# Heuristic Algorithm for Identifying Critical Nodes in Graphs

**Dalaijargal Purevsuren[1], Gang Cui[1], Nwe Nwe Htay Win[1], and Xiufeng Wang[1]**

[1] **School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China**
***{dalaijargal, cg, nwenwehtaywin, wxf}@hit.edu.cn***

**Abstract**
The paper presents Greedy Randomized Adaptive Search Procedure with Path Relinking (GRASP with PR) for the Critical Node Detection Problem (CNDP). An evolutionary Path Relinking mechanism is added to GRASP with PR to intensify. Our computational experiments show that this algorithm is a competitive method compared with the previously proposed methods for solving CNDP such as Variable Neighborhood Search and Simulated Annealing.

***Keywords:*** *Combinatorial Optimization; Heuristic Search; GRASP with Path Relinking; Critical Node Detection Problem;*

## 1. Introduction

Identifying a small number of key nodes from a network has important role for many cases in the real world. For example, identifying key members can be used for network immunization when it is expensive to vaccinate all members of network, and only a limited number of members can be vaccinated. The study of covert terrorist networks can be illustrated another application. Destroying a member of a terrorist network may need a large amount of resources. In this case, identifying key members of a terrorist network could be crucial for a government.

Given an undirected graph $G(V, E)$, and an integer k, the identifying critical nodes of a graph involves the finding a set of k nodes $V' \subseteq V$ of the graph whose deletion become the graph with minimum pairwise connectivity [1]. It is called the Critical Node Detection Problem (CNDP). Mathematically, the objective function of CNDP is calculated by the formula defined in eq. (1).

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

Formally, CNDP is to find $V'$ whose objective function $f(V')$ gives minimum value.

An integer linear programming model and proof of NP-completeness for CNDP are provided in [1], and the authors introduce a heuristic on a limited number of network structures with a small number of nodes. Also, CNDP is NP-complete on trees with non-unit edge costs while it is solvable within polynomial time when the input graphs have tree structure with unit edge costs [2], and the authors in [2] propose a dynamic programming method. With limitations on graph's structure, i.e., bounded tree-width, and series parallel graphs, dynamic programming approaches are proposed in [3], [4], and a branch and cut algorithm is presented in [5]. An approximation algorithm based on a randomized rounding is introduced in [6]. For larger general graphs with up to 5000 vertices, Metaheuristics, a simulated annealing (SA) and population- based incremental learning algorithms (PHIL), are proposed and experimentally compared in [7], and the authors create random graphs based on Barabasi-Albert, Erdos-Renyi, Forest-Fire, Watts-Strogatz network models for evaluation. A variable neighborhood search is proposed in [8] and tested on the same instances with [7].

In the paper, we propose GRASP with PR for CNDP and test it on well-known 16 instances defined in [7]. The computational experiments show that the hybrid algorithm is a competitive method compared to previously proposed metaheuristics such as SA, PHIL, and VNS. The remainder of the paper is structured as follows. Section two describes the proposed algorithm in detail. The results of the computational experiments are presented in section three. Finally, we conclude the paper and discuss some future work.

## 2. GRASP with PR for CNDP

GRASP is a multi-start procedure that iteratively executes two main steps, namely construction and local search. The aim of the construction step is to discover a promising basin from solution space while the aim of the local search step is to improve the constructed solution and find the optimal solution of the basin. For a more complete description and recent survey of GRASP, we refer the readers to [9], [10].

This section describes the design of GRASP with Path Relinking (GRASP+PR) for CNDP. Firstly, we give general scheme of hybridization of GRASP with PR. Then, the sub-procedures of the scheme are described in the following sub-sections.

PR, originally proposed as intensification tool in the context of tabu search in [11], can be used as an enhancement to the basic GRASP procedure by exploring trajectories between the two high quality solutions obtained by a heuristic algorithm [9, 10]. A set of the high quality solutions is maintained during GRASP iterations. The set of high quality is called elite set. A member in the elite set is randomly selected for first solution of PR. The second solution is obtained by GRASP. Then PR will relink the two solutions. After that, the new solution created by PR will be a candidate for inclusion in the elite set.

The general scheme of GRASP with PR is given in Fig. 1. The percentage symbol (%) indicates module operator in the line 15.

```
    Input: $iterationNo_{grasp}$, $eliteSet\_size$, $step_{evoPR}$
    Ouput: $s^*$;
1.  $s^* \leftarrow Null$;
2.  $ES \leftarrow \emptyset$;
3.  $t \leftarrow 0$;
4.  while ($t < iterationNo_{grasp}$) do
5.      $s \leftarrow Construction(\cdot)$;
6.      $s_1 \leftarrow LocalSeach(s)$;
7.      if ($|ES| < eliteSet_{size}$) then
8.          $ES \leftarrow ES \cup s_1$;
9.      else
10.         $s_2 \leftarrow SelectRandomSolution(ES)$;
11.         $s_3 \leftarrow PathRelinkin(s_1, s_2)$;
12.         $ES \leftarrow UpdateEliteSet(ES, s_3)$;
13.     end
14.     $t \leftarrow t + 1$;
15.     if ($t \% step_{evoPR} = 0$) then
16.         $ES \leftarrow EvolutionaryPR(ES)$;
17.     end
18. end
19. $s^* \leftarrow$ min $\{ES\}$;
20. return $f(s^*)$;
```

Fig. 1 Pseudocode of GRASP with PR algorithm for CNDP.

## 2.1 Local Search

The aim of the local search step is to explore neighborhoods of the solution obtained by previous construction step in order to find the optimal solution in the region. The standard hill climbing is used for local search in this study. To explore neighborhoods, a neighborhood structure is needed to be defined for a specific problem. In the context of CNDP, neighborhoods of a solution are the set of solutions which can be visited by removing a node from V′ and adding a node among V\V′ into the current solution. Local search stops if the number of iterations is reached the pre-defined limit ($iterationNo_{LS}$ ). The best solution is returned as the local optimal solution.

## 2.2 Path Relinking

Let A and B be two solutions to be relinked by PR. PR procedure starts with A and gradually transforms it into B by successively swapping elements from $A\backslash B$ and $B\backslash A$ [11]. At each step of PR, two nodes are selected by examining contribution in objective function’s value. The first node is a node from A\B that is minimum increase in objective function’s value. The second node is a node from B\A that is maximum increase in objective function’s value. Then the first node is removed from A and the second node is added into A. This procedure is repeated until A reaches B. The pseudocode of PR is presented in Fig. 2. The standard Hill Climbing addressed in section 2.1 is also used for the local search in PR.

```
    Input: $A, B$
    output: $C$
1.  $C \leftarrow A$;
2.  while ($|A \setminus B| > 0$) do
3.      $a \leftarrow \text{argmin}\{f(A \setminus j): j \in A \setminus B\}$;
4.      $b \leftarrow \text{argmax}\{f(B \setminus j): j \in B \setminus A\}$;
5.      $A \leftarrow A \cup b$;
6.      $A \leftarrow A \setminus a$;
7.      $A_{ls} \leftarrow LocalSearch(A)$;
8.      if ($f(C) > f(A_{ls})$) then
9.          $C \leftarrow A_{ls}$;
10.     end
11. end
12. return $LocalSearch(C)$;
```

Fig. 2 Pseudocode of evolutionary PR for CNDP.

## 2.3 Updating Elite Set

Candidate solutions have to satisfy two conditions in order to become a member of the elite set. The first condition is that the candidate solution has to be different with each solution in the current elite set. If the candidate satisfies the first condition, it can be considered as different from the elite set, and will be checked with the second condition. The second condition is that the candidate has to be better than its own parent solution in terms of quality of solution. The candidate is replaced with its parent solution if it satisfies both two conditions.

## 2.4 Evolutionary Path Relinking

PR can also be used as intensification tool of elite sets [10]. It is called evolutionary PR. In the evolutionary PR, PR is periodically applied to several members of the elite set. In the study, PR is called in every $step_{evoPR}$ iterations and each pair in the elite set is relinked by PR. The resulting solution of each evolutionary path relinking will be a candidate for inclusion in elite set.

## 3. Computational Experiments

In this section, we will address about computational experiment and comparison of the performance of the proposed algorithm with the previously proposed methods for CNDP. We use instances from [7] for evaluation and the detailed information of the instances is presented in Table 1.

Table 1: Sizes of the graphs from [7]

| *Name of instance* | *Vertices* | *Edges* | *k-Critical Nodes* |
|---|---|---|---|
| ER250 | 235 | 250 | 50 |
| ER500 | 466 | 700 | 80 |
| ER1000 | 941 | 1400 | 140 |
| ER2500 | 2344 | 3500 | 200 |
| BA500 | 500 | 499 | 50 |
| BA1000 | 1000 | 999 | 75 |
| BA2500 | 2500 | 2499 | 100 |
| BA5000 | 5000 | 4999 | 150 |
| WS250 | 250 | 1246 | 70 |
| WS500 | 500 | 1496 | 125 |
| WS1000 | 1000 | 4996 | 200 |
| WS1500 | 1500 | 4498 | 265 |
| FF250 | 250 | 514 | 50 |
| FF500 | 500 | 828 | 110 |
| FF1000 | 1000 | 1817 | 150 |
| FF2000 | 2000 | 3413 | 200 |

### 3.1. Experimental setup

There were several design and parameter selection issues while developing the proposed algorithm. The number of maximum iterations ($\mathrm{iterationNo}_{\mathrm{grasp}}$) the elite set size ( eliteSet_size ), and the step for evolutionary PR ($\mathrm{step}_{\mathrm{evoPR}}$). We executed several preliminary experiments to define these parameters. For all the runs, the parameter values used are presented in Table 2. Running time for GRASP+PR is limited to 3600 seconds.

Table 2: Parameters used for all experiments

| *Parameters* | *Values* |
|---|---|
| $iterationNo_{\mathrm{grasp}}$ | 100 |
| *eliteSet_size* | 3 |
| $step_{\mathrm{evoPR}}$ | 10 |

The proposed algorithm was implemented in C++ and compiled with gcc 4.9.3. It was tested on a PC equipped with a 2.7GHz AMD Athlon(tm) II X2 215 of CPU, and 4.0 GB of RAM.

### 3.2. Experimental results and comparison with previously proposed methods

The summary of experimental results (the value of the objective function defined in eq. (1)) is given in table 3 in detail. These results have been summarized from 10 independent runs. GRASP+PR performs in stable except ER2500, WS250, and WS1000 instances because the standard deviation is high for these three instances.

Table 3: Summary results (mean (μ), standard deviation (σ), minimum (min) and maximum (max)) for GRASP+ePR

| *Instance* | *GRASP+PR* | | | |
|---|---|---|---|---|
| | μ | σ | min | max |
| ER250 | 298.4 | 1.6 | 297 | 301 |
| ER500 | 1627.2 | 33.1 | 1575 | 1673 |
| ER1000 | 5788.0 | 179.9 | 5482 | 5948 |
| ER2500 | 1069360.8 | 15947.6 | 1048464 | 1091041 |
| BA500 | 195.0 | 0.0 | 195 | 195 |
| BA1000 | 558.0 | 0.0 | 558 | 558 |
| BA2500 | 3704.0 | 0.0 | 3704 | 3704 |
| BA5000 | 10196.0 | 0.0 | 10196 | 10196 |
| WS250 | 11800.6 | 1856.6 | 9351 | 14734 |
| WS500 | 2233.4 | 30.1 | 2209 | 2263 |
| WS1000 | 306413.0 | 8756.9 | 297241 | 318801 |
| WS1500 | 14869.2 | 358.7 | 14177 | 15167 |
| FF250 | 194.0 | 0.0 | 194 | 194 |
| FF500 | 258.6 | 1.0 | 257 | 260 |
| FF1000 | 1263.1 | 1.4 | 1261 | 1266 |
| FF2000 | 4561.5 | 5.0 | 4554 | 4568 |

Recently proposed three metaheuristics, i.e., VNS in [8], and SA and PHIL in [7] are used for the comparison of the performance of GRASP+PR. To evaluate these methods on a same environment, we have re-implemented these methods in C++ following the guidelines in the original publications. Running times for all methods are limited to 3600 seconds. The objective function's values of experiments are presented in table 4. In table 5, the computational time of GRASP+PR is displayed with its main competitor, VNS. VNS is stopped if it meets one of the following two conditions: (1) the number of iterations without improvement is over 1000 and (2) running time reaches 3600 seconds. It is important to note that for SA and PHIL methods, the quality of solution obtained by them is far from that of solutions obtained by GRASP+PR. Therefore, comparison of GRASP+PR with SA and PHIL in terms of computational time does not make sense.

From results in table 4 and 5, we highlight the following elements:

- GRASP+PR finds the best objective function's values in 13 of 16 instances.

- GRASP+PR is faster than VNS that is the main competitor of GRASP+PR in terms of quality of solution (see Table 4).

Table 4: Comparison of the minimum value of objective function for SA, PHIL, VNS, and GRASP+PR in terms of quality of solution

| *Instance* | *SA* | *PHIL* | *VNS* | *GRASP+PR* |
|---|---|---|---|---|
| ER250 | 7700 | 6700 | 298 | **297** |
| ER500 | 48627 | 44255 | **1542** | 1575 |
| ER1000 | 234479 | 229576 | 5628 | **5482** |
| ER2500 | 2011122 | 2009132 | 1052406 | **1048464** |
| BA500 | 997 | 892 | **195** | **195** |
| BA1000 | 3770 | 3057 | 559 | **558** |
| BA2500 | 31171 | 28044 | **3704** | **3704** |
| BA5000 | 170998 | 146753 | **10196** | **10196** |
| WS250 | 14251 | 13786 | **6610** | 9351 |
| WS500 | 54201 | 53779 | 2230 | **2209** |
| WS1000 | 311700 | 308596 | **154813** | 297241 |
| WS1500 | 717369 | 703241 | 15692 | **14177** |
| FF250 | 1841 | 1386 | **194** | **194** |
| FF500 | 2397 | 1904 | **257** | **257** |
| FF1000 | 92800 | 59594 | 1263 | **1261** |
| FF2000 | 387248 | 256905 | 4584 | **4554** |

The best of four results is displayed in bold font

Table 5: Comparison of computational time for VNS and GRASP+PR

| *Instance* | *VNS* | *GRASP+PR* |
|---|---|---|
| ER250 | 116.1 | 15.6 |
| ER500 | 913.6 | 133.3 |
| ER1000 | 3600.0 | 980.0 |
| ER2500 | 3600.0 | 3600.0 |
| BA500 | 301.4 | 35.0 |
| BA1000 | 1874.1 | 187.0 |
| BA2500 | 3600.0 | 1905.0 |
| BA5000 | 3600.0 | 3600.0 |
| WS250 | 1352.6 | 21.3 |
| WS500 | 3600.0 | 187.0 |
| WS1000 | 3600.0 | 777.0 |
| WS1500 | 3600.0 | 2779.0 |
| FF250 | 41.6 | 12.1 |
| FF500 | 419.2 | 63.0 |
| FF1000 | 3600.0 | 594.5 |
| FF2000 | 3600.0 | 3600.0 |

## 4. Conclusions and future work

We have proposed GRASP with Path Relinking for the Critical Node Detection Problem. The algorithm has been tested on the well-known 16 instances that have from 235 to 5000 nodes and compared with the previously proposed three methods. The proposed algorithm finds the best solution in 13 of 16. In addition, GRASP with Path Relinking is faster than the main competitor, VNS.

This work can be extended as creating larger instances and testing the proposed algorithm on them. Also, we propose a simple swapping method based on greedy strategy for the moving of Path Relinking. Hence, other variants of the Path Relinking for Critical Node Detection Problem may exist and examining these variants could be a promising work.

## References

[1] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos, “Detecting critical nodes in sparse graphs”, Computers & Operations Research, Vol. 36, No. 7, 2009, pp. 2193–2200.

[2] M. Di Summa, A. Grosso, and M. Locatelli, “Complexity of the critical node problem over trees”, Computers & Operations Research, Vol. 38, No. 12, 2011, pp. 1766–1774.

[3] B. Addis, M. Di Summa, and A. Grosso, “Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth”, Discrete Applied Mathematics, Vol. 161, No. 16–17, 2013, pp. 2349–2360.

[4] S. Shen and J. C. Smith, “Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs”, Networks, Vol. 60, No. 2, 2012, pp. 103–119.

[5] M. Di Summa, A. Grosso, and M. Locatelli, “Branch and cut algorithms for detecting critical nodes in undirected graphs”, Computational Optimization and Applications, Vol. 53, No. 3, 2012, pp. 649–680.

[6] M. Ventresca and D. Aleman, “A derandomized approximation algorithm for the critical node detection problem”, Computers & Operations Research, Vol. 43, 2014, pp. 261–270.

[7] M. Ventresca, “Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem”, Computers & Operations Research, Vol. 39, No. 11, 2012, pp. 2763–2775.

[8] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia, “VNS solutions for the Critical Node Problem”, Electronic Notes in Discrete Mathematics, Vol. 47, 2015, pp. 37–44.

[9] M. G. C. RESENDE and C. C. Ribeiro, “GRASP: Greedy randomized adaptive search procedures”, in Search Methodologies - Introductory tutorials in optimization and decision support systems, 2nd ed., Springer, 2014, pp. 287–312.

[10] P. Festa and M. G. C. RESENDE, “Hybridizations of GRASP with Path-Relinking”, in Hybrid Metaheuristics, Vol. 434, Berlin Heidelberg: Springer, 2013, pp. 135–155.

[11] F. Glover, “Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges”, in Interfaces in Computer Science and Operations Research, Vol. 7, Springer US, 1997, pp. 1–75.