# Tackling the $\alpha$-Domination Problem Heuristically

Enrico Iurlano, Johannes Varga, and Günther R. Raidl

Algorithms and Complexity Group, TU Wien,
Favoritenstraße 9/192-01, Vienna, 1040, Austria

{eiurlano|jvarga|raidl}@ac.tuwien.ac.at

**Abstract.** We focus on the $\alpha$-domination problem, which is capable of modeling influence phenomena in social networks. It formally asks for a minimum cardinality subset of vertices of a given graph such that any vertex is either included in this subset or at least a fraction $\alpha$ of its neighbors is $(0 < \alpha \leq 1)$. We address the search for solutions of high quality within a tight margin of computation time by designing, firstly, a Greedy Randomized Adaptive Search Procedure and, secondly, a Configuration Checking metaheuristic. The latter excels in terms of solution quality and speed and is able to outperform an integer programming formulation solved by the commercial solver Gurobi on a majority of tested instances which have thousands of vertices.

**Keywords:** Domination · Influence in social networks · GRASP · Configuration checking

## 1 Introduction

The problem of finding a minimum dominating set of vertices of an undirected simple graph $G = (V, E)$ is one of the best-known and most intensely studied optimization problems on graphs. Dunbar et al. [6] introduced a generalization of this problem called *$\alpha$-domination problem*, in which for a fixed proportionality parameter $0 < \alpha \leq 1$ the goal is to find a minimum cardinality set $S \subseteq V$ such that each vertex $v \in V$ is already contained in $S$ or at least a fraction $\alpha$ of its neighbors is comprised by $S$; in both cases we say that $v$ is $\alpha$-dominated. The $\alpha$-domination problem can be seen as an interpolation between the classical domination problem, for sufficiently small $\alpha > 0$, and the vertex covering problem, for $\alpha = 1$. It is shown in [6] that for any $0 < \alpha \leq 1$, it is NP-hard to find an optimal solution; moreover, upper and lower bounds for optimal solutions on general graphs but also for special graph classes are derived.

This problem is strongly related to the so-called *Minimum Positive Influence Dominating Set* (MPIDS) problem introduced in [13], in which the aim is to maintain sufficient influence in a social network by utilizing the minimum number of influencing actors (here, even actors need to receive influence from their equals). In the age of constantly growing data on social networks and the increasing relevance of disinformation campaigns [3] and viral marketing [13], the $\alpha$-domination problem can be a more suitable model to determine how

This preprint was presented at the 19th International Conference on Computer Aided Systems Theory (EUROCAST 2024), Las Palmas de Gran Canaria, Spain, 27th of February 2024 (final version is to appear).

influence—even before consolidating it—is achievable by using a few simultaneously deployed actors of assumed loyal behavior.

Practical computational methods for the MPIDS problem and other problems with parallels on a conceptual level are recently receiving increased attention [2,10], such as, e.g., the *Target Set Selection Problem* [9], which studies the influence under a propagation aspect. Still there are only few approaches known to us that (heuristically) address the $\alpha$-domination problem. An important *caveat* should be taken into account: We have encountered some works, e.g., [11], in which the problem name MPIDS actually refers to 0.5-domination. Thus there has to be paid careful attention not to mix this problem up with the stronger constrained one of the original work [13] and its follow-up papers, e.g., [2].

The following notation together with some conventions will be used. We only consider undirected, simple graphs $G = (V, E)$. We denote for a vertex $v \in V$ by $N(v)$ its neighborhood, i.e., set of all adjacent vertices, and by $\deg(v) = |N(v)|$ the cardinality of the latter. Given a binary vector $c \in \{0,1\}^{|V|}$ associated to a graph $G = (V, E)$, let us call it an *$\alpha$-dominating labeling* if it is the incidence vector of an $\alpha$-dominating subset of vertices $S$, i.e., $c = (\mathbb{1}_S(v))_{v \in V}$ with $\mathbb{1}_S(v') := 1$ if $v' \in S$ and $\mathbb{1}_S(v') := 0$, otherwise. Moreover, assume that $\alpha \in (0, 1]$ is a fixed constant and let us denote by $\alpha$-Domination the problem of finding an $\alpha$-dominating labeling that minimizes the sum of labels (see later also (1)). Henceforth, given the locality of $\alpha$-Domination, we assume that our graph instances consist of a single connected component containing at least three vertices.

## 2 Solution approaches

The following is an integer linear program (IP) formulation of $\alpha$-Domination, where $\lceil \cdot \rceil$ denotes the Gaussian ceiling function:

$$\begin{aligned} \min \quad & \textstyle\sum_{v \in V} c_v \\ \text{subject to} \quad & \textstyle\sum_{w \in N(v)} c_w + c_v \cdot \lceil \alpha \deg(v) \rceil \geq \lceil \alpha \deg(v) \rceil, && v \in V \\ & c_v \in \{0,1\}, && v \in V \end{aligned} \tag{1}$$

*Remark 1.* For a slightly generalized version of $\alpha$-Domination, Raghavan and Zhang [11] proposed a strengthened formulation based on the idea of placing a dummy vertex on each edge. Within the setting of our computational evaluation (see Section 3) we found that this more recent formulation is surpassed in terms of solution quality by (1) and we therefore do not consider this alternative here.

Our first goal is to design a *Greedy Randomized Adaptive Search Procedure* (GRASP) [7], which should undercut the runtime of a state-of-the-art solver applied to (1) and should at the same time have a comparably good (or even better) solution quality. As the main component of this metaheuristic, we propose the following greedy construction heuristic for $\alpha$-Domination; see also

**Algorithm 1** Greedy construction

**Input**: Graph $G = (V, E)$;
**Output**: $\alpha$-dominating labeling $c$

1: $c_v \leftarrow 0\ \forall v \in V$ `// initialize {0,1}-labeling`
2: $d_v \leftarrow 0\ \forall v \in V$ `// current α-domination status (in {0,1})`
3: $U_v \leftarrow \deg(v)\ \forall v \in V$ `// number of not α-dominated neighbors`

4: **while** $\{v \in V : d_v = 0\} \neq \emptyset$ **do**
5: Let $A_v$, $v \in V$, be the number of neighbors of $v$ which would be newly $\alpha$-dominated if $c_v$ was set to 1.
6: Let $s \leftarrow \text{argmax}\{(A_v, U_v) : v \in V\}$ (ties are broken uniformly at random)
7: $c_s \leftarrow 1$
8: Update $d_s$, $U_s$; $d_{s'}$ for $s' \in N(s)$, $U_{s'}$ for $s' \in N(s)$.
9: **end while**

10: For $v \in V$, set $c_v \leftarrow 0$ whenever $c_v = 1$ and $c$ allows for updating $c_v$ to 0 without violating $\alpha$-domination of any vertex. `// postprocessing step`

11: **return** $c$

Algorithm 1: Initially all vertices are 0-labeled. Then, iteratively, the label of a 0-labeled vertex is set to 1; the choice of the vertex is here based on the number of *newly* $\alpha$-dominated vertices of $G$ which this update would yield. In the case of ties, the vertex having the highest number of not yet $\alpha$-dominated neighbors is picked in the spirit of providing progress in as many vertices as possible. Note that in particular for graph classes having frequent occurrences of vertices of degree one (leaves), the following preprocessing can be employed: Without any loss of optimality, the labels of leaves can be fixed to 0 whereas the labels of their neighbors are set to 1.

For the GRASP, we randomize this greedy heuristic by always 1-labeling a next vertex that is randomly chosen from the top-$\rho$ percentile of the candidate vertices, where $0\% < \rho \leq 100\%$ is the randomization parameter. Our GRASP follows the standard template from [7] by iteratively applying this randomized greedy heuristic for obtaining diversified starting solutions, and each of them undergoes a local search. Our local search relies on the following move operator DLC called "Decrease label and compensate". Given a feasible solution $c$ for $\alpha$-Domination together with the choice of a 1-labeled vertex $v$, we perform the update $c_v \leftarrow 0$ and compensate all violations of feasibility (now disallowing a change of $c_v$) by greedily solving an arising set multicovering problem: The set to be covered consists of the vertices that lost their $\alpha$-domination status and the set system consists of closed neighborhoods comprising these vertices either as neighbors or directly as "center" of the neighborhood; the multiplicity of coverage for $v$ depends on the number of its 1-labeled neighbors and is addressed in more detail in Algorithm 2, which relies on two auxiliary quantities: Given a labeling

$c$, let us define

$$e_v := \{w \in N(v) : c_w = 1\},\ v \in V, \tag{2}$$

respectively

$$isr_v := \left\{ w \in N(v) : c_w = 0,\ \sum_{z \in N(w)} c_z = \lceil \alpha \deg(w) \rceil \right\},\ v \in V, \tag{3}$$

capturing the set of 1-labeled neighbors of $v$, respectively the set of (0-labeled) neighbors of $v$ that are *indispensable support recipients* of $v$, i.e., which would lose their $\alpha$-domination status if we would carry out the update $c_v \leftarrow 0$.

Now having the DLC operator available, for a feasible solution $c$, we call another feasible solutions $c'$ a neighbor of $c$, denoted $c \sim c'$, if there is a vertex $v$ for which $c_v = 1$ and DLC applied to $c$ yields $c'$. Denote by $\mathcal{U}_k(c)$ the DLC-*neighborhood of order at most* $k$ of $c$ as the set of all $(s_v)_{v \in V} \in \{0,1\}^{|V|}$ which are $\alpha$-dominating and for which there are pairwise distinct labelings $c^{(1)}$, $c^{(2)}$, $\ldots$, $c^{(k)}$ satisfying $c \sim c^{(1)} \sim \cdots \sim c^{(k)} = s$.

For the local search within the GRASP, we fall back on $\mathcal{U}_K(\cdot)$ for some larger value $K$: As long as a quality-improving neighboring solution $c_{\text{next}}$ in $\mathcal{U}_K(\cdot)$ is found, the search is repeated in the solution-neighborhood of $c_{\text{next}}$, and the process iterates until this search for improvement fails. Due to the vast size of this neighborhood, the idea is to find a promising neighbor in $\mathcal{U}_K(\cdot)$ by employing the scoring function subsequently introduced in (4) and yielding a solution in $\mathcal{U}_{j+1}(\cdot)$ starting from one in $\mathcal{U}_j(\cdot)$. Eventually, each solution lying on the so-arising trajectory of solutions is in particular contained in $\mathcal{U}_K(\cdot)$ such that we postulate a sufficient local exploration around our starting solution and eventually consider the best-quality solution on this trajectory as our result of scanning $\mathcal{U}_K(\cdot)$.

Note that incremental updates of the quantities $e_v$ and $isr_v$ introduced above will not only be important for an efficiency gain in the context of several implementations but also allow the computation of the scoring function

$$\mathsf{Score}(v) := \begin{cases} -\infty & \text{if } c_v = 0, \\ |e_v| - \lceil \alpha \deg(v) \rceil - \tau |isr_v| & \text{if } c_v = 1, \end{cases} \tag{4}$$

which expresses the attractiveness of decreasing the label of a vertex: For already 0-labeled vertices $-\infty$ is returned, and for all other vertices, their "saturation" $|e_v| - \alpha \lceil \deg(v) \rceil$ provides a certain reward that is relativized by a penalty arising from a potentially large cardinality of $|isr_v|$; $\tau > 0$ is a parameter to be tuned.

Next, we propose an alternative approach based on *Configuration Checking* (CC) introduced in [4]; see Algorithm 3. Given a local search procedure relying on a move operator, the main idea here is to each time save for the part of the solution being subjected to the move operator the current so-called configuration (a snapshot of attributes of its neighboring solution parts). The move operator

**Algorithm 2** Decrease-Label-and-Compensate

**Input**: $G = (V, E)$; labeling $c$; $e$ and $isr$ already meeting (2)–(3); vertex $i \in V$ with $c_i = 1$
**Output**: *None*, overwrites all input arguments except for $G$ and $i$.

`// Decrease label and store not anymore` $\alpha$`-dominated vertices:`
1: $U \leftarrow isr_i$ `// these vertices will cause (local) infeasibility`
2: $c_i \leftarrow 0$ and update $e, isr$ where affected by this assignment

`// Compensate all arisen constraint violations:`
3: Build family of restricted sets $\mathcal{S} := \{e_w \cap U : w \in \bigcup_{v \in isr_i} (N(v) \setminus \{i\})\}$
4: Greedily, find a set covering of $U$ given by $e_{v_1} \cap U, \ldots, e_{v_k} \cap U$
5: **for** $\ell = 1, \ldots, k$ **do**
6: $\quad c_{v_\ell} \leftarrow 1$ and update $e, isr$ where affected by this assignment
7: **end for**
8: **while** $|e_i| < \lceil \alpha \deg(i) \rceil$ **do**
9: $\quad$ Pick a random 0-labeled vertex $j \in N(i) \setminus e_i$
10: $\quad c_j \leftarrow 1$ and update $e, isr$ where affected by this assignment
11: **end while**

`// Eliminate potential redundancies:`
12: **while** $\emptyset \neq I := \{v : c_v = 1 \wedge isr_v = \emptyset \wedge |e_v| \geq \lceil \alpha \deg(v) \rceil\}$ **do**
13: $\quad$ Randomly draw $v$ from $I$; $c_v \leftarrow 0$; update $e, isr$; re-evaluate $I$
14: **end while**
15: **return**;

can afterwards be re-applied on that part only if its now re-encountered circumstance of attributes differs from what is stored in the current snapshot. In the following, we rely on a "practical" [4] version of this metaheuristic requiring just the occurrence of a (potentially ultimately reverted) change of configurations in the history of configuration changes since the last move application. With the intention to avoid excessive cycling in the solution landscape, the technique is successfully applied to the vertex covering problem and to Boolean satisfiability solving [4].

In our setting, we fall back on the operator DLC and interpret each 1-labeled vertex $v$ as a solution part around which we consider its current configuration, namely the tuple capturing the restriction $c\big|_{N(v)}$.

*Remark 2.* In preliminary experimentation, we found, by analyzing convergence plots for the CC approach, that in exceptional situations a repeating cycle traversing a small set of solutions of equal quality occurred, lasting until the termination of the procedure. Therefore, we add an additional simple prevention strategy which is implemented in Line 11 of Algorithm 3.

**Algorithm 3** Configuration Checking

**Input**: Graph $G = (V, E)$; $b \in \mathbb{N}$, $\tau > 0$
**Output**: $\alpha$-dominating labeling of $G$

1: $c \leftarrow$ greedy solution via Algorithm 1
2: *configCh*$[v] \leftarrow 1 \; \forall v \in V$ `// tracker for changes of configuration`
3: **while** no stopping criterion satisfied **do**
4: Pick $v^*$ uniformly at random among the vertices which lie within the best $b$ candidates of $\{\mathsf{Score}(v) : v \in V \text{ with } c_v = 1 \text{ and } \textit{configCh}[v] = 1\}$.
5: DLC($c$, $e$, *isr*, $v^*$)
6: *configCh*$[v^*] \leftarrow 0$
7: *configCh*$[w] \leftarrow 1 \;\; \forall w \in N(v^*)$.
8: **if** $\sum_{v \in V} c_v < \sum_{v \in V} c_v^{\text{best}}$ **then**
9: $c^{\text{best}} \leftarrow c$
10: **end if**
11: **if** objective function value unchanged since at least 32 iterations **then**
12: In next iteration, in Line 4, pick $v^*$ among the $b$ best elements of $\{\mathsf{Score}(v) : v \in V \text{ with } c_v = 1 \text{ and } \textit{configCh}[v] = 0\}$ `// see Remark 2.`
13: **end if**
14: **end while**
15: **return** $c^{\text{best}}$

## 3 Computational results

We fix $\alpha = 0.5$ for $\alpha$-Domination, mainly motivated by the strong relationship to MPIDS and also the so-called *majority thresholds* [1] for the Target Set Selection Problem. All algorithms of Section 2 have been implemented in `Julia` 1.10, which also served with the `JuMP` package as an interface to `Gurobi` 10.0.3 [8] for solving the integer linear program (1).

Table 1 contains on the left side information on the graph instances we use for our experimental analysis; the first ten instances are adopted from [5], while the remaining three are taken from [12]. We emphasize that the latter instances constitute graphs from real-world social networks. The table reports the number of vertices, the average degree ("deg avg"), and the standard deviation of the degrees of the vertices ("deg std"). In preliminary tuning experiments suitable values were determined for the parameters of our procedures: The penalty coefficient is chosen as $\tau := 2$ for the DLC operator; the randomization parameter $\rho$ of the greedy construction heuristic as $\rho = 5\%$; the choice of $b := 4$ turned out to be meaningful for both the GRASP and the CC.

Table 1 further shows in Column "IP" results from the integer linear program: listed are incumbent solution values ("incmb.") and optimality gaps ("gap") $(z^* - z^{DB})/z^*$, where $z^*$ denotes the best solution found and $z^{DB}$ the dual bound. The next column "greedy avg" shows the average solution values over ten runs of the greedy Algorithm 1—experiments for the greedy algorithm have been ran multiple times due to the randomization present in the tie-breaker (see Line 6 of

the algorithm). Similarly, columns "GRASP" and "CC" report these quantities for the respective two metaheuristic approaches again averaged over ten runs; in addition objective values of a best run is additionally reported here. As in the experimental evaluation of [2], the time budget for each run of both metaheuristics was set to $|V|/100$ CPU seconds and is therefore magnitudes smaller than the 3600 seconds runtime reserved for `Gurobi`. All experiments were carried out using a single thread on a cluster endowed with an Intel(R) Xeon(R) E5-2640 v4 CPU with 2.40GHz and 160GB RAM running Ubuntu 18.04.6 LTS.

**Table 1.** Results of all proposed approaches. Time budget IP: 3600 seconds; time budget GRASP and CC: $|V|/100$ seconds.

| Instance | $\|V\|$ | deg | deg | IP | | greedy | GRASP | | CC | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | avg | std | incmb. | gap | avg | best | avg | best | avg |
| `Bipartite-350-350-80` | 700 | 280 | 8 | 350 | 19% | 357.0 | 347 | 347.7 | 346 | **346.7** |
| `Bipartite-350-350-90` | 700 | 316 | 6 | 350 | 18% | 354.6 | 348 | 348.4 | 348 | **348.0** |
| `Net-20-20` | 400 | 7 | 1 | **161** | 6% | 185.3 | 169 | 170.0 | 170 | 174.2 |
| `Net-30-20` | 600 | 8 | 1 | **242** | 8% | 279.2 | 256 | 258.5 | 251 | 263.9 |
| `Planar-650` | 650 | 86 | 15 | 310 | 24% | 329.5 | 312 | 314.6 | 306 | **308.5** |
| `Planar-700` | 700 | 88 | 15 | 334 | 24% | 356.2 | 338 | 340.3 | 332 | **333.0** |
| `Random-950-10` | 950 | 95 | 9 | 463 | 26% | 479.1 | 465 | 465.9 | 453 | **456.4** |
| `Random-950-20` | 950 | 191 | 12 | 473 | 26% | 480.4 | 471 | 473.2 | 460 | **462.0** |
| `Random-1000-10` | $1k$ | 100 | 9 | 488 | 26% | 503.5 | 489 | 491.1 | 477 | **480.3** |
| `Random-1000-20` | $1k$ | 200 | 13 | 494 | 25% | 507.0 | 497 | 499.0 | 485 | **486.4** |
| `socfb-Amherst41` | $2k$ | 81 | 63 | **740** | 15% | 807.0 | 801 | 804.4 | 769 | 775.5 |
| `socfb-Dartmouth6` | $8k$ | 79 | 75 | **2513** | 20% | 2588.9 | 2731 | 2782.8 | 2529 | 2548.7 |
| `socfb-Harvard1` | $15k$ | 109 | 112 | 7588 | 92% | 5263.3 | 5733 | 5793.5 | 5116 | **5158.8** |

Inspecting the results of Table 1, we notice generally quite large optimality gaps for the integer programming approach, in particular, for instances with higher average degrees. With one exception, the greedy construction heuristic always yielded solutions of poorer quality than the IP approach and these greedy solutions are considerably inferior to the ones resulting from the two metaheuristics. However, the greedy heuristic also is very fast, with a maximum runtime of 0.32 seconds for all instances except for `socfb-Dartmouth6` respectively `socfb-Harvard1` which took 3.05 seconds respectively 20.49 seconds. Due to the evident advantage of CC over GRASP, we conclude that even multiple restarts with randomized greedy solutions do seem to not direct the local search trajectory towards regions of solutions having the quality of those found by the CC approach. The latter, in fact, is able to achieve the best results on several instances. The only instances where this is not the case are the sparser graphs `Net-20-20` and `Net-30-20` as well as the social networks `socfb-Amherst41` and `socfb-Dartmouth6`; note the standard deviation of their degrees in contrast to the other instances. The last instance indicates that (social) networks with more

than 10.000 vertices are apparently better addressed heuristically, since here the (meta)heuristics considerably outperform the IP approach.

## 4 Conclusion

We proposed two metaheuristics that are able to rapidly generate solutions of high quality for $\alpha$-DOMINATION. Among these, CC turned out to be the more favorable one, which in general surpasses the IP approach already on medium-sized instances while using just a fraction of the computation time. As experimental results indicate, our proposed move operator seems to be an effective tool to find—in combination with the introduced scoring function—promising parts of the solution landscape. In future work we plan to increase performance specifically on social networks by trying to integrate into our greedy construction strategy, but possibly also in the selection of local moves, additional information such as, e.g., the deviation of the degree of a vertex from the average degree and by identifying alternative metrics that could help to address the desire for earlier—or even a priori—inclusion of suitable vertices into the $\alpha$-dominating set.

**Acknowledgments.** This research was funded in part by the Austrian Science Fund (FWF) [10.55776/W1260-N35] within the framework of the doctoral program "Vienna Graduate School on Computational Optimization". Moreover, we acknowledge partial support from Austria's Agency for Education and Internationalization (OeAD) under grant BA05/2023.

## References

1. Ackerman, E., Ben-Zwi, O., Wolfovitz, G.: Combinatorial model and bounds for target set selection. Theoretical Computer Science **411**(44-46), 4017–4022 (2010)
2. Akbay, M.A., López Serrano, A., Blum, C.: A self-adaptive variant of CMSA: application to the minimum positive influence dominating set problem. International Journal of Computational Intelligence Systems **15**(1), 44 (2022)
3. Bradshaw, S., Bailey, H., Howard, P.N.: Industrialized disinformation: 2020 global inventory of organized social media manipulation. Tech. rep., Computational Propaganda Research Project at the Oxford Internet Institute (2021), https://demtech.oii.ox.ac.uk/wp-content/uploads/sites/12/2021/01/CyberTroop-Report-2020-v.2.pdf, last accessed: 2023-11-15
4. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence **175**(9-10), 1672–1696 (2011)
5. Currò, V.: The Roman domination problem on grid graphs. Ph.D. thesis, Universitá di Catania (2014), https://hdl.handle.net/20.500.11769/585454
6. Dunbar, J.E., Hoffman, D.G., Laskar, R.C., Markus, L.R.: $\alpha$-domination. Discrete Mathematics **211**(1-3), 11–26 (2000)
7. Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. Journal of global optimization **6**, 109–133 (1995)
8. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2024), https://www.gurobi.com

9. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 137–146 (2003)
10. Lozano-Osorio, I., Oliva-García, A., Sánchez-Oro, J.: Dynamic path relinking for the target set selection problem. Knowledge-Based Systems **278**, 110827 (2023)
11. Raghavan, S., Zhang, R.: Rapid influence maximization on social networks: The positive influence dominating set problem. INFORMS Journal on Computing **34**(3), 1345–1365 (2022)
12. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 29 (2015), https://networkrepository.com
13. Wang, F., Camacho, E., Xu, K.: Positive influence dominating set in online social networks. In: Combinatorial Optimization and Applications, COCOA 2009. LNCS, vol. 5573, pp. 313–321. Springer (2009)