Graph Fragmentation Problem

Juan Piccini, Franco Robledo and Pablo Romero
Laboratorio de Probabilidad y Estadística, Instituto de Matemática y Estadística, Prof. Ing. Rafael Laguardia (Imerl), Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, Montevideo, Uruguay

Keywords: Graph Theory, Combinatorial Optimization Problem, Metaheuristics, GRASP, Path Relinking.

Abstract: A combinatorial optimization problem called Graph Fragmentation Problem (GFP) is introduced. The decision variable is a set of protected nodes, which are deleted from the graph. An attacker picks a non-protected node uniformly at random from the resulting subgraph, and it completely affects the corresponding connected component. The goal is to minimize the expected number of affected nodes S. The GFP finds applications in fire fighting, epidemiology and robust network design among others. A Greedy notion for the GFP is presented. Then, we develop a GRASP heuristic enriched with a Path-Relinking post-optimization phase. Both heuristics are compared on the lights of graphs inspired by a real-world application.

1 INTRODUCTION

In robust network design, the major cause of concern is connectivity. The goal is to find a minimum cost design, meeting high connectivity requirements. Network connectivity is a rich field of knowledge, and the related literature is vast (Monma et al., 1990; Stoer, 1993). However, in several real-world applications a malfunctioning or affection of a single element is immediately propagated to neighboring elements. This is the case of fire fighting, electric shocks, epidemic propagations, etc., where an incorrect protection scheme might have catastrophic effects. In this paper, an abstract setting of the previous problems is presented as a combinatorial optimization problem. The reader can find problems related with graph partitioning, which are similar in nature to ours in (Borgatti, 2006), (Ortiz-Arroyo, 2010). In (Borgatti, 2006), the author studies a combinatorial problem inspired in game theory, where key players are protected (deleted) in order to cope with network attackers. Several scores are proposed in order to capture a notion of network resilience. In (Ortiz-Arroyo, 2010), an entropy-based score is considered for network resilience. In this article we consider a score which is slightly different to that of Borgatti. Additionally, we mathematically prove sufficient conditions for a solution to be optimal. These properties are then used as part of a GRASP heuristic. The document is organized in the following manner. Section 2 formally presents the Graph Fragmentation Problem. Desired properties of candidate solutions are included in Section 3. A Greedy notion and a more sophisticated heuristic for the GFP is developed in Section 4. Section 5 shows the results of both heuristics in a real-world application, while Section 6 presents concluding remarks.

2 GRAPH FRAGMENTATION PROBLEM

We are given a simple graph $G = (V,E)$ and a budget constraint B. The decision variable is a subset V^*, called *protected nodes*, which will be deleted from the graph. The result is an induced subgraph $G' = (V', E(V'))$, with $V' = V - V^*$. A node $v \in V'$ is uniforlmy chosen at random, and the full component from G' that contains v is affected, this is, damaged by an atacker or an accident that starts at v.

The goal is to choose the set $V^* : |V^*| \leq B$ in order to minimize the expected value (or *Score*) of affected nodes. If the resulting graph G' is partitioned into k connected components with orders $n_1, \ldots, n_k$ such that $n = |V'|$, then the Graph Fragmentation Problem (GFP) is the following combinatorial optimization problem:

$$\min_{V^*} Sc(G') = \sum_{i=1}^{k} p_i n_i, \quad (1)$$

$$s.t. |V^*| \leq B, \quad (2)$$

being $p_i = \frac{n_i}{n}$ the probability of the event $v \in V_i$, being v the node uniformly chosen at random.

Piccini, J., Robledo, F. and Romero, P.
Graph Fragmentation Problem.
DOI: 10.5220/0005697701370144
In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 137-144
ISBN: 978-989-758-171-7

3 PROPERTIES

In this section, we study properties of the FGP. Observe that the best case occurs when only a singleton is affected, so $Sc(G') \geq 1$. The equality is achieved if and only if G' consists of isolated nodes. Furthermore, if n_{max} denotes the number of nodes from the largest component, then $Sc(G') \leq n_{max}$ by its definition.

Recall that the union between graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$, and it is denoted $G = G_1 \cup G_2$. A counterintuitive result is that a uniformly random node protection strategy might lead to a worse solution. In fact, let us consider $G = K_2 \cup K_4$, being K_n the complete graph with n nodes. The score is $Sc(G) = \frac{2}{6} \times 2 + \frac{4}{6} \times 4 = \frac{10}{3}$. However, if we pick $v \in K_2$ then $Sc(G - v) = \frac{1}{5} \times 1 + \frac{4}{5} \times 4 = \frac{17}{5}$, so $Sc(G - v) > Sc(G)$.

The intuition suggests that it is better to disconnect the graph whenever possible, this is, to protect nodes in such a way that the resulting subgraphs has as many components as possible. Then, only few nodes will be affected. This property explains the name of the problem. These results are mathematically formalized in the following paragraphs.

Proposition 1 (Load Balancing)**.** *The best resulting graph G' among all feasible graphs with k components and identical order n should have balanced components: $n_i = n/k, \forall i = 1, \ldots, k$.*

Proof. The score is precisely $Sc(G') = \frac{\|x\|_2^2}{\|x\|_1}$, being $x = (n_1, \ldots, n_k)$, $\|x\|_2$ and $\|x\|_1$ the respective Euclidean and 1-norm for vector x. We should minimize the Euclidean distance in the hyperplane $\|x\|_1 = n$ constant, whose normal vector is $\overrightarrow{1} \in \mathbb{R}^k$, with all unit coordinates. The optimum is found in the orthogonal projection of the null vector onto the polyhedra: $x_{opt} = \overrightarrow{0} + \frac{n}{\|\overrightarrow{1}\|_2} \frac{\overrightarrow{1}}{\|\overrightarrow{1}\|_2} = \overrightarrow{1}\frac{n}{k}$. □

Now, let us determine whether it is better to protect an additional node. Let G be an arbitrary graph with k components and cardinalities $n_1, \ldots, n_k$, and $n = \sum_{j=1}^{k} n_i$. If we delete some node v from the first component (observe that the labels are not relevant for analysis), there are two cases:

a) The number of connected components is the same.

b) The number of connected components is increased.

First, assume that Condition [a] holds. Then:

$$Sc(G - v) - Sc(G) = \sum_{i=2}^{k} \frac{(n_i)^2}{n} \frac{1}{n-1} + \frac{(n_1 - 1)^2}{n-1} - \frac{(n_1)^2}{n}$$
$$= \frac{A}{n-1} + B_v - B,$$

being $A = 1/n \sum_{i=2}^{k} (n_i)^2$, $B_v = (n_1 - 1)^2/(n-1)$ and $B = n_1^2/n$. As a consequence, $Sc(G - v) < Sc(G)$ if and only if n_1 meets the following inequality:

$$P(n_1) = (n_1)^2 - 2nn_1 + (1 + A)n < 0 \qquad (3)$$

Observe that the minimum (or the highest score reduction) is achieved when $n_1 = n$, this is, when G is connected. In that case $Sc(G - v) - Sc(G) = 1$. We have proved the following

Proposition 2 (Best Singleton)**.** *If there is no cut-node, the best node protection belongs to the highest connected component.*

Proof. The polynomial P is monotonically decreasing with respect to n_1. □

Let n_{max} be the size of the highest connected component. Studying the sign of $P(n_{max})$, there is a positive score reduction if and only if:

$$n_{max} \geq n - \sqrt{n(n - 1 - A)}. \qquad (4)$$

We will see that this inequality always holds:

Proposition 3 (Single Balancing)**.** *If G does not present a cut-node, then there exists v such that $Sc(G - v) < Sc(G)$.*

Proof. Inequality 4 occurs if and only if $nA + n_{max}^2 \leq n(2n_{max} - 1)$. But $nA + n_{max}^2 = \sum_i n_i^2 = nSc(G)$, so Inequality 4 holds if and only if $Sc(G) \leq 2n_{max} - 1$. But $Sc(G) \leq n_{max} \leq 2n_{max} - 1$ always holds. □

Let us now focus our study to Condition [b], and denote by v a cut-node in G. First of all, observe that a node-protection in a balanced way always produces a score reduction. However, in some cases, the deletion of a cut node is not a good idea. Consider for instance $G = C_9 \cup P_3$, being C_9 an cycle with 9 nodes and P_3 an elementary path with 3 nodes, and v the central node from P_3. Then, we have that $Sc(G) = \frac{9^2}{12} + \frac{3^2}{12} = \frac{90}{12} = \frac{15}{2}$, but $Sc(G - v) = \frac{9^2}{11} + 2 \times \frac{1^2}{11} = \frac{83}{11} = 7 + \frac{6}{11}$, so $Sc(G - v) > Sc(G)$. However, if we choose a cut-node v from a component with n_j nodes, such that $P(n_j) < 0$, then the score is decreased. Furthermore, the score reduction is even better than in the case of no cut-node.

Proposition 4 (Fragmentation)**.** *If G presents a cut-node $v \in V_j$ where $|V_j| = n_j$ and $P(n_j) < 0$, then $Sc(G - v) < Sc(G)$. Furthermore, if $v' \in V$ is not a cut-node, then $Sc(G - v) < Sc(G - v') < Sc(G)$.*

Proof. If $V_j - \{v\} = V_a \cup V_b$, $n_a + n_b = n_j - 1$, then $(n_a)^2 + (n_b)^2 < (n_j - 1)^2$. This implies that the score reduction is even larger than in a non cut-node deletion. A similar argument is met when the cut node produces more than two components. As a consequence, the score reduction is even larger than the protection of a non-cut-node from V_j. By Proposition 3, a score reduction is achieved if v' is not a cut-node. □

Theorem 1 (Score Reduction)**.** *Consider an arbitrary graph $G = (V,E)$. There is some $v \in V$ such that $Sc(G - v) < Sc(G)$, unless G consists of isolated nodes.*

Proof. This is a Corollary of Single Balancing and Fragmentation. We always pick a node v from the largest connected component with n_{max} nodes. If v is not a cut node, by Proposition 4 we have $Sc(G - v) < Sc(G)$. Otherwise, by Proposition 3 the score reduction is even larger, so $Sc(G - v) < Sc(G)$ again. In both cases a score reduction is produced. □

4 HEURISTICS

Combinatorial optimization problems arise in several real-world problems (economics, telecommunication, transport, politics, industry), were human beings have the opportunity to choose among several options. Usually, that number of options cannot be exhaustively analyzed, mainly because its number increases exponentially with an input size of the system. Much work has been done over the last six decades to develop optimal seeking methods that do not explicitly require an examination of each alternative, giving shape to the field of *Combinatorial Optimization* (Papadimitriou and Steiglitz, 1982). Several combinatorial problems belong to the $\mathcal{NP}$-Hard class, or the search space is sufficiently large to admit an exact algorithm, and a smart search technique should be considered exploiting the real structure of the problem via heuristics. Optimality is not guaranteed, but compromised at the cost of computational efficiency. Metaheuristics are an abstraction of search methodologies which are widely applicable to optimization problems. The most promising are Simulated Annealing (Kirkpatrick, 1984), Tabu Search (Glover, 1989), Genetic Algorithms (Goldberg, 1989), Variable Neighborhood Search (Hansen and Mladenovic, 2001), GRASP (Feo and Resende, 1989), Ant Colony Optimization (Dorigo, 1992) and Particle Swarm Optimization (Kennedy and Eberhart, 1995), among others. The interested reader can find a list of metaheuristics and their details in the Handbook of Metaheuristics (Gendreau and Potvin, 2010).

In this section, we develop a Greedy notion and a Grasp heuristic enriched with a Path Reliking post-optimization stage. First, we review basic elements of Grasp and Path Relinking.

4.1 GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start or iterative process (Lin and Kernighan, 1973), where feasible solutions are produced in a first phase, and neighbor solutions are explored in a second phase. The best overall solution is returned as the result. The first implementation is due to Tomas Feo and Mauricio Resende, were the authors address a hard set covering problem arising for Steiner triple systems (Feo and Resende, 1989). They introduce adaptation and randomness to the classical Greedy heuristic for the set covering problem (where $P_1, \ldots, P_n$ cover the set $J = \{1, \ldots, m\}$ and the objective is to find the minimum cardinality set $I \subset \{1, \ldots, n\}$ such that $\cup_{i \in I} P_i = J$).

It is a powerful metaheuristic to address hard combinatorial optimization problems, and has been succesfully implemented in particular to several telecommunications problems, such as Internet Telephony (Srinivasan et al., 2000), Cellular Systems (Amaldi et al., 2003a; Amaldi et al., 2003b), Cooperative Systems (Romero, 2012), Connectivity (Canuto et al., 2001) and Wide Area Network design (Robledo Amoza, 2005). Here we will sketch the GRASP metaheuristic based on the work from Mauricio Resende and Celso Ribeiro, which is useful as a template to solve a wide family of combinatorial problems (Resende and Ribeiro, 2003; Resende and Ribeiro, 2014). Consider a ground set $E = \{1, \ldots, n\}$, a feasible set $F \subseteq 2^E$ for the optimization problem $\min_{A \subseteq E} f(A)$, and an objective function $f : 2^E \to \mathbb{R}$. The Pseudo-code 1 illustrates the main blocks of a GRASP procedure for minimization, where *Max_Iterations* iterations are performed, $\alpha \in [0,1]$ is the quantity of randomness in the process and $\mathcal{N}$ is a neighborhood structure of solutions (basically, a rule that defines a neighbor of a certain solution). The cycle includes Lines $1-5$, and the best solution encountered during the cycle is finally returned in Line 6. Lines 2 and 3 represent respectively the Construction and Local Search phases, whereas the partially best solution is updated in Line 4.

A general approach for the Greedy Randomized Construction is specified in Pseudo-code 2. Solution S is empty at the beginning, in Line 1, and an auxiliary set C has the potential elements to be added to S. A

Algorithm 1: $S = GRASP(MaxIterations, \mathcal{N})$.

```
1: for $k = 1$ to $Max\_Iterations$ do
2:     $S \leftarrow Greedy\_Randomized(\alpha)$
3:     $S \leftarrow Local\_Search(S, \mathcal{N})$
4:     $Update\_Solution(S)$
5: end for
6: return $S$
```

carefully chosen element from C is picked up during each iteration of the While loop (Lines $3-9$), which is finished once a feasible solution is met. A Greedy construction would choose c^{min}, which is the element with the lowest cost to be added to the partial non-feasible solution (Line 4). On the other hand, c^{max} is the most expensive element to be added (Line 5). The *Restricted Candidate List* RCL is defined in Line 6, and has all the elements whose cost are below a certain threshold (see Line 6). In Line 7, an element from the RCL is uniformly picked at random and added to the solution S. The process is repeated until a feasible solution S is found. It is worth to notice the effect of the input parameter $\alpha \in [0,1]$. When $\alpha = 0$, the Greedy construction is retrieved. On the contrary, $\alpha = 1$ means a completely random construction. Therefore, the parameter α imposes a trade-off between diversification and greediness.

Algorithm 2: $S = Greedy\_Randomized(\alpha)$.

```
1: $S \leftarrow \emptyset$
2: $C \leftarrow E$
3: while $C \neq \emptyset$ do
4:     $c^{min} \leftarrow \min_{c \in C} f(S \cup \{c\})$
5:     $c^{max} \leftarrow \max_{c \in C} f(S \cup \{c\})$
6:     $RCL \leftarrow \{c \in C : f(S \cup \{c\}) \leq f(S \cup \{c^{min}\}) + \alpha(f(S \cup \{c^{max}\}) - f(S \cup \{c^{min}\}))\}$
7:     $S \leftarrow S \cup Random(RCL)$
8:     $Update(C)$
9: end while
10: return $S$
```

The Greedy Randomized Construction does not provide guarantee of local optimality. For that reason, a Local Search phase is finally introduced, in order to return a locally optimal solution (which could be incidentally globally optimal). In order to define this phase, a rule to define neighbors of a certain solution is mandatory, called a *neighborhood structure*. A better neighbor solution is iteratively picked until no improvement is possible. A general local search phase is presented in pseudo-code 3.

The success of the local search phase strongly depends on the quality of the starting solution, the computational cost for finding a better local solution, and naturally, on the richness of the neighborhood structure. The interested reader can find valuable literature and GRASP enhancements.

Algorithm 3: $S = Local\_Search(S, \mathcal{N})$.

```
1: $H(S) = \{X \in \mathcal{N}(S) : f(X) < f(S)\}$
2: while $H(S) \neq \emptyset$ do
3:     $S \leftarrow ChooseIn(H)$
4:     $H(S) = \{X \in \mathcal{N}(S) : f(X) < f(S)\}$
5: end while
6: return $S$
```

4.2 Greedy for the GFP

Usually, once we face a new combinatorial optimization problem, a Greedy notion is developed. In specific combinatorial structures, Greedy produces the globally optimum solution. Greedy heuristic builds a solution in a stepwise manner. The best step is chosen whenever possible. Therefore, Greedy tries to build the global optimum by means of the best local steps. Naturally, Greedy rarely produces the best solution (see for instance its performance in the celebrated Traveling Salesman Problem).

In our problem, Greedy iteratively applies the best node protection. Function *ChooseBestNode* finds v such that $v = arg\min_w\{Sc(G-w)\}$. Greedy is supported by Theorem 1, and the score reduction is guaranteed for the GFP.

Algorithm 4: $G_{out} = Greedy(G, B)$.

```
1: for $i = 1 : B$ do
2:     $v \leftarrow ChooseBestNode(G)$
3:     $G \leftarrow G - v$
4: end for
5: $G_{out} \leftarrow G$
6: return $G_{out}$
```

A linear search among all nodes $w \in V$ is developed in order to find the best node protection in Greedy. Observe that if there is no cut node, a node is picked uniformly at random from the largest connected component, since they produce the same score reduction. In order to trade computational effort, we propose an alternative algorithm that always improves the score. It is supported by Proposition 3.

Algorithm 5: $G_{out} = Balance(G, B)$.

```
1: for $i = 1 : B$ do
2:     $V_{max} \leftarrow LargestComponent(G_{out})$
3:     $v \leftarrow ChooseRandom(V)$
4:     $G \leftarrow G - \{v\}$
5: end for
6: $G_{out} \leftarrow G$
7: return $G_{out}$
```

Balance iteratively picks nodes from the largest connected component. Observe that no score evaluation is required, hence, the computational effort is below that of *Greedy*.

4.3 Grasp for the GFP

We already have a Greedy notion for the GFP, and a Balance heuristic. Both reduce the score in each iteration. A key point is to note that a fragmentation in large components always improves the score. Therefore, *Separator* function finds the node-connectivity for a given connected graph. It returns a node separator set V_{aux} of the largest component V_{max}.

Algorithm 6: $G_{out} = Grasp(G, B, \alpha)$.

```
1: $G_{out} \leftarrow G$
2: $Counter \leftarrow B$
3: $LocalImprove \leftarrow True$
4: while $Counter > 0$ do
5:     if $random < \alpha$ then
6:         $v = RandomNode(G)$
7:         $G_{out} \leftarrow G_{out} - v$
8:     else
9:         $V_{max} \leftarrow LargestComponent(G_{out})$
10:        $V_{aux} \leftarrow Separator(G, V_{max})$
11:        if $Counter \geq |V_{aux}|$ then
12:            $G_{out} \leftarrow G_{out} - V_{aux}$
13:            $Counter \leftarrow Counter - |V_{aux}|$
14:        end if
15:    end if
16: end while
17: while $Improve(G) = True$ do
18:     $(G_{out}, LocalImprove) \leftarrow Swap(G_{out}, G)$
19: end while
20: return $G_{out}$
```

The construction phase is first applied (Lines 1-15) and then, a Local Search phase takes place (Lines 16-18). The graph (G_{out}) and number of remaining nodes to protect (*Counter*) are initialized in Lines 1-2. Nodes are protected in a While loop (Lines 3-15). If a uniform random variable over the compact set $[0,1]$ is greater than the input α (Line 4), a random node v is then picked and removed (Lines 5-6). Otherwise, the largest component is selected (Line 8), and the node separator in that component is found (Line 9). If it is feasible, that node separator is removed from the graph (Lines 10-12). Finally, a Local Search phase takes place. It guarantees a local optimum solution. The core is *Swap* function, explained in the following lines.

Non-protected nodes are those from the first argument G_1 (Line 1), while the remaining nodes belong to the difference $V(G_2) - V(G_1)$ (see Line 2). A Boolean constant determines whether there exists some improvement or not (Line 3). Iteratively, all (protected,non-protected) pairs are considered and switched to see whether there is some improvement or not (block of Lines 5-15). If there is some improvement, *LocalImprovement* is set to True (Line 11) and iterative process is finished (Line 12). The pair $(G_{out}, Improve)$ is returned. It is worth to remark that $G_{out} = G_1$ if and only if G_1 is a local optimum. Otherwise, the best first movement is produced.

Algorithm 7: $(G_{out}, Improve) = Swap(G_1, G_2)$.

```
1: $\{v_1, \ldots, v_{n-B}\} \leftarrow V(G_1)$
2: $\{v_{n-B+1}, \ldots, v_n\} \leftarrow V(G_2) - V(G_1)$
3: $Improve \leftarrow False$
4: $G_{out} \leftarrow G_1$
5: for $i = 1 : B$ do
6:     for $j = 1 : n - B$ do
7:         $G_{aux} \leftarrow (G_{out} + v_{n-B+i}) - v_j$
8:         $reduction \leftarrow E(G) - E(G_{aux})$
9:         if $reduction > 0$ then
10:            $G_{out} \leftarrow G_{aux}$
11:            $Improve \leftarrow True$
12:            break
13:        end if
14:    end for
15: end for
16: return $(G_{out}, Improve)$
```

4.4 Path Relinking

Thanks to the randomization introduced to the Grasp heuristic, new solutions are obtained with different runs. Then, once we consider a pool $\{G_1, \ldots, G_s\}$ of s elite solutions (they are the best solutions obtained using $N >> s$ runs), new solutions could be found via elementary paths in the graph $\mathcal{G}$ of solutions. In this case, the node set of $\mathcal{G}$ is the induced subgraph of G with precisely $n - B$ nodes. Two solutions G_1 and G_2 are incident if and only if there is a single swap that moves one solution into the other (i.e., if they differ in one node).

Algorithm 8: $Pool = Relinking(G_1, G_2, \ldots, G_r)$.

```
1: $S \leftarrow (G_1, G_2, \ldots, G_r)$
2: for all $(u, v) \in Pool$ do
3:     $Path \leftarrow ShortestWalk(u, v)$
4:     $S_{u,v} \leftarrow Best(Path)$
5:     $S \leftarrow S \cup \{S_{u,v}\}$
6: end for
7: $Pool \leftarrow SelectBest(r, S)$
8: return $Pool$
```

Relinking receives a pool of r solutions and returns another pool of r solutions, with better score.New candidate solutions $S_{u,v}$ are found for every pair of elite solutions u and v. The best r solutions are returned.

4.5 Main Algorithm

The main algorithm combines Grasp strength and a Path relinking post-optimization stage, in a straightforward fashion.

Algorithm 9: $G_{out} = Main(G, B, N_1, N_2)$.

```
1: $S \leftarrow \emptyset$
2: for $i = 1 : N_1$ do
3:     $G_i \leftarrow Grasp(G, B)$
4:     $S \leftarrow S \cup G_i$
5: end for
6: $(G_1, \ldots, G_r) \leftarrow Best(r, S)$
7: for $i = 1 : N_2$ do
8:     $(G_1, \ldots, G_r) \leftarrow Relinking(G_1, \ldots, G_r)$
9: end for
10: $G_{out} \leftarrow SelectBest(1, \{G_1, G_2, \ldots, G_r\})$
11: return $G_{out}$
```

5 RESULTS

In order to highlight the effectiveness of our three heuristics, we introduce *Greedy*, *Balance* and *Main* to three real-life graphs. These graphs G_{USA}, G_{FON} and G_{PEG} represent respectively the neighborhood of the states from USA, a real Fiber Optic Network and a part of a real Power Electric Grid. In all cases, it is highly desirable to minimize the risk of the neighboring elements, once a failure or catastrophic event occurs. Thanks to the randomization effect during *Balance* call, the performance of different runs is variable. Figures 1, 2 and 3 show the score of *Greedy* (solid line) and the scores of 30-runs of *Balance* (dashed lines) versus *Main* with $N_1 = N_2 = 30$, $r = 6$, $\alpha = 0.5$. The score for the different heuristics is expressed as a function of the budget B. Red point's abcissa is the cost reached by *Main* after removing 20 nodes. Runs were made on a computer Dell Inspiron-N4010 with 1.8 GiB of memory, proccesor Intel Core i3 CPU M380 @ 2.53 GHz x 4, 64 bit's OS. CPU times are 0.924 min. for G_{USA}, 1.065 min. for G_{FON} and 13.021 min. for G_{PEG}.

It can be appreciated that our *Main* heuristic outperforms both naive solutions *Greedy* and *Balance*, under all possible budgets. Even though *Balance* has a reduced computational cost, its performance presents a large gap with respect to *Greedy* heuristic. Figures 4, 5 and 6 show the pruning result for the different heuristics and graphs under study.

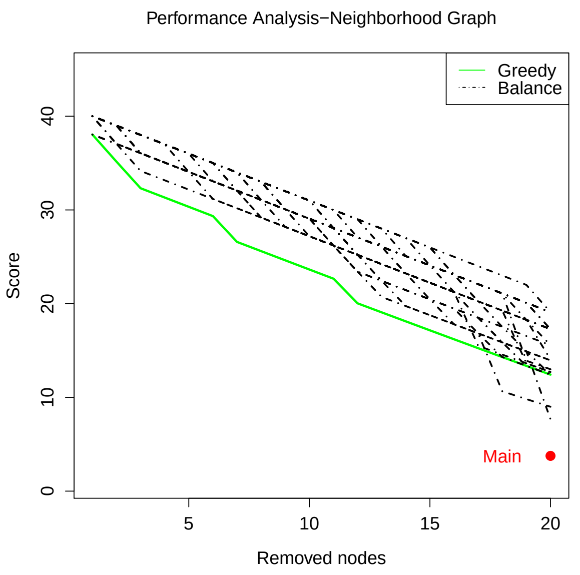

Figure 1: Greedy (solid) vs. 30 Balance runs (dashed) for the Neighborhood Graph.

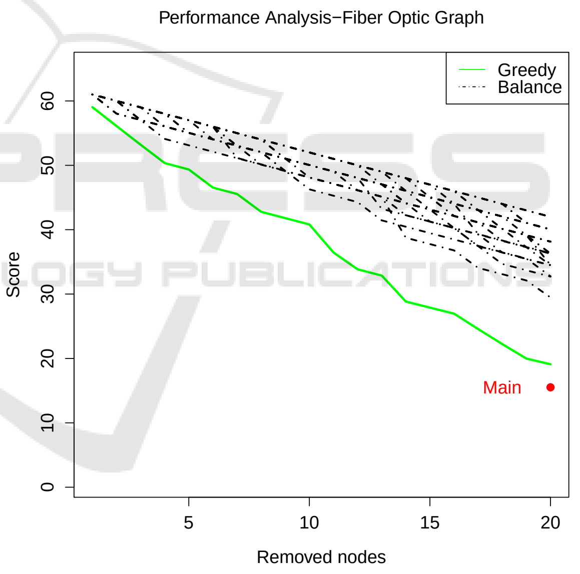

Figure 2: Greedy (solid) vs. 30 Balance runs (dashed) for the Fiber Optic Graph.

6 CONCLUSIONS

The Graph Fragmentation Problem (GFP) has been introduced. The goal is to protect (remove) B nodes from a graph G, in such a way that a random attack to an arbitrary node v affects the lowest expected number of nodes (where the whole connected component from v is affected). The GFP finds applications to fire fighting, highly virulent epidemic propagations and electric shocks, among others.

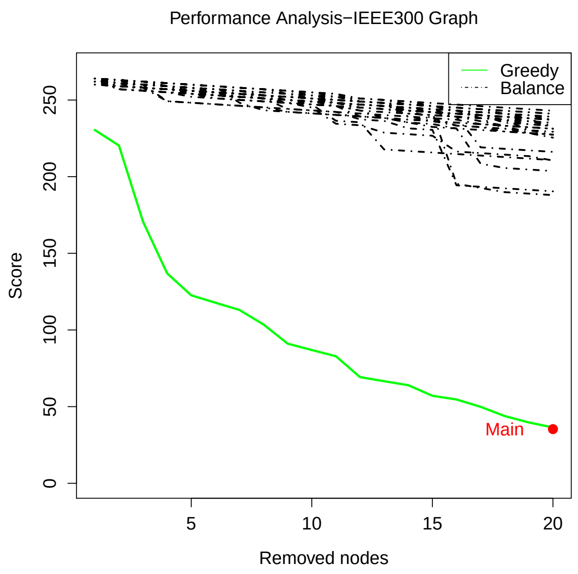

Figure 3: Greedy (solid) vs. 30 Balance runs (dashed) for the IEEE 300 Graph.

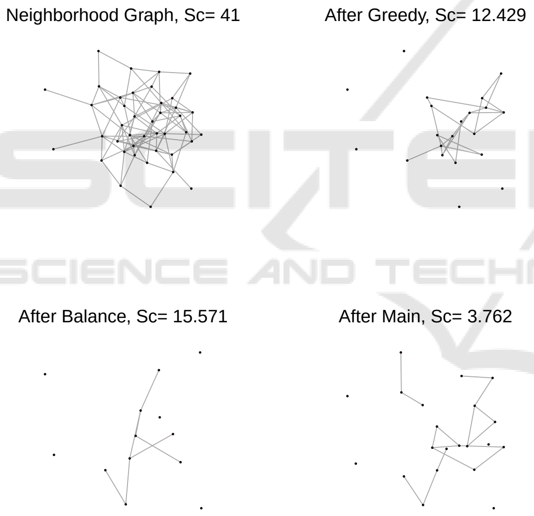

Figure 4: Graph G_{USA} when $\alpha = 1/2$.

In this paper, elementary properties of the GFP were studied. Specifically, graph fragmentation and balancing are good strategies. Together, they define a Greedy notion for the problem. Furthermore, we proved that Greedy achieves improvement in each iteration (i.e., in each node protection). A more sophisticated Grasp heuristic enriched with a Path Relinking post-optimization scheme has been developed. The effectiveness of our more sophisticated heuristic has been tested on a real-life networks.

As a future work, we would like to establish the intractability of GFP and develop different heuristics.

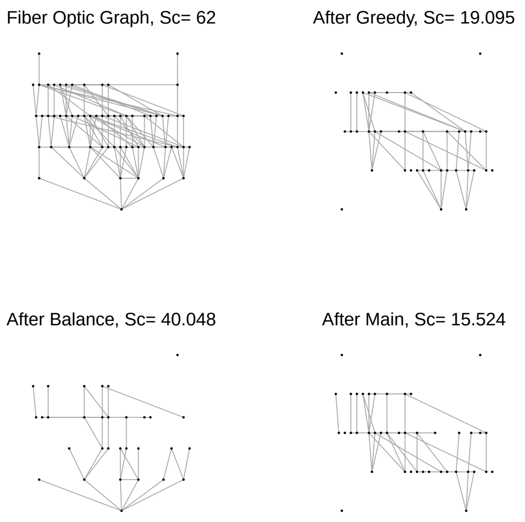

Figure 5: Graph G_{FON} when $\alpha = 1/2$.

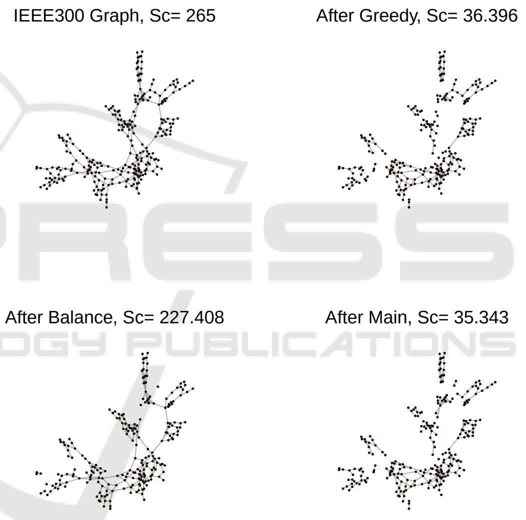

Figure 6: Graph G_{PEG} when $\alpha = 1/2$.

REFERENCES

Amaldi, E., Capone, A., and Malucelli, F. (2003a). Planning UMTS base station location: Optimization models with power control and algorithms. *IEEE Transactions on Wireless Communications*, 2(5):939–952.

Amaldi, E., Capone, A., Malucelli, F., and Signori, F. (2003b). Optimization models and algorithms for downlink UMTS radio planning. In *Wireless Communications and Networking, (WCNC'03)*, volume 2, pages 827–831.

Borgatti, S. (2006). Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, 12:21–34.

Canuto, S., Resende, M., and Ribeiro, C. (2001). Local search with perturbation for the prize-collecting Steiner tree problems in graphs. *Networks*, 38:50–58.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, DEI Politecnico de Milano, Italia.

Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition.

Glover, F. (1989). Tabu search - part i. *INFORMS Journal on Computing*, 1(3):190–206.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Hansen, P. and Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5):975–986.

Lin, S. and Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516.

Monma, C. L., Munson, B. S., and Pulleyblank, W. R. (1990). Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(1-3):153–171.

Ortiz-Arroyo, D. (2010). *Discovering sets of key players in social networks, in Computational Social Network Analysis, Chp. 2*. Springer London.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Resende, M. G. C. and Ribeiro, C. C. (2003). *Greedy Randomized Adaptive Search Procedures*. In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers.

Resende, M. G. C. and Ribeiro, C. C. (2014). Grasp: Greedy randomized adaptive search procedures. In Burke, E. K. and Kendall, G., editors, *Search Methodologies*, pages 287–312. Springer US.

Robledo Amoza, F. (2005). *GRASP heuristics for Wide Area Network design*. PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France.

Romero, P. (2012). *Mathematical analysis of scheduling policies in Peer-to-Peer video streaming networks*. PhD thesis, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay.

Srinivasan, A., Ramakrishnan, K., Kumaram, K., Aravamudam, M., and Naqvi, S. (2000). Optimal design of signaling networks for Internet telephony. In *Proceedgins of the IEEE INFOCOM 2000*.

Stoer, M. (1993). *Design of Survivable Networks*. Lecture Notes in Mathematics. Springer.