Hybridization of GRASP with Exterior Path Relinking for Identifying Critical Nodes in Graphs

Dalaijargal Purevsuren, Gang Cui, Mingcheng Qu, and Nwe Nwe Htay Win

***Abstract*—This paper deals a problem with detecting the critical nodes in order to achieve minimum pair-wise connectivity in the residual graph after removing a limited number of nodes from the graph. It is called the critical node detection problem (CNDP). In this paper, we propose a new hybridization scheme of greedy randomized adaptive search procedure (GRASP) with exterior path-relinking. Exterior path-relinking, a recently proposed metaheuristic method, creates paths of successive solutions starting from two high-quality solutions to other high quality solutions. A randomly selected solution from the elite solution pool collecting throughout GRASP iterations and the resulting solution from each GRASP iteration are relinked by exterior path-relinking for finding better solutions. The proposed algorithm is assisted on test instances from the literature. Computational experiments demonstrate that the proposed method outperforms other existing algorithms in the area of CNDP.**

***Index Terms*—critical node detection problem, exterior path-relinking, greedy randomized adaptive search procedure, heuristic algorithms**

I. Introduction

THE critical node detection problem (CNDP) finds a set of nodes (S) to be removed from a given undirected graph $G=(V, E)$ in order to fragment the graph. The aim of the problem discussed in this paper is to minimize the number of pair-wise connectivity of the nodes in the remaining graph $G[V \setminus S]$. A set of k critical nodes S is searched to meet with the objective function as shown in (1) so that the total number of pair-wise connectivity of nodes in the remaining graph can be minimized. An integer linear programming based definition and the proof of NP-hard for CNDP are presented in [1].

$$f(S) := |\{u, v \in V \setminus S: u \text{ and } v \text{ are in a same component of } G[V \setminus S]\}| \quad (1)$$

CNDP problem has been recently attracting some attentions in many applications. Indeed, identifying a small number of key nodes in a network plays an important role for many cases in the real world. For example, when it is expensive to vaccinate all members of a network, a possible solution is to vaccinate key members of the network so it is essential to find them in this research [2]. Detecting critical members in a network are widely used not only in academic area but also in political issues i.e., it may be useful to find pivotal members of a terrorist network for a government. It would be effective to maximize damage of the terrorist network if the key members are destroyed [3].

In this paper, we introduce a hybridization of GRASP with exterior path-relinking (GRASP with ePR) to identify the critical nodes in graphs. To the best of our knowledge, we are the first of introducing GRASP with ePR in CNDP, and this paper focuses to show how ePR could improve GRASP with our proposed GRASP with ePR algorithm. Regarding to path-relinking, there are two variants, namely interior path-relinking and exterior path-relinking introduced in [4] and [5], respectively. Interior path-relinking is firstly introduced in the context of GRASP by Laguna and Marti [6]. In the last decade, the hybridization of GRASP with interior path-relinking demonstrates its ability that provides high quality outcomes for some types of hard optimization problems.

Although many researchers have paid great attention of GRASP improvement by combining with interior path-relinking, there are only few researches that consider "exterior" path-relinking in the context of GRASP. In this study, we consider exterior path-relinking to intensify GRASP in a new way with our proposed GRASP with ePR algorithm. Exterior path-relinking is used as intensification method to explore the neighbors beyond the resulting solution of each GRASP iteration and a randomly selected member from the elite set as well. Elite set is a small-sized set of high quality solutions that is obtained throughout GRASP iterations. In addition, we modify the local search usage in exterior path-relinking defined in its original proposal [5]. The modification to local search usage is accomplished by applying a local search procedure to each member in a path instead of considering only the best member in a path. The modification is experimentally evaluated in section IV-*B*.

Manuscript received July 30, 2016; revised December 30, 2016. This work is supported by the National Science Foundation of China under Grant No.61402131; the China postdoctoral science foundation under Grant No.(2014M551245, 2016T90293); the Heilongjiang postdoctoral science foundation under Grant No.LBH-Z13105 and the Fundamental Research Funds for the Central Universities under Grant No.HIT.NSRIF.201651.

Dalaijargal Purevsuren is with the Harbin Institute of Technology, Harbin, 150001, China (corresponding author, phone: 86-18746017740; e-mail: dalaijargal@gmail.com).

Gang Cui is with School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. (e-mail: cg@ hit.edu.cn).

Mingcheng Qu is with School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. (e-mail: qumingcheng@hit.edu.cn).

Nwe Nwe Htay Win is with School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. (e-mail: nwenwehtaywin@yahoo.com).

The proposed algorithm is tested on benchmark test instances from the literature. The performance of GRASP with ePR is compared with existing other methods for solving CNDP such as simulated annealing (SA) [8], population-based incremental learning (PBIL) [8], and Variable Neighborhood Search (VNS) [9]. The results of computational experiments show that GRASP with ePR outperforms the other methods.

The remainder of the paper is structured as follows. In section two, the related works to CNDP and GRASP with path-relinking are critically reviewed and analyzed. Section three describes the proposed algorithms in detail. The results of the computational experiments are presented in section four. Finally, we conclude the paper and discuss some possible future work.

II. Related Works

A. CNDP

CNDP finds a set of certain nodes for network fragmentation. A plenty of researches [8], [9], [10], [11] have focused on the issues of detecting critical nodes in graphs with significant methodologies and algorithms. The work described in [8] approached CNDP problem using population-based incremental learning and simulated annealing optimization algorithms with the help of a combinatorial unranking-based problem representation. The study defined in [9] presented two metaheuristics for CNDP using iterated local search and variable neighborhood search frameworks.

B. GRASP

To grasp the underlying methodology of GRASP, the related literatures are studied as follows. The GRASP is a multi-start metaheuristic introduced by Feo and Resende, 1989, 1995, [12], [13]. A recent complete survey of this method is presented in the work [14]. As far as we studied the literature related with GRASP, we can see them into two kinds of researches such as those which consider interior path-relinking and those which exploit exterior path-relinking in GRASP.

1) GRASP with exterior path-relinking

A new variant of path-relinking, exterior path-relinking, is proposed by Glover [5]. Instead of exploring trajectories "between" solutions, exterior path-relinking focuses on trajectories "beyond" two solutions. They regarded exterior path-relinking as beyond-form of path-relinking that focuses on its relevance for diversification in binary optimization.

Abraham Duarte *et al*. 2015 [7] proposed several types of new hybrid heuristics which are composed of GRASP with sampled greedy construction and exterior path-relinking. They firstly introduced exterior path-relinking in the context of GRASP for solving the differential dispersion problem. The heuristic maintains an elite set of high-quality solutions throughout the GRASP iterations to integrate exterior path-relinking. Exterior path-relinking is used after a fixed number of iterations for only intensifying the elite set. Extensive computational experiments are executed, and experimental results show the competitiveness of this algorithm.

2) GRASP with interior path-relinking

The idea of using interior path-relinking (also called path-relinking) in GRASP procedure was proposed by Laguna and Marti [6]. Celso C. Ribeiro *et al*. [15] reviewed the fundamentals and implementation strategies about interior path-relinking for advanced hybridizations with metaheuristic schemes such as GRASP, genetic algorithms, tabu search, scatter search, etc. The readers may explore the studies [14], [15] for detailed survey of GRASP with interior path-relinking.

III. A New Hybridization Scheme of Exterior Path-Relinking with GRASP

This paper introduces a new hybridization scheme of exterior path-relinking with GRASP. First of all, we describe the structure of proposed hybridization scheme and later the inside sub-procedures will be presented in the following sub-sections in detail. GRASP developed by Feo and Resende [12], [13] is a multi-start meta-heuristics that iteratively executes two main phases, namely construction and local improvement procedures. At every iteration of GRASP, a solution is created by a construction method, and improved by a local improvement method. Both of construction and local improvement methods are needed to be defined for the concrete problem. These two phases are repeated until they reach to a pre-defined number of iterations. For a more complete description and recent surveys of GRASP, we refer the readers to the studies [14].

In our GRASP with ePR algorithm, we propose a construction and local improvement method for CNDP. A small-sized set of high quality solutions (elite set) is maintained throughout GRASP iterations to integrate exterior path-relinking. In this scheme, exterior path-relinking is used as intensification method for GRASP. The resulting solution obtained from each GRASP iteration is relinked with a randomly selected member from the elite set by using exterior path-relinking. Exterior path-relinking creates a path by exploring neighbors beyond the two solutions. The best solution in the path will be a candidate for inclusion in the elite set. An updating rule for the elite set is defined in section III-*D*. Path-relinking can also be used as intensification tool for elite set [15]. To intensify the elite set, path-relinking is periodically applied to several members of the elite set. In this study, only interior path-relinking is used for this purpose.

The pseudocode of GRASP with ePR is presented in Algorithm 1, where *ES* is the elite set, t is the counter of GRASP iterations, q is the number of consecutive iterations without an improvement, *elite_set_size* is the size of elite set, τ is the step of evolutionary path-relinking, symbol % indicates the module operation, and *min* finds the element whose objective function has minimum value. The detailed processes of each function used in GRASP with ePR are explained in the following sub-sections.

A. Construction Phase

We propose a construction method based on a random walk technique. A random walk on a graph creates a path that consists of succession of random steps over the graph. To make a random step from a node, the next node is

randomly selected from the current node's neighbors based on transition probabilities. For more detailed explanation of random walk, the readers are referred to [16].

Algorithm 1: GRASP_ePR

```
1:  ES ← ∅;
2:  t ← 0;
3:  q ← 0;
4:  repeat
5:      s ← Construction();
6:      s1 ← Local_Improvement(s);
7:      if (|ES| < elite_set_size) then
8:          ES ← ES ∪ s1;
9:      else
10:         s2 ← Select_Random_Solution(ES);
11:         s3 ← Exterior_Path_Relinking(s2, s1);
12:         ES ← Update_Elite_Set(ES, s3, q);
13:     end
14:     t ← t + 1;
15:     if (t % τ = 0) then
16:         ES ← Intensifying_Elite_Set(ES);
17:     end
18: until (reach to termination condition);
19: return min{ES};
```

In this study, we consider that all nodes have the same roles, so transition probabilities for all nodes equal to 1/|*V*| where |*V*| represents the number of vertices of the input graph. Random walk starts a selected node, and then randomly selects the next node with a probability of 1/|*V*|. The length of the path created by the random walk is 2**k* where *k* is the number of critical nodes. The set of nodes in the path can be considered as a solution for CNDP, but it is infeasible because the number of nodes in the path is larger than *k*. To make the solution feasible, we use an algorithm proposed in [1] that iteratively removes one node at a time in which contribution of objective function value is minimum. The pseudocode of the construction phase is presented in Algorithm 2, where N_u is the neighbors of node *u*, *S* is the constructing solution, and *random*(*Q*) is a function that randomly selects a node from a set of vertices *Q*. The output of *Construction*() is a feasible CNDP solution.

Algorithm 2: *Construction*()

```
1:  S ← ∅;
2:  u ← random(V);
3:  S ← S ∪ u;
4:  repeat
5:      v ← random(Nu);
6:      S ← S ∪ v;
7:  until (|S| > 2*k);
8:  repeat
9:      w = argmin{f(S \ {w}) – f(S)};
10:     S ← S \ {w};
11: until (|S| = k)
12: return S;
```

B. Local Improvement Phase

We use a simple neighborhood structure for local improvement function. Neighborhood of a solution is defined as a set of solutions that can be visited by swapping two nodes: which is one from S while another is from $V \setminus S$. With this structure, local improvement is accomplished by executing the swapping | *V* | times. The neighbor solution that has maximum improvement in objective function value and has better improvement than the current solution is selected as the next move. Local improvement stops if the number of consecutive attempts without an improvement reaches to the pre-defined limit (λ) that is an input parameter. The best solution is returned as the local optimal solution. This procedure is indicated in Algorithm 1 as "*Local_Improvement*(*s*)" which receives a solution as input, improves it, and finally the best improved solution is returned as output.

C. Exterior Path-Relinking

Exterior path-relinking generates the paths beyond (exterior) two high quality solutions in the neighborhood space. Let *A* and *B* are the high quality (local optimal) CNDP solutions to be linked by exterior path-relinking. To shed some light on the problem of exterior path-relinking, set representation diagrams are presented in Fig. 1 where Ω is the CNDP solution space. Exterior path-relinking solely focuses on the elements that lie in $A \cap B$ and $\Omega \setminus (A \cup B)$, and there are no changes on elements that lie in $A \setminus B$ and $B \setminus A$ (see Fig. 1 (b) and (c)).

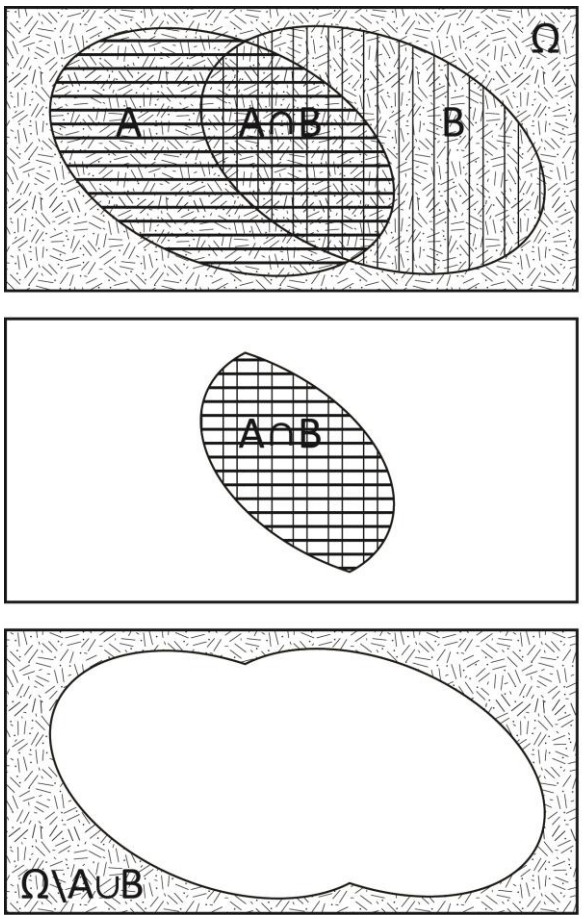


Fig. 1. A set representation diagram for solution *A* and *B* where white space indicates inactive region for exterior path-relinking.

The processes of exterior path-relinking are as follows. First of all, it starts with A and gradually transforms itself with a direction which becomes farther from A and B by successively removing elements from $A \cap B$, and adding elements from $\Omega \setminus (A \cup B)$. The pseudocode of exterior path-relinking is described in Algorithm 3 that receives two solutions as input, relinks them, and returns the best solution as output where ρ ($0 < \rho \leq 1$) is an input parameter. A conceptual illustration of exterior path-relinking is presented in Fig. 2.

In exterior path-relinking algorithm, every step selects two elements by examining contribution in objective function's value. The first element is a node from $A \cap B$ whose deletion from A is a minimum increase (worst node in A) in objective function's value (line 4). It is the node having minimum impact in objective function. The reason is because the objective function value increases or does not change by deleting an element from A i.e., $f(A) \leq f(A \setminus \{i\})$ and therefore, the node that has minimum objective function value by deleting it from A has minimum impact in objective function value of A. The second element is a node from $\Omega \setminus (A \cup B)$ whose insertion to B is maximum improvement (best node in B) in objective function's value (line 5). That is because the objective function value decreases or does not change by adding a node into solution B i.e., $f(B) \geq f(B \cup \{i\})$ and therefore, the node having minimum objective function value by adding it to B has maximum impact in objective function value. The first element is then removed from A and the second element is added into A (line 6 and 7 respectively). The modified solution A is improved by using the local search method defined in section III-*B*. If the improved solution is better than the current best solution, the best solution is replaced with the new solution. These steps are repeated until the termination condition meets.

Algorithm 3: *Exterior_Path_Relinking* (A, B)

1: $C \leftarrow A$;
2: $N \leftarrow |A \cap B|$;
3: **repeat**
4: $a \leftarrow \text{argmin}\{f(A \setminus \{i\}) : i \in A \cap B\}$;
5: $b \leftarrow \text{argmin}\{f(B \cup \{i\}) : i \in \Omega \setminus (A \cup B)\}$;
6: $A \leftarrow A \setminus \{a\}$;
7: $A \leftarrow A \cup \{b\}$;
8: $A_{ls} \leftarrow$ *Local_Improvement*(A);
9: **if** ($f(A_{ls}) < f(C)$) **then**
10: $C \leftarrow A_{ls}$;
11: **end**
12: **until** ($|A \cap B| < (1-\rho)*N$);
13: **return** C;

D. Updating Elite Set

The candidate solution for inclusion in the elite set is replaced with its own parent solution in the elite set if it satisfies the following two conditions:

1. It is better than its own parent in terms of the quality of solution
2. It is different from all member of the elite set.

This process refers to "*Update_Elite_Set*(ES, s_3, q)" of Algorithm 1. It receives the elite set (ES), a candidate solution (s_3), and a variable for counting the number of consecutive iterations without an improvement (q) as input. Then it carries out the abovementioned updating conditions. If the candidate satisfies the updating conditions, the updated elite set is returned as output, and q is set by zero. If the candidate cannot satisfy the conditions, it is simply discarded, q is increased by one, and the last elite set is returned as output.

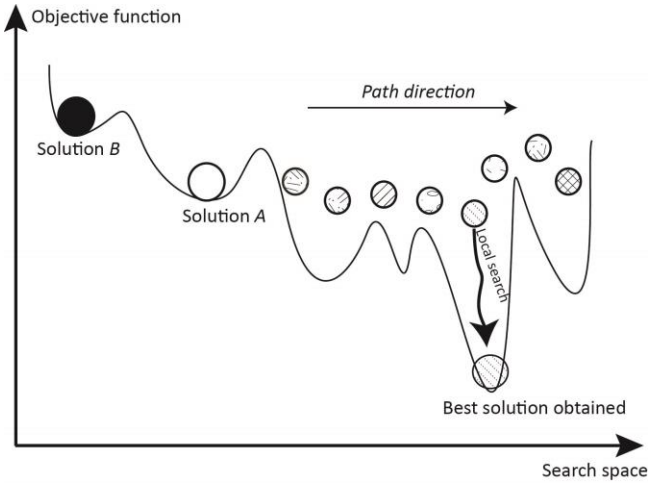


Fig. 2. A conceptual illustration of exterior path-relinking.

E. Intensifying Elite Set (Also known as evolutionary path-relinking)

The members in elite set do not mix with each other because the updating rule we propose does not allow it. However, mixing the members in elite set each other could give promising solutions. For this propose, we use interior path-relinking to intensify the elite set. Therefore, in our paper, interior path-relinking is called at every fixed number of iterations to intensify the elite set. Each pair of the elite set is relinked by executing interior path-relinking, and the best solution in the path will be a candidate for inclusion in the elite set. The initial solution (solution A in Algorithm 4) of interior path-relinking is considered as its parent. This procedure is denoted by "*Intensifying_Elite_Set*(ES)" in Algorithm 1. It receives the elite set as input, intensifies that elite set, and returns the intensified elite set as output. Interior path-relinking is described in the following section.

F. Interior Path-Relinking

We propose interior path-relinking (also known as path-relinking) to relink two high quality CNDP solutions in the elite set. Interior path-relinking procedure starts with A, and it is gradually transformed into B by successively swapping elements from $A \setminus B$ and $B \setminus A$. The pseudocode of interior path-relinking is presented in Algorithm 4. It receives two solutions as input, relinks them, and returns the best solution as output. At each step of interior path-relinking, two elements are selected by examining contribution in objective function's value. The first element is a node from $A \setminus B$ whose deletion from A is a minimum increase (worst node in A) in objective function's value (line 3). The second element is a node from $B \setminus A$ whose deletion from B is a maximum

increase (best node in B) in objective function's value (line 4). Then the first element is removed from A and the second element is added into A (line 5 and 6 respectively). The resulting solution A is improved by executing the local search method defined section III-*B* (line 7). If the improved solution is better than the current best solution in the path, the best solution is replaced with the new solution. These steps are repeated until solution A reaches to solution B.

Algorithm 4: *Interior_Path_Relinking* (A, B)

1: $C \leftarrow A$;
2: **repeat**
3: $a \leftarrow \text{argmin}\{f(A \setminus \{i\}): i \in A \setminus B\}$;
4: $b \leftarrow \text{argmax}\{f(B \setminus \{i\}): i \in B \setminus A\}$;
5: $A \leftarrow A \setminus \{a\}$;
6: $A \leftarrow A \cup \{b\}$;
7: $A_{ls} \leftarrow$ *Local_Improvement*(A);
8: **if** ($f(A_{ls}) < f(C)$) **then**
9: $C \leftarrow A_{ls}$;
10: **end**
11: **until** ($|A \setminus B| = 0$);
12: **return** C;

G. Termination conditions

We use two different termination conditions for GRASP_ePR defined in Algorithm 1. The first condition is based on the total number of GRASP iterations. The maximum number of iterations ($T1$) is defined by input parameter. The GRASP_ePR with iteration based a termination condition is denoted by "GRASP_ePR1" in section IV. The second condition is based on the number of consecutive iterations without an improvement. The maximum number of consecutive iterations without an improvement ($T2$) is defined as input parameter. The GRASP_ePR with a termination condition based on consecutive no improvement is denoted by "GRASP_ePR2". These two termination conditions enable us to control trade-off between running time and the solution quality.

IV. Computational Experiment

In this section, we evaluate the performance of proposed GRASP_ePR1 and GRASP_ePR2 algorithms. For testing purposes, we use benchmark test instances defined in [8], publicly available at http://individual.utoronto.ca/mventresca/cnd.html. The main characteristics (number of vertices and edges of graphs, the number of critical nodes to be removed, and the number of cut points of graphs) and optimal solution's objective function value (OFV) of the benchmark test instances are presented in Table I.

The proposed algorithms are implemented in C++ and compiled with gcc 4.9.3 using optimization flags –O2. All experiments were performed on a Lenovo machine equipped with Windows 7 x64 operating system and an Intel Core i5-3210M (CPU 2.50GHz) processor, RAM 6GB. Running time is limited to 5000 seconds for all algorithms assisted in this work.

A. Parameter study

As mentioned in section III, termination conditions enable us to control trade-off between running time and solution quality. We design GRASP_ePR1 with a goal that finds good quality solutions in reasonable time, whereas GRASP_ePR2 is designed to find high quality solutions without much concerning running time except the general requirement of 5000s for running time limit. To define promising region of parameters, a large number of preliminary experiments are executed. As a result, several possible choices for each parameter are considered as the following: λ = {5, 10, 15, 20}; τ = {4, 6, 8, 10}; ρ = {0.1, 0.3, 0.5, 0.7}; $T1$ = {25, 50, 75, 100}; $T2$ = {2, 10, 20, 30}. Parameters' value is selected from experimental evaluation results with a great care of the design goal of each algorithm. The parameter *elite_set_size* is a standard parameter in GRASP with path-relinking. As discussed in [17], parameter *elite_set_size* should be small (around ten). In our case, the number of total iterations is relatively small. In addition, updating conditions for the elite set are strict to mix elite set members each other. Thus, the parameter *elite_set_size* is selected by a smaller value "3". We selected two representative test instances, WS500 and ER500, for experimental evaluation of parameter study and investigation on GRASP with ePR because they are instances of the hardest graphs for CNDP [9], and their sizes are moderate. The average value over 30 independent runs for each parameter combination is considered for parameter evaluation.

TABLE I
CHARACTERISTICS OF BENCHMARK TEST INSTANCES

Benchmark test instances	Vertices	Edges	Critical nodes	# of cut points	Optimal solution's OFV [a]
ER250	235	250	50	48	N/A
ER500	466	700	80	84	N/A
ER1000	941	1400	140	177	N/A
ER2500	2344	3500	200	419	N/A
BA500	500	499	50	164	195
BA1000	1000	999	75	324	558
BA2500	2500	2499	100	825	3704
BA5000	5000	4999	150	1672	10196
WS250	250	1246	70	0	N/A
WS500	500	1496	125	0	N/A
WS1000	1000	4996	200	0	N/A
WS1500	1500	4498	265	0	N/A
FF250	250	514	50	83	194
FF500	500	828	110	195	257
FF1000	1000	1817	150	362	1260
FF2000	2000	3413	200	725	4545

[a] Optimal solution's OFVs are reported in [9]. The optimal solutions are calculated by using an exact method with running time limit of 5 days.

GRASP_ePR's result over different parameter values is reported in Table II - VI where Cost is the objective function value and Time is the running time. First parameter λ is evaluated with fixed values of $T1$ = 50, τ = 6, and ρ = 0.3 (see Table II). Then, parameter τ is evaluated with fixed values of $T1$ = 50, λ = 5, and ρ = 0.3 (see Table III). After that, parameter ρ is evaluated with fixed values of $T1$ = 50, λ = 5, and τ = 6 (see Table IV).

TABLE II
GRASP_EPR'S RESULTS FOR EVALUATION ON PARAMETER λ

	5		10		15		20	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time
WS500	2175	246	2146	479	2137	719	2120	914
ER500	1603	74	1584	149	1561	199	1549	269

TABLE III
GRASP_EPR'S RESULTS FOR EVALUATION ON PARAMETER τ

	4		6		8		10	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time
WS500	2181	204	2178	193	2188	184	2197	170
ER500	1612	89	1603	79	1606	77	1604	77

TABLE IV
GRASP_EPR'S RESULTS FOR EVALUATION ON PARAMETER ρ

	0.1		0.3		0.5		0.7	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time
WS500	2197	44	2172	170	2174	281	2179	369
ER500	1628	23	1591	78	1589	136	1595	224

From Table III, when parameter τ equal to 6, the algorithm is more likely to be efficient and effective. According to the experimental results in Table IV, when parameter ρ is higher than 0.3, there is almost no benefit in the solution quality and large increase in time consumption. Thus, we select ρ = 0.1 for GRASP_ePR1, and ρ = 0.3 for GRASP_ePR2 with considering their design goals. When parameter ρ = 0.1, the algorithm saves time, but the solution quality is not high. To compensate the solution quality of GRASP_ePR1, another parameter is assigned by a value, parameter λ = 20, with high solution quality. We select parameter λ = 5 for GRASP_ePR2 because it already has a slow but a good for quality parameter value i.e., ρ = 0.3.

Finally, termination conditions for both algorithms, parameter $T1$ and $T2$, are evaluated with fixed values of τ = 6, λ = 20, and ρ = 0.1 for GRASP_ePR1, and τ = 6, λ = 5, and ρ = 0.3 for GRASP_ePR2. There is a sharp decrease in OFV with $T1$ = 50 according to the results in Table V. With increase in $T1$ value such as 75 and 100 there is a slight improvement in OFV and a huge increase in time consumption. Therefore, parameter $T1$'s value is set by "50" with concerning its design goal.

GRASP_ePR obtains steady improvements in OFV until value of T2 reaches 20 (see Table VI). To promote the goal of GRASP_ePR2, parameter $T2$ is set by "20". The selected parameters for algorithms are summarized in Table VII.

TABLE V
GRASP_EPR'S RESULTS FOR EVALUATION ON PARAMETER $T1$

	25		50		75		100	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time
WS500	2201	111	2155	193	2147	288	2139	386
ER500	1613	49	1580	102	1570	140	1565	175

TABLE VI
GRASP_EPR'S RESULTS FOR EVALUATION ON PARAMETER $T2$

	2		10		20		30	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time
WS500	2200	134	2171	337	2146	502	2145	777
ER500	1661	40	1598	127	1580	169	1574	269

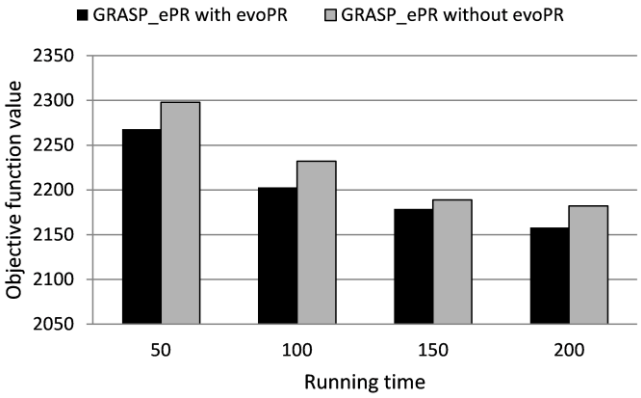
B. Some investigations on GRASP with ePR

In this section, we investigated two components, not typical in this field, of the proposed GRASP with ePR, namely having evolutionary path-relinking (we called it "evoPR"), and calling local search (LS) method for each member of paths generated by path-relinking (we called it "path-inside LS"). As mentioned in previous section, WS500 and ER500 are also used for this investigation.

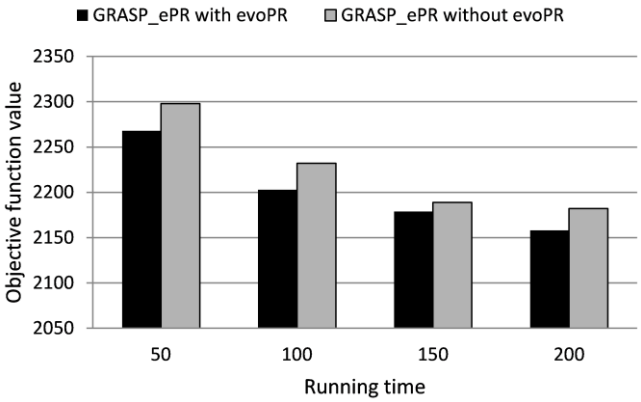
TABLE VII
PAREMETERS USED IN GRASP_EPR1 AND GRASP_EPR2

Parameters	Values for GRASP_ePR1	Values for GRASP_ePR2
$T1$	50	--
$T2$	--	20
Elite_set_size	3	3
T	6	6
Λ	20	5
P	0.1	0.3

To evaluate the impact of evoPR and path-inside LS on GRASP_ePR, we executed GRASP_ePR with a termination condition of running time limitation (50, 100, 150, 200, and/or 250 seconds). For a test instance and a fixed time limitation, GRASP_ePR are executed 30 independent runs. The average value over 30 runs is considered for evaluation. The comparison graphic is displayed in Fig. 3 for evaluation of evoPR, and Fig. 4 for evaluation of path-inside LS. According to the result in Fig. 3, having evoPR is an efficient and effective way to improve the performance of GRASP_ePR. With increase in the running time, path-inside LS improves the performance of GRASP_ePR according to Fig. 4. A conclusion rising from the abovementioned two facts is that GRASP_ePR with evoPR and path-inside LS may open a door of a promising hybridization scheme for hard optimization problems.



(a) WS500 test instance



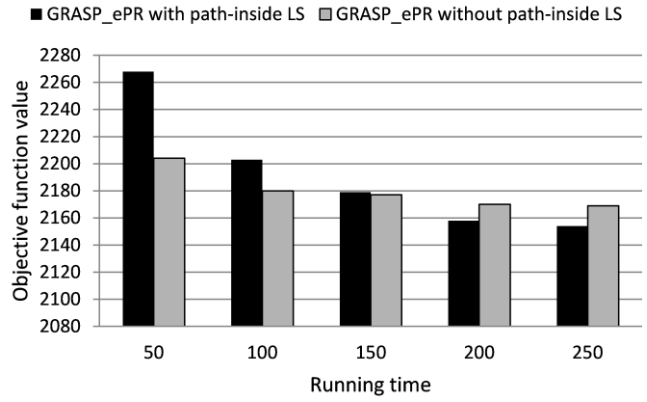
(b) ER500 test instance

Fig. 3. Comparison between GRASP_ePR with evoPR and without evoPR.

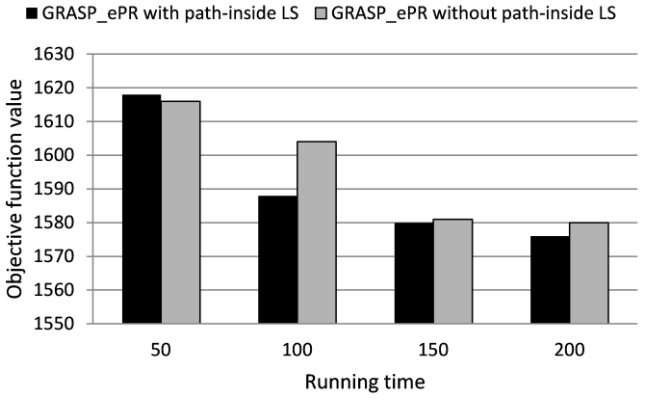
C. Experimental Results and Comparison with Existing Other Methods

Experimental results of GRASP_ePR1 and GRASP_ePR2 are summarized in Table VIII, and Table IX respectively. We show the values of best, worst, average (avg.), and

standard deviation (sd.) over 30 independent runs. GRASP_ePR2 finds the optimal solution for all instances of BA, and FF (see Table I, and Table IX). GRASP_ePR1 finds the optimal solution for most of BA and FF instances.



(a) WS500 test instance



(b) ER500 test instance

Fig. 4. Comparison between GRASP_ePR with path-inside LS and without path-inside LS;

To evaluate the performance of the proposed algorithms, we chose three recently proposed algorithms from the literature. The authors in [9] propose several heuristic algorithms for CNDP, and we chose the best one among those algorithms, namely Variable Neighborhood Search with first improvement strategy (called VNS-F). The two heuristic algorithms, Simulated Annealing (SA), and Population-based Incremental Learning (PBIL), proposed in [8] are used for performance evaluation because we use the benchmark instances defined in that work. To test these methods in a same experimental environment, we re-implemented these methods in C++ according to the guidelines of the original publications. All algorithms have been executed 30 times at each of the instances, using different random seeds for each run. The results of SA, PHIL, VNS-F, GRASP_ePR1, and GRASP_ePR2 are summarized in Table X.

According to comparison in Table X, GRASP_ePR2 outperforms other methods. More specifically, GRASP_ePR2 has large difference (improvement) on the densest graphs (WS and ER) from others.

Computational time consumed by GRASP_ePR1 and GRASP_ePR2 is compared with that consumed by VNS-F. Their computational times (average over 30 runs) are tabulated in Table XI. From Table X and Table XI, we can conclude that the proposed GRASP_ePR2 clearly outperforms the other methods in terms of solution quality. In addition, the proposed GRASP_ePR1 finds competitive results with less time than other methods.

TABLE VIII
SUMMARY RESULTS (BEST, WORST, AVERAGE (AVG.), STANDARD DEVIATION (SD.)) FOR GRASP_EPR1

Benchmark test instances	best	worst	avg.	sd.
ER250	297	300	297.8	0.9
ER500	1566	1668	1620.9	29.2
ER1000	5500	6078	5807.5	214.8
ER2500	1048464	1070207	1058018.1	6523.8
BA500	195	195	195.0	0.0
BA1000	558	558	558.0	0.0
BA2500	3704	3704	3704.0	0.0
BA5000	10196	10196	10196.0	0.0
WS250	4301	5509	4791.7	279.5
WS500	2102	2220	2173.6	42.7
WS1000	134006	137748	135877.0	1871.0
WS1500	14047	15051	14427.0	261.3
FF250	194	194	194.0	0.0
FF500	257	259	258.1	0.5
FF1000	1260	1271	1262.9	2.9
FF2000	4549	4590	4564.7	13.4

TABLE IX
SUMMARY RESULTS (BEST, WORST, AVERAGE (AVG.), STANDARD DEVIATION (SD.)) FOR GRASP_EPR2

Benchmark test instances	best	worst	avg.	sd.
ER250	295	301	297.3	1.8
ER500	1536	1635	1580.1	25.4
ER1000	5102	5696	5442.2	176.3
ER2500	1010487	1087914	1049958.1	19515.3
BA500	195	195	195.0	0.0
BA1000	558	558	558.0	0.0
BA2500	3704	3704	3704.0	0.0
BA5000	10196	10200	10197.0	1.7
WS250	3192	4797	4047.1	433.2
WS500	2099	2234	2146.4	35.7
WS1000	128657	141987	135317	3495.4
WS1500	13773	14251	13981.7	318.6
FF250	194	195	194.1	0.3
FF500	257	261	258.1	1.3
FF1000	1260	1266	1261.8	1.9
FF2000	4545	4566	4553.3	6.0

TABLE X
COMPARISON OF THE BEST OBJECTIVE FUNCTION VALUE FOR SA, PBIL, VNS-F, GRASP_EPR1, AND GRASP_EPR2.

Bench. test instances	SA	PBIL	VNS-F	GRASP_ePR1	GRASP_ePR2
ER250	7700	6700	298	297	**295** [a]
ER500	48627	44255	1542	1566	**1536**
ER1000	234479	229576	5198	5500	**5102**
ER2500	2011122	2009132	1034333	1048464	**1010487**
BA500	997	892	**195**	**195**	**195**
BA1000	3770	3057	559	**558**	**558**
BA2500	31171	28044	**3704**	**3704**	**3704**
BA5000	170998	146753	10218	**10196**	**10196**
WS250	14251	13786	6610	4301	**3192**
WS500	54201	53779	2148	2102	**2099**
WS1000	311700	308596	139653	134006	**128657**
WS1500	717369	703241	14619	14047	**13773**
FF250	1841	1386	**194**	**194**	**194**
FF500	2397	1904	**257**	**257**	**257**
FF1000	92800	59594	1263	**1260**	**1260**
FF2000	387248	256905	4549	4549	**4545**
# of best value:	0	0	4	7	16

[a] The best results are displayed in bold font.

D. New best known values

We compare our best values with the best known values for the benchmark test instances considered in this work. The best known values are collected from the study presented in [9]. The following Table XII shows their comparison.

TABLE XI
COMPARISON OF COMPUTATIONAL TIME (IN SECONDS) FOR VNS-F, GRASP_EPR1 AND GRASP_EPR2

Benchmark test instances	VNS-F	GRASP_ePR1	GRASP_ePR2
ER250	7	4	13
ER500	57	41	169
ER1000	691	355	2480
ER2500	5002	3240	5009
BA500	11	18	19
BA1000	86	72	130
BA2500	563	487	2022
BA5000	5016	5012	5006
WS250	5	8	126
WS500	566	119	502
WS1000	415	342	4219
WS1500	5051	2792	5032
FF250	8	5	8
FF500	175	26	85
FF1000	1455	192	983
FF2000	5005	1657	5012

TABLE XII
COMPARISON OF BEST VALUES OBTAINED BY GRASP_EPR1 AND GRASP_EPR2 WITH BEST KNOWN VALUES FROM THE LITERATURE.

Bench. test instances	Old best known values	GRASP _ePR1	GRASP _ePR2	New best known values	Gap %
ER250	295	297	295	295	0.00%
ER500	1542	1566	**1536**	1536	-0.39%
ER1000	5198	5500	**5102**	5102	-1.85%
ER2500	1012849	1048464	**1010487**	1010487	-0.23%
BA500	195	195	195	195	0.00%
BA1000	559	**558**	**558**	558	-0.18%
BA2500	3704	3704	3704	3704	0.00%
BA5000	10196	10196	10196	10196	0.00%
WS250	3241	4301	**3192**	3192	-1.51%
WS500	2130	2102	**2099**	2099	-1.46%
WS1000	139653	134006	**128657**	128657	-7.87%
WS1500	14138	14047	**13773**	13773	-2.58%
FF250	194	194	194	194	0.00%
FF500	257	257	257	257	0.00%
FF1000	1260	1260	1260	1260	0.00%
FF2000	4549	4549	**4545**	4545	-0.09%

[a] The new best known values obtained by GRASP_ePR1 and GRASP_ePR2 are presented in bold.

The proposed algorithms discover new best known solutions for all instances of the densest graphs (WS and ER graphs) except ER250 whose best known value is replicated by the proposed algorithms. In addition, new best known values (that is also optimal value) for BA1000, and FF2000 instances are discovered. As mentioned in previous section, our proposed GRASP_ePR2 has ability to discover optimal solutions for all instances of BA and FF graphs. It is worth to note that WS graphs are hardest test instances because they are the densest graphs in this benchmark set (average degree of nodes is up to five times higher than other graphs). In addition, WS graphs have no cut points (see Table I). If a graph has many cut points, the graph can easily be divided into connected components by removing cut points. We can therefore conclude that the proposed algorithms can find higher quality solutions for dense graphs than other methods. It is also important to note that the best known values defined in [9] are collected from 24 different variants of VNS and 6 different variants of iterated local search-based methods with 10000 seconds of running time limitation.

V. CONCLUSION

This paper proposed a new hybrid heuristic algorithm for the critical node detection problem. The heuristic uses exterior and interior path-relinking in the context of GRASP in a new way. Experimental evaluation shows that the proposed heuristic outperforms other existing methods in the literature.

As our future work, this proposed hybridization scheme would be used for other combinatorial optimization problems, and therefore, its application for other problems is an interesting future research. By improving the method for defining transition probabilities in the construction phase, more high quality outcomes may be created. In general sense, methods for the community detection problem such as methods in [18], [19] that is a network related problem, may be successful in constructing solution for GRASP.

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, pp. 2193–2200, Jul. 2009.

[2] R. Cohen, S. Havlin, and D. ben-Avraham, "Efficient Immunization Strategies for Computer Networks and Populations," *Physical Review Letters*, vol. 91, no. 24, p. 247901, Dec. 2003.

[3] Krebs, V. Uncloaking terrorist networks. First Monday 2002, 7.

[4] F. Glover, "Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges," in Interfaces in Computer Science and Operations Research, R. S. Barr, R. V. Helgason, and J. L. Kennington, Eds. Springer US, 1997, pp. 1–75.

[5] F. Glover, "Exterior Path Relinking for Zero-One Optimization," *International Journal of Applied Metaheuristic Computing*, vol. 5, no. 3, pp. 1–8, Jul. 2014.

[6] M. Laguna and R. Marti, "GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44–52, Feb. 1999.

[7] A. Duarte, J. Sánchez-Oro, M. G. C. Resende, F. Glover, and R. Martí, "Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization," *Information Sciences*, vol. 296, pp. 46–60, Mar. 2015.

[8] 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, pp. 2763–2775, Nov. 2012.

[9] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia, "Local search metaheuristics for the critical node problem," *NETWORKS*, vol. 67, no. 3, pp. 209–221, May 2016.

[10] 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, pp. 2349–2360, Nov. 2013.

[11] M. Di Summa, A. Grosso, and M. Locatelli, "Complexity of the critical node problem over trees," *Computers & Operations Research*, vol. 38, no. 12, pp. 1766–1774, Dec. 2011.

[12] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, Apr. 1989.

[13] T. A. Feo and M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, Mar. 1995.

[14] M. G. C. Resende and C. C. Ribeiro, "GRASP: Greedy Randomized Adaptive Search Procedures," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Springer US, 2014, pp. 287–312.

[15] C. C. Ribeiro and M. G. C. Resende, "Path-relinking intensification methods for stochastic local search algorithms," *Journal of Heuristics*, vol. 18, no. 2, pp. 193–214, Apr. 2012.
[16] P. Sarkar and A. W. Moore, "Random Walks in Social Networks and their Applications: A Survey," in *Social Network Data Analytics*, C. C. Aggarwal, Ed. Springer US, 2011, pp. 43–77.
[17] M. Laguna and R. Marti, *Scatter Search - Methodology and Implementations in C*. Springer US, 2003.
[18] A. Song, M. Li, X. Ding, W. Cao, and K. Pu, "Community Detection Using Discrete Bat Algorithm," *IAENG International Journal of Computer Science*, vol. 43, no. 1, pp. 37–43, 2016.
[19] X. Bai, P. Yang, and X. Shi, "An overlapping community detection algorithm based on density peaks," *Neurocomputing*, vol. 226, pp. 7–15, Feb. 2017.