_{i_1} and S_{i_2} from $\mathcal{P}$: $|S_i'| = |S_{i_1} \cup S_{i_2}| \geq K$; the solution is then made feasible using the greedy procedure proposed in [4] and explicited in line 4 of the pseudo-code 2, the only difference with [4] being that we break ties at random between the best nodes at each step until $|S_i'| = K$. This focusses the search towards good quality solutions more quickly.

Algorithm 2 *Reproduction phase*

New_Generation $(\mathcal{N}, \mathcal{P}, \gamma)$
1 $\mathcal{P}' := \{\}$;
for $i = 1...\mathcal{N}$
2 $i_1 := IntRand(\{0, ..., \mathcal{N}\})$; $i_2 := IntRand(\{0, ..., \mathcal{N}\} \setminus \{i_1\})$;
3 $S_i' := S_{i_1} \cup S_{i_2}$;
while $|S_i'| > K$
4 $n = \arg\min\{f(S \setminus \{n\}) - f(S)\}$; $S := S \setminus \{n\}$;
5 $\mathcal{P}' := \mathcal{P}' \cup \{S_i'\}$;
return $\mathcal{P}'$

The mutation phase is implemented so as to diversify the search for good

solutions. For each child solution S'_i, if a flipped coin is less than the mutation parameter π, a number $n_g \in \{1...K\}$ of nodes, determined from a probability distribution function $p(k) \propto 1/k$, are deleted from S'_i (thus favouring smaller values for n_g that do not disturb too much the solution). Using a greedy rule similar to the one used in the reproduction phase, we add back nodes to S'_i until $|S'_i| = K$ (at each step, here again we break ties at random between the best possible nodes). The details can be seen in pseudo-code 3.

Algorithm 3 *Mutation phase*

Mutate $(\mathcal{P}', \pi)$
for $i = 1...\mathcal{N}$
  if $IntRand(\{1, ..., 100\}) \leq \pi$
1 $n_g := IntRand(\{0, ..., K\}, p)$;
  for $j = 1...n_g$
2 $n := S'_i(IntRand(\{1...|S'_i|\}))$; $S'_i := S'_i \setminus \{n\}$;
  while $|S'_i| < K$
3 $n = \arg\max\{f(S) - f(S \cup \{n\})\}$; $S := S \cup \{n\}$;

Populations $\mathcal{P}$ and $\mathcal{P}'$ are then merged together and ordered according to the fitness function $F(S, \gamma, S^*)$, the best individual being updated when a better solution is found. The best $\mathcal{N}$ solutions are selected and then returned to replace those in $\mathcal{P}$.
We finally update the value of γ according to the previously specified rule. As for π, if a new best solution has been found, it is set to $\pi := 5$, otherwise it is increased at $\pi :=$Min$(\pi + 5, 50)$; this allows to diversify the solutions more whenever no improvement of the best solution has been found.

As a last step, a local search is applied on S^*, applying 2-nodes swap between S^* and $V \setminus S^*$ until no more improving move is found.

2.2 Cardinality-Constrained CNP

Let us now see how one can adapt the GA to alternative connectivity measures over the graph, such as with the CC-CNP. In [5], the NP-hardness of CC-CNP is proved and two heuristics are proposed to find solutions, among which a greedy approach very similar to the one of [4]. The greedy procedure is again quite simple: it starts from a set $S \subseteq V$ and moves nodes back from S to $V \setminus S$ until no node can be removed from S without obtaining a con-

nected component of cardinality greater than L. However, we noticed that the criterium for adding back nodes from S to $V \setminus S$ does not discriminate enough between good and bad solutions and thus does not provide very good solutions for difficult graphs. Therefore we propose to select the node which will be integrated in the smallest possible connected component of the graph.

Using this new greedy rule we can adapt the reproduction and mutation procedures of our GA to the CC-CNP without too much trouble, as well as set up a competing greedy algorithm that repeatedly starts from a vertex cover S of G and diminishes $|S|$ until feasibility can no longer be maintained.

3 Preliminary computational analysis and conclusions

In this section we report the preliminary computational analysis of the Genetic Algorithm 1. It was 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 [2]) for the Canonical CNP, 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. We use the parameter values $t_{max} = |V| + |E|$ seconds (given that many steps of the algorithm have $\mathcal{O}(|V| + |E|)$ complexity) and $\mathcal{N} = 300$, as a larger population would make the resolution too slow for the largest graphs. As no benchmark results for the CC-CNP are provided in the litterature for these graphs, we compare with results of a multistarting greedy as described in section 2.2: we launch it $\mathcal{N}$ times for each instance.

The results in Tab. 1 are promising as they are very close to the best known results for almost all graphs. In fact, compared to competing algorithms in [2] it has the best average gap to the best solutions, i.e. 4% while all other solutions have at least 6% average gap: although the set of instances does not allow to statistically confirm this as significant from a Wilcoxon test, this comes as a hint of good robustness for the GA. As a next step, variants of the algorithm will be explored and applied to larger real graphs, as well as other variants of the CNP (e.g., maximising the number of connected components).

	CNP			CC-CNP		
graph	K	BK	GA	L	Greedy	GA
BA500	50	195*	195	25	9	9
BA1000	75	558*	558	50	8	8
BA2500	100	3704*	3704	100	9	9
BA5000	150	10196*	10196	200	11	11
ER235	50	295*	300	30	37	34
ER466	80	1542	1551	50	69	65
ER941	140	5198	5451	100	129	118
ER2344	200	1012849	1171633	250	329	306
FF250	50	194*	194	25	20	20
FF500	110	257*	259	40	19	19
FF1000	150	1260*	1261	70	37	36
FF2000	200	4545*	4565	100	41	38
WS250	70	3241	**3136**	45	82	75
WS500	125	2130	2267	80	110	83
WS1000	200	139653	188317	110	336	267
WS1500	265	14138	15228	150	272	210

Table 1
Results of the GA over the Canonical CNP (BK stands for Best Known result from previous works, while K is the number of nodes deleted from the graph) and over the CC-CNP (L is the maximum cardinality of remaining connected components). BK results with an asterisk are known to be exact optima.

References

[1] Addis, B., R. Aringhieri, A. Grosso and P. Hosteins, *New greedy algorithms for the critical node problem* Submitted for publication.

[2] Aringhieri, R., A. Grosso, P. Hosteins and R. Scatamacchia, *Local search metaheuristics for the critical node problem* Submitted to Networks.

[3] Aringhieri, R., A. Grosso, P. Hosteins and R. Scatamacchia, *Vns solutions for the critical node problem*, in: *3rd International Conference on Variable Neighborhood Search*, Electronic Notes in Discrete Mathematics, 2014, to appear.

[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] Arulselvan A., S. O., Commander C.W. and P. P.M., *Cardinality-constrained critical node detection problem*, in: N. Gülpnar, P. Harrison and B. Rüstem, editors, *Performance Models and Risk Management in Communications Systems*, Springer Optimization and Its Applications **46**, Springer New York, 2011 pp. 79–91.

[6] Borgatti, S. P., *Identifying sets of key players in a network*, Computational and Mathematical Organization Theory **12** (2006), pp. 21–34.

[7] Dinh, T. N. and M. T. Thai, *Precise structural vulnerability assessment via mathematical programming*, in: *MILCOM 2011*, IEEE, 2011, pp. 1351–1356.

[8] Dinh T. N. et al, *On new approaches of assessing network vulnerability: Hardness and approximation on approximation of new optimization methods for assessing network vulnerability*, IEEE/ACM Transactions on Networking **20** (2012), pp. 609–619.

[9] Edalatmanesh, M., "Heuristics for the Critical Node Detection Problem in Large Complex Networks," Ph.D. thesis, Faculty of Mathematics and Science, Brock University, St. Catharines, Ontario (2013).

[10] Shen, S. and J. Cole Smith, *Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs*, Networks **60** (2012), pp. 103–119.

[11] Shen, S., J. C. Smith and R. Goli, *Exact interdiction models and algorithms for disconnecting networks via node deletions*, Discrete Optimization **9** (2012), pp. 172–88.

[12] Summa, M. D., A. Grosso and M. Locatelli, *Branch and cut algorithms for detecting critical nodes in undirected graphs*, Computational Optimization and Applications **53** (2012), pp. 649–680.

[13] Ventresca, M., *Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem*, Computers & Operations Research **39** (2012), pp. 2763–2775.

[14] Ventresca, M. and D. Aleman, *A derandomized approximation algorithm for the critical node detection problem*, Computers and Operations Research **43** (2014), pp. 261–270.

[15] Wollmer, R., *Removing arcs from a network*, Operations Research **12** (1964), pp. 934–940.

[16] Wood, R. K., *Deterministic network interdiction*, Mathematical and Computer Modelling **17** (1993), pp. 1–18.