Exact Algorithms and Heuristics for the Perfect Awareness Problem

Felipe de C. Pereira, Pedro J. de Rezende and Cid C. de Souza
Institute of Computing, University of Campinas, Campinas, Brazil
felipe.pereira@students.ic.unicamp.br, {rezende,cid}@ic.unicamp.br

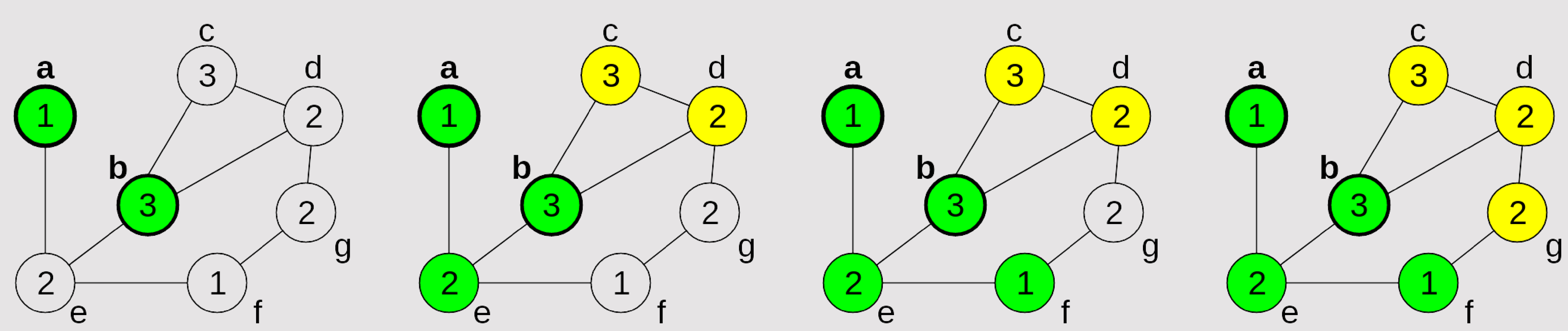
Keywords: Perfect Awreness Problem, Spreading of information, Active influence, Metaheuristic, Integer Programming.
This work was supported in part by Grants from CNPq, Brazil: #313329/2020-6, #130838/2019-5, #309627/2017-6; and FAPESP, Brazil: #2020/09691-0, #2019/22297-1, #2018/26434-0, #2014/12236-1.

The Perfect Awareness Problem (PAP)

The PAP is an NP-hard problem related to the spreading of information in social networks [1]. Let $G = (V, E)$ be a graph that represents a network. For each round $\tau \geq 0$, every $v \in V$ is either *ignorant*, (merely) *aware* or *spreader* of a given information.

In round $\tau = 0$, all vertices are ignorant, except for a set of spreaders $S \subseteq V$ called a *seed set*, whose elements are the *seeds*. Dissemination goes as follows: a vertex v is aware in round $\tau \geq 1$, if there is one spreader neighbor of v in round $\tau - 1$. Considering a *threshold function* $t : V \rightarrow N^{+}$, v is a spreader in round $\tau \geq 1$, if v have at least $t(v)$ spreader neighbors in round $\tau - 1$. When dissemination stops (i.e., no vertex becomes a new spreader) with all vertices being aware, S is called a *perfect seed set*.

Let $\{G, t\}$ be an instance of PAP. The objective of the problem is to find a perfect seed set of minimum size. In Figure 1, the ignorant, aware and spreader vertices are in gray, yellow and green, respectively. The thresholds are indicated inside the circles.



(a) $\tau = 0$ (b) $\tau = 1$ (c) $\tau = 2$ (d) $\tau = 3$

Figure 1: Example of the propragation process of PAP.

Techniques for Preprocessing Instances

(I) Solve the PAP for each connected component of G separately;

(II) Contract any edge $\{u, v\}$, with $t(u) = t(v) = 1$;

(III) Collapse u and v into one vertex, whenever $t(v) = 1$ and v is u's only neighbor.

Integer Programming

Let $n = |V|$. Define a binary variable $s_{v,\tau}$ to be 1 iff vertex v is a spreader at round τ. Our integer programming formulation for PAP reads:

$$\min z = \sum_{v \in V} s_{v,0} \tag{1}$$

$$\sum_{u \in N(v)} s_{u,\tau-1} - t(v)(s_{v,\tau} - s_{v,0}) \geq 0 \qquad \forall v \in V\ \forall \tau \in [1, n] \tag{2}$$

$$s_{v,0} + \sum_{u \in N(v)} s_{u,n-1} \geq 1 \qquad \forall v \in V \tag{3}$$

$$s_{v,\tau} \in \{0, 1\} \qquad \forall v \in V\ \forall \tau \in [0, n] \tag{4}$$

The objective function (1) minimizes the size of the seed set. Constraints (2) ensure that v is a spreader in round τ, only if either v belongs to the seed set or the number of its neighboring spreaders in round $\tau - 1$ is greater than or equal to $t(v)$.

Since $|V| = n$, a full propagation takes at most $n + 1$ rounds to end. Hence, constraints (3) enforce that either v is a seed or has at least one neighboring spreader in round $\tau = n - 1$, which means that v is necessarily aware in round n.

Metaheuristic GRASP

GRASP [2] is a metaheuristic comprised of iterations with two distinct phases. Firstly, a feasible solution S is built by inserting elements from a candidate list (CL) into a set S, according to a randomized greedy criterion based on a *benefit function*.

Secondly, a local search is applied with the objective of improving S. The best solution obtained from all iterations is returned. Algorithms 1 and 2 illustrate a GRASP.

Algorithm 1: Metaheuristic GRASP

```
Input  : Instance I
Output: Solution S
S ← ∅
while stop condition not satisfied do
    S' ← ConstructionPhase(I)
    S' ← LocalSearchPhase(I, S')
    if (S = ∅) ∨ (S' is better than S) then
        S ← S'
return S
```

Algorithm 2: Construction Phase

```
Input  : Instance I
Output: Solution S
S ← ∅
CL ← BuildCL(I)
while S is not feasible do
    CalculateBenefits(CL)
    v ← SelectElement(CL)
    S ← S ∪ {v}
return S
```

We developed four heuristics for PAP based on GRASP called Greedy Randomized (GR), Weighted Greedy Randomized (WGR), Random plus Greedy (RG), and Sampled Greedy (SG). They differ mainly by the approach adopted in the construction phase.

In each of them, while S is being augmented, we extend the propagation started from the initial seed set whenever a new seed is chosen. We also define the benefit function $b(v)$ as the number of ignorant neighbors of v.

In GR, we randomly select a new seed from $\{v \in \text{CL} : b(v) \geq b_{\max} - \lfloor \alpha\,(b_{\max} - b_{\min}) \rfloor\}$, for a given $\alpha \in [0, 1]$, where $b_{\min}$ and $b_{\max}$ are the minimum and maximum benefits.

For WGR, the criterion is the same, but the chances of v be selected are proportional to the number of neighbors that will immediately become spreaders if we add v into S.

For RG, we randomly choose a new seed from CL in the first p steps, for a given p. For the remaining steps, we select the vertex v that maximizes $b(v)$.

In SG, we get a random subset R of CL with size ℓ in each step, for a given ℓ. Then, we select the vertex $v \in R$ that maximizes $b(v)$ as the next new seed.

The local search of our heuristics are similar and based on the following idea. Given $S' \subset S$, we simulate a propagation starting from $S \setminus S'$. If $S \setminus S'$ is feasible, we make $S \leftarrow S \setminus S'$. Otherwise, we remove from S all vertices in S' that became spreaders.

Results

We generated a *benchmark* of 840 PAP instances. For 281 of them we obtained the optimum value using the integer programming formulation. Table 1 shows for how many of these 281 instances our heuristics produced an optimal solution.

Table 1: Optimal solutions attained by our heuristics

Heuristic	GR	WGR	RG	SG
Optimal	277	258	280	281

References

[1] G. Cordasco, L. Gargano, and A. A. Rescigno. "Active influence spreading in social networks". In: *Theoretical Computer Science*, 764 (2019), pp. 15–29.

[2] M. G. Resende and C. C. Ribeiro. "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications". In: *Handbook of Metaheuristics*. Springer, 2020, pp 283–319.