Negative Learning Ant Colony Optimization for the Minimum Positive Influence Dominating Set Problem

Albert López Serrano*
Universitat Autònoma de Barcelona
Bellaterra, Spain
albert.lopezi@e-campus.uab.cat

Teddy Nurcahyadi
Artificial Intelligence Research Institute (IIIA-CSIC)
Bellaterra, Spain
teddy.nurcahyadi@iiia.csic.es

Salim Bouamama
Department of Computer Science, Ferhat Abbas University Sétif 1
Sétif, Algeria
salim.bouamama@univ-setif.dz

Christian Blum
Artificial Intelligence Research Institute (IIIA-CSIC)
Bellaterra, Spain
christian.blum@iiia.csic.es

ABSTRACT

Recent research has shown that adding negative learning to ant colony optimization, in addition to the traditional positive learning mechanism, may improve the algorithms' performance significantly. In this paper we consider the application of this novel ant colony optimization variant to an NP-hard combinatorial optimization problem known as the minimum positive influence dominating set problem. This problem has applications especially in the context of social networks. Our results show, first, that the negative learning variant significantly improves over the standard ant colony optimization variant. Second, the obtained results show that our algorithm outperforms all competitors from the literature.

CCS CONCEPTS

• **Theory of computation → Discrete optimization**.

KEYWORDS

ant colony optimization; negative learning; minimum positive influence dominating set

ACM Reference Format:
Albert López Serrano, Teddy Nurcahyadi, Salim Bouamama, and Christian Blum. 2021. Negative Learning Ant Colony Optimization for the Minimum Positive Influence Dominating Set Problem. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3449726.3463130

1 INTRODUCTION

The minimum positive influence dominating set (MPIDS) problem [15, 16] is an NP-hard combinatorial optimization problem with applications in social networks. Hereby, each vertex represents an individual and edges indicate relationships, respectively interactions, between those individuals. The problems' background is that ideas and information propagated in social networks can have a significant impact. Social norms theory has shown that the behavior of individuals can be affected by perceptions of others' thoughts and behaviors [4]. Thus, exploiting the relationships among people in social networks can provide great benefits to both economy and society. The aim of the MPIDS problem is to identify a small subset of key influential individuals to speed up the spread of positive influence [5, 8]. Other applications of the MPIDS problem can be found in e-learning software [17], online business [13], drinking, smoking, and drug related problems [15]. In technical terms, the MPIDS problem can be described as follows. Given a simple, connected undirected graph $G = (V, E)$, the problem requires to find a dominating set of minimum cardinality such that at least half of the neighbors of each vertex form part of the dominating set. Most of the recent research efforts concerning the MPIDS problem were focused on greedy heuristics [2, 3, 12, 14, 16] and on two evolutionary approaches [6, 7].

In this paper we apply a very recent variant of ant colony optimization (ACO) to the MPIDS problem. This ACO variant [11] is characterized by the fact that it incorporates a negative learning component in addition to the traditional positive learning mechanism that makes use of good solutions for updating the pheromone values. In [11], this negative learning ACO approach was shown to outperform earlier negative learning approaches such as, for example, [10]. The negative learning component of this new variant is based on applying an exact solver to sub-instances of the original problem instance. The results that are obtained in this way for sub-instances, are used for updating an additional set of pheromone values: the negative pheromone values. Both sets of pheromone values are used for generating solutions at each iteration. Our results show that the negative learning variant of ACO outperforms both the standard ACO variant and the metaheuristic approaches from the literature for the MPIDS problem.

The rest of this paper is organized as follows. In Section 2 we provide the standard integer linear programming (ILP) model for the MPIDS problem. In Section 3, we outline our algorithmic proposal. Finally, in Section 4 we present and discuss the experimental results.

*Undergraduate Student

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463130

Algorithm 1 $\text{ACO}^{+}_{\text{neg}}$ for the MPIDS problem

1: **input:** a problem instance $G = (V, E)$
2: **parameters:** n_a, d_{rate}, ρ, ρ^{neg}, t_{ILP}
3: $S^{\text{bs}} := V$, $S^{\text{rb}} := V$, $cf := 0$, bs_update := FALSE
4: InitializePheromoneValues($\mathcal{T}, \mathcal{T}^{\text{neg}}$)
5: **while** termination condition not met **do**
6: $\mathcal{S}^{\text{iter}} := \emptyset$
7: **for** $k := 1, \ldots, n_a$ **do**
8: $S^k :=$ ConstructSolution($\mathcal{T}, \mathcal{T}^{\text{neg}}$)
9: $\mathcal{S}^{\text{iter}} := \mathcal{S}^{\text{iter}} \cup \{S^k\}$
10: **end for**
11: $S^{\text{sub}} :=$ SolveSubinstance($\mathcal{S}^{\text{iter}}$)
12: $S^{\text{ib}} := \text{argmin}\{f(S) \mid S \in \mathcal{S}^{\text{iter}} \cup \{S^{\text{sub}}\}\}$
13: **if** $f(S^{\text{ib}}) < f(S^{\text{rb}})$ **then** $S^{\text{rb}} := S^{\text{ib}}$
14: **if** $f(S^{\text{ib}}) < f(S^{\text{bs}})$ **then** $S^{\text{bs}} := S^{\text{ib}}$
15: ApplyPheromoneUpdate(cf, bs_update, S^{ib},S^{rb},S^{bs}, S^{sub})
16: $cf :=$ ComputeConvergenceFactor($\mathcal{T}$)
17: **if** $cf > 0.999$ **then**
18: **if** bs_update = TRUE **then**
19: $S^{\text{rb}} :=$ NULL, and bs_update := FALSE
20: InitializePheromoneValues($\mathcal{T}, \mathcal{T}^{\text{neg}}$)
21: **else**
22: bs_update := TRUE
23: **end if**
24: **end if**
25: **end while**
26: **output:** S^{bs}, the best solution found by the algorithm

2 MPIDS PROBLEM: ILP MODEL

The MPIDS problem can easily be stated in terms of an ILP as follows. The model is based on a binary variable x_i associated to each vertex $v_i \in V$.

$$\textbf{Minimize} \quad \sum_{i=1}^{n} x_i \tag{1}$$

$$\textbf{Subject to} \quad \sum_{v_j \in N(v_i)} x_j \geq \left\lceil \frac{deg(v_i)}{2} \right\rceil \quad \forall v_i \in V \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

Hereby, $N(v_i)$ is the neighborhood of v_i in the input graph G, and $deg(v_i)$ is the degree of vertex v_i, that is, the number of its neighbors. Equation (2) ensures that a feasible solution contains at least half of the neighbors of each vertex $v_i \in V$.

3 THE PROPOSED ALGORITHM

Our approach—called $\text{ACO}^{+}_{\text{neg}}$—is based on a standard $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System (MMAS) implemented in the hyper-cube framework [1]; see Algorithm 1 for the pseudo-code. As any other MMAS algorithm, $\text{ACO}^{+}_{\text{neg}}$ keeps three different solutions at any time: (1) the best solution generated at the current iteration, that is, the *iteration-best* solution S^{ib}, (2) the best solution generated since the last restart of the algorithm, that is, the *restart-best* solution S^{rb}, and (3) the best solution generated overall, that is, the *best-so-far* solution S^{bs}. Both S^{bs} and S^{rb} are initialized to V, which is the worst solution possible; see line 3. Moreover, a Boolean control variable (bs_update) used for controlling the pheromone update is initialized to FALSE. The model $\mathcal{T}$ (standard pheromone values) contains a value τ_i for each vertex $v_i \in V$. Similarly, the model of the negative pheromone values ($\mathcal{T}^{\text{neg}}$) contains a value τ_i^{neg} for each vertex $v_i \in V$. Function InitializePheromoneValues($\mathcal{T}, \mathcal{T}^{\text{neg}}$) initializes all pheromone values from $\mathcal{T}$ to 0.5, and all pheromone values from $\mathcal{T}^{\text{neg}}$ to $\tau_{\text{min}} = 0.001$. At each iteration of the algorithm, n_a solutions are generated based on greedy information and on pheromone information, and stored in set $\mathcal{S}^{\text{iter}}$; lines 6–10. A detailed description of function ConstructSolution($\mathcal{T}, \mathcal{T}^{\text{neg}}$) is provided below. Subsequently, a sub-instance of the original problem instance is generated in function SolveSubinstance($\mathcal{S}^{\text{iter}}$) on the basis of $\mathcal{S}^{\text{iter}}$ as follows. Vertices that are found in all solutions from $\mathcal{S}^{\text{iter}}$ are stored in set S_{in} and vertices that are found in none of the solutions from $\mathcal{S}^{\text{iter}}$ are stored in S_{out}. Then, the ILP model from the previous section is solved by applying the ILP solver CPLEX 12.10, after adding a constraint $x_i = 1$ for all vertices from S_{in} and a constraint $x_i = 0$ for all vertices from S_{out}. The computation time limit of CPLEX is set to t_{ILP} seconds, which is one of the parameters of the algorithm. When CPLEX stops after maximally t_{ILP} seconds, it either returns the optimal solution to the sub-instance, or the best solution found within the allowed computation time. In both cases the obtained solution is denoted by S^{sub}. After updating solutions S^{ib}, S^{rb} and S^{bs} in lines 12–14, the pheromone update is conducted in function ApplyPheromoneUpdate(cf, bs_update, S^{ib},S^{rb},S^{bs}, S^{sub}); see below for a detailed description. Finally, the value of the convergence factor is determined in function ComputeConvergenceFactor($\mathcal{T}$), and—in case $cf > 0.999$ and bs_update = TRUE—the algorithm is restarted (see lines 17–24). In the following, the main functions of the algorithm are described in more detail.

ConstructSolution($\mathcal{T}$): The solution construction mechanism is adopted from the newest available greedy algorithm for the MPIDS problem [2]. First, a pre-processing procedure adds those vertices to a set S_{init} that must form part of an optimal solution. Note that this pre-processing procedure is, of course, only executed once by $\text{ACO}^{+}_{\text{neg}}$. Each solution construction process starts by initializing the partial solution under construction to S_{init}, that is, $S := S_{\text{init}}$. Given any partial solution $S \subset V$, the number of neighbors of v that must be added to S in order to cover v is computed as $h_S(v) := \lceil \frac{deg(v)}{2} \rceil - |N_S(v)|$, where $N_S(v) := N(v) \cap S$ denotes the set of neighbors of $v \in V$ belonging to S. We say that v is covered if $h_S(v) \leq 0$, and not covered otherwise. Moreover, let $C_S \subset V$ denote the set of uncovered vertices with respect to S. At each step of the solution construction procedure, first a vertex $v_i \in C_S$ is chosen such that $deg(v_i) \leq deg(v_j)$ for all $v_j \in C_S$. Then, $h_S(v_i)$ vertices from $N(v_i) \setminus N_S(v_i)$ are chosen, based on greedy information and on pheromone information. The probability to choose a vertex $v_k \in N(v_i) \setminus N_S(v_i)$ is defined as follows:

$$\mathbf{p}(v_k \mid S) := \frac{\eta_k \cdot \tau_k \cdot (1 - \tau_k^{\text{neg}})}{\sum_{v_l \in N(v_i) \setminus N_S(v_i)} \eta_l \cdot \tau_l \cdot (1 - \tau_l^{\text{neg}})} \tag{4}$$

where $\eta_k := |\{v \in N(v_k) : h_S(v) > 0\}| + 1$. Based on these probabilities, the selection of v^*—that is, the vertex to be added to S—is

done as follows. First, a random number $r \in [0,1]$ is chosen uniformly at random. In case $r \leq d_{\text{rate}}$, $v^* := \text{argmax}\{\mathbf{p}(v \mid S) \mid v \in N(v_i) \setminus N_S(v_i)\}$. Otherwise, v^* is chosen by *roulette wheel selection*. Note that $d_{\text{rate}} \in [0,1]$—the so-called determinism rate—is an important parameter of the algorithm.

ApplyPheromoneUpdate(cf, bs_update, S^{ib},S^{rb},S^{bs}, S^{sub}): The pheromone value update has two parts. First, the update of the standard pheromone values from $\mathcal{T}$ is the same as in any other MMAS algorithms implemented in the hypercube framework. In particular, solutions S^{ib}, S^{rb}, and S^{bs} are used in the following way. The update weight of each solution is determined on the basis of the convergence factor (cf) and the value of bs_update as shown in the following.

	bs_update = FALSE				bs_update
	$cf < 0.4$	$cf \in [0.4, 0.6)$	$cf \in [0.6, 0.8)$	$cf \geq 0.8$	= TRUE
κ_{ib}	1	2/3	1/3	0	0
κ_{rb}	0	1/3	2/3	1	0
κ_{bs}	0	0	0	0	1

Each pheromone value τ_i is updated as follows: $\tau_i := \tau_i + \rho \cdot (\xi_i - \tau_i)$, where $\xi_i := \kappa_{ib} \cdot \Delta(S^{\text{ib}}, v_i) + \kappa_{rb} \cdot \Delta(S^{\text{rb}}, v_i) + \kappa_{bs} \cdot \Delta(S^{\text{bs}}, v_i)$, and $\rho \in [0,1]$ is the learning rate. Note that κ_{ib} is the weight of solution S^{ib}, κ_{rb} the one of solution S^{rb}, and κ_{bs} the one of solution S^{bs}. Moreover, $\Delta(S, v_i)$ evaluates to 1 if and only if $v_i \in S$. Otherwise, the function evaluates to 0. Note also that (according to the table above) $\kappa_{ib} + \kappa_{rb} + \kappa_{bs} = 1$. After this pheromone update, those values that exceed $\tau_{\max} = 0.999$ are set back to $\tau_{\max}$, and those values that have dropped below $\tau_{\min} = 0.001$ are set back to $\tau_{\min}$. This prevents the algorithm from reaching the state of complete convergence.

The second part of the pheromone update concerns the modification of the negative pheromone values from $\mathcal{T}^{\text{neg}}$. In particular, this update only concerns the negative pheromone values of those vertices that form part of at least one solution of $\mathcal{S}^{\text{iter}}$. The update formula is as follows: $\tau_i^{\text{neg}} := \tau_i^{\text{neg}} + \rho^{\text{neg}} \cdot (\xi_i^{\text{neg}} - \tau_i^{\text{neg}})$, where ρ^{neg} is the negative learning rate and $\xi_i^{\text{neg}} = 1$ if $c_i \notin S^{\text{sub}}$, resp. $\xi_i^{\text{neg}} = 0$ otherwise. In other words, the negative pheromone values of those components that do not form part of S^{sub} are increased.

ComputeConvergenceFactor($\mathcal{T}$): The convergence factor cf is computed in all MMAS algorithm implemented in the hypercube framework in the same way:

$$cf := 2\left(\left(\frac{\sum_{\tau \in \mathcal{T}} \max\{\tau_{\max} - \tau, \tau - \tau_{\min}\}}{|\mathcal{T}| \cdot (\tau_{\max} - \tau_{\min})}\right) - 0.5\right)$$

This means that the value of cf is zero when all standard pheromones have value 0.5. In contrast, when all pheromones have either value $\tau_{\min}$ or $\tau_{\max}$, cf evaluates to one. In all other cases, cf has a value between 0 and 1.

4 EXPERIMENTAL EVALUATION

Three versions of the proposed algorithm are evaluated. In addition to the full version ($\text{ACO}^+_{\text{neg}}$), we also evaluated the following versions: (1) the standard MMAS version (henceforth simply called ACO). This version is obtained by not executing the update of the negative pheromone values and by setting $S^{\text{sub}} = \emptyset$ at each iteration (that is, line 11 of Algorithm 1 is not executed); (2) ACO_{neg}, which is obtained by removing S^{sub} from the argmax-operation in line 12, that is, S^{sub} is not used for the standard pheromone update, only for the update of the negative pheromone values. All experiments concerning the three algorithm versions were performed on a cluster of machines with Intel® Xeon® CPU 5670 CPUs with 12 cores of 2.933 GHz and a minimum of 32 GB RAM. Moreover, for solving the sub-instances in ACO_{neg} and $\text{ACO}^+_{\text{neg}}$ we used CPLEX 12.10 in one-threaded mode. The three algorithms were evaluated on 17 social networks that are usually used in the literature on the MIPDS problem. These networks are of small and medium size. In addition, we evaluated the algorithms on 10 larger social networks from the SNAP library (https://snap.stanford.edu/data/).

Table 1: Tuning results obtained by irace. Left: Small-/medium size networks. Right: large networks.

Algorithm	n_a	d_{rate}	ρ	ρ^{neg}	t_{ILP}
ACO	10	0.4	0.1	n.a.	n.a.
ACO_{neg}	20	0.1	0.1	0.5	30
$\text{ACO}^+_{\text{neg}}$	20	0.0	0.1	0.2	17

Algorithm	n_a	d_{rate}	ρ	ρ^{neg}	t_{ILP}
ACO	2	0.6	0.1	n.a.	n.a.
ACO_{neg}	10	0.7	0.3	0.5	20
$\text{ACO}^+_{\text{neg}}$	14	0.6	0.2	0.5	13

Our algorithms require well-working values for n_a (number of solution constructions per iteration), d_{rate} (determinism rate), and ρ (learning rate). ACO_{neg} and $\text{ACO}^+_{\text{neg}}$ require additionally a value for ρ^{neg} (negative learning rate) and a value for t_{ILP} (time limit for CPLEX per iteration). We made use of the scientific tuning software irace [9] for the purpose of parameter tuning. This tool was used for generating two parameter settings for each variant: one of the 17 small/medium sized instances, and another one for the 10 large networks. Networks `CA-AstroPh`, `Email-Enron` amd `socfb-Brandeis99` were used for the first tuning experiment, and networks `Amazon0312` and `com-youtube` were used for the second one. Finally, for each tuning experiment the budget was fixed to 2000 runs, each one with a time limit of 600 CPU seconds. The considered parameter value domains were as follows: $n_a \in \{2, \ldots, 20\}$, $d_{\text{rate}} \in \{0.0, 0.1, 0.2, \ldots, 0.8, 0.9\}$, $\rho, \rho^{\text{neg}} \in \{0.1, \ldots, 0.5\}$, and $t_{\text{ILP}} \in \{1, \ldots, 30\}$ (in seconds). The obtained parameter value settings are shown in Table 1.

The three ACO versions were applied 10 times, with a CPU time limit of 600 CPU seconds, to each of the 27 problem instances. The results, in comparison to those of HSIA [7], ILPMA [6], and CPLEX (with a time limit of 2 hours per instance) are shown in Table 2. The first 17 of these instances are generally used in the related literature and are of small, resp. medium, size. CPLEX was able to solve 11 of these instances to optimality, as indicated by a value of 0.00 in the column labeled 'Gap (%)'. Note that both HSIA and ILPMA were only applied to 9, resp. 12, of these first 17 problem instances. The remaining 10 instances are larger and were taken from the SNAP database. The results of our three ACO versions are marked by a light gray column background. They are separated into the best result obtained in 10 runs, the average results over 10 runs, and the average computation time.

Table 2: Numerical results

Network	CPLEX		best					average					average time			
	Result	Gap (%)	HSIA	ILPMA	ACO	ACO_{neg}	ACO^{+}_{neg}	HSIA	ILPMA	ACO	ACO_{neg}	ACO^{+}_{neg}	ILPMA	ACO	ACO_{neg}	ACO^{+}_{neg}
Karate	**15**	0.00	n.a.	**15**	**15**	**15**	**15**	n.a.	**15.0**	**15.0**	**15.0**	**15.0**	0.03	0.00	0.002	0.002
Dolphins	**30**	0.00	n.a.	**30**	**30**	**30**	**30**	n.a.	**30.0**	**30.0**	**30.0**	**30.0**	0.13	0.004	0.008	0.004
Football	**63**	0.00	n.a.	65	64	**63**	**63**	n.a.	65.65	64.6	**63.0**	**63.0**	0.54	74.08	40.71	19.73
Jazz	**79**	0.00	n.a.	n.a.	**79**	**79**	**79**	n.a.	n.a.	79.9	**79.0**	**79.0**	n.a.	5.92	0.39	0.12
CA-AstroPh	**6740**	0.30	6905	6857	6886	6742	6742	6906.6	6865.45	6897.1	6744.1	**6743.8**	300.41	502.58	282.73	438.62
CA-GrQc	**2587**	0.00	2597	2594	2588	**2587**	**2587**	2598.4	2596.05	2589.1	**2587.0**	**2587.0**	45.07	144.90	2.96	0.65
CA-HepPh	**4718**	0.01	4791	4770	4769	4720	4720	4792.4	4773.85	4772.9	4721.4	**4720.3**	157.43	526.81	341.16	347.01
CA-HepTh	**4471**	0.00	4515	4502	4494	**4471**	**4471**	4516.2	4506.25	4496.7	**4471.0**	**4471.0**	107.93	468.96	13.44	14.62
CA-CondMat	**9584**	0.06	9729	9683	9692	9587	9588	9734.0	9689.6	9696.3	9588.8	**9588.4**	506.37	432.98	394.02	471.59
Email-Enron	**11682**	0.00	11865	11814	11826	11685	11684	11873.4	11818.95	11832.9	11685.2	**11684.1**	760.08	440.34	221.48	164.53
ncstrlwg2	**2994**	0.00	3004	3001	2998	**2994**	**2994**	3005.4	3002.85	2998.9	2994.9	**2994.1**	65.69	327.34	18.70	259.49
actors-data	**3092**	0.24	3143	3130	3145	3093	3093	3144.5	3134.5	3149.0	**3093.7**	**3093.7**	137.74	419.42	273.89	260.08
ego-facebook	**1973**	0.00	1726[a]	1737[a]	1974	**1973**	**1973**	1726.6[a]	1741.55[a]	1974.9	1973.6	**1973.1**	56.91	16.44	33.27	65.28
socfb-Brandeis99	1400	1.41	n.a.	n.a.	1456	1398	**1397**	n.a.	n.a.	1462.7	1399.0	**1397.7**	n.a.	398.99	347.46	480.68
socfb-nips-ego	**1398**	0.00	n.a.	n.a.	**1398**	**1398**	**1398**	n.a.	n.a.	1398.0	**1398.0**	**1398.0**	n.a.	2.86	2.71	1.28
socfb-Mich67	1329	1.56	n.a.	n.a.	1384	1329	**1327**	n.a.	n.a.	1387.9	1329.9	**1328.5**	n.a.	420.84	335.62	366.89
soc-gplus	**8244**	0.00	n.a.	n.a.	8294	**8244**	**8244**	n.a.	n.a.	8298.3	8244.1	**8244.0**	n.a.	446.28	169.60	21.20
musae_git	**9752**	0.00	n.a.	n.a.	10383	10006	9872	n.a.	n.a.	10409.3	10031.3	**9828.8**	n.a.	589.06	414.45	357.81
loc-gowalla_edges	**67617**	0.07	n.a.	n.a.	68815	67946	67943	n.a.	n.a.	68836.9	67972.0	**67964.0**	n.a.	547.53	550.78	503.48
gemsec_facebook_artist	**15194**	1.20	n.a.	n.a.	16010	15537	15480	n.a.	n.a.	16029.0	15593.9	**15505.2**	n.a.	554.54	488.09	511.42
deezer_HR	54573	95.68	n.a.	n.a.	23413	22906	**22840**	n.a.	n.a.	23434.7	23152.1	**22904.9**	n.a.	518.96	426.34	426.67
com-youtube	**351281**	0.00	n.a.	n.a.	353715	352110	351556	n.a.	n.a.	353975.2	352243.7	**351567.1**	n.a.	601.46	592.07	599.96
com-dblp	**120492**	0.08	n.a.	n.a.	121854	120998	120853	n.a.	n.a.	121874.6	121056.7	**120932.0**	n.a.	465.34	564.34	513.59
Amazon0302	262111	97.50	n.a.	n.a.	134146	132797	**131836**	n.a.	n.a.	134241.0	132832.6	**131901.3**	n.a.	370.61	395.71	576.98
Amazon0312	400727	95.41	n.a.	n.a.	180443	180613	**180049**	n.a.	n.a.	180546.3	180690.8	**180284.1**	n.a.	597.82	597.19	611.33
Amazon0505	410236	95.19	n.a.	n.a.	182851	182839	**182152**	n.a.	n.a.	182955.3	182928.9	**182464.5**	n.a.	596.00	601.65	617.66
Amazon0601	403394	96.94	n.a.	n.a.	179768	179726	**179112**	n.a.	n.a.	179847.5	179799.0	**179662.2**	n.a.	598.81	598.46	612.19
average					49351.48	4906.88	**48966.59**			49381.25	49137.72	**49017.73**	n.a.	372.92	285.45	305.29

[a]: the papers on HSIA and ILPMA must have used a different ego-facebook instance, as their result values are below the optimal solution value derived by CPLEX

The following observations can be made. First, the standard ACO version obtains results similar to those of HSIA and ILPMA. The two ACO versions with negative learning (ACO_{neg} and ACO^{+}_{neg}) clearly outperform the other three competitors. Concerning the comparison between ACO_{neg} and ACO^{+}_{neg} it can be stated that, in the context of the 17 small/medium size instances, ACO^{+}_{neg} is only slightly better than ACO_{neg}. However, this difference in quality grows significantly in the context of the 10 larger problem instances. Moreover, CPLEX clearly fails to provide solutions of reasonable quality in the case of five large problem instances, that is, the deezer_HR instance, and the four Amazon instances. All three ACO versions are clearly superior to CPLEX in these cases.

Summarizing we can state that ACO^{+}_{neg} is the new state-of-the-art metaheuristic for solving the MPIDS problem. Moreover, this shows again that making use of negative learning in addition to positive learning can be very beneficial.

ACKNOWLEDGMENTS

This work was funded by project CI-SUSTAIN of the Spanish Ministry of Science and Innovation (PID2019-104156GB-I00).

REFERENCES

[1] Christian Blum and Marco Dorigo. 2004. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34, 2 (2004), 1161–1172.

[2] Salim Bouamama and Christian Blum. 2021. An Improved Greedy Heuristic for the Minimum Positive Influence Dominating Set Problem in Social Networks. *Algorithms* 14, 3 (2021), 79.

[3] Mai Fei and Chen Weidong. 2016. An improved algorithm for finding minimum positive influence dominating sets in social networks. *Journal of South China Normal University* 48, 3 (2016), 59–63.

[4] Angela K. Fournier, Erin Hall, Patricia Ricke, and Brittany Storey. 2013. Alcohol and the social network: Online social networking sites and college students' perceived drinking norms. *Psychology of Popular Media Culture* 2, 2 (2013), 86.

[5] Dilek Günneç, Subramanian Raghavan, and Rui Zhang. 2020. Least-cost influence maximization on social networks. *INFORMS Journal on Computing* 32, 2 (2020), 289–302.

[6] Geng Lin, Jian Guan, and Huibin Feng. 2018. An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Physica A: Statistical Mechanics and its Applications* 500 (2018), 199–209.

[7] Geng Lin, Jinyan Luo, Haiping Xu, and Meiqin Xu. 2020. A Hybrid Swarm Intelligence-Based Algorithm for Finding Minimum Positive Influence Dominating Sets. In *Proceedings of ICNC-FSKD 2019 – Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, Yong Liu, Lipo Wang, Liang Zhao, and Zhengtao Yu (Eds.). Springer International Publishing, 506–511.

[8] Cheng Long and Raymond Chi-Wing Wong. 2011. Minimizing seed set for viral marketing. In *2011 IEEE 11th International Conference on Data Mining*. IEEE press, 427–436.

[9] Manuel López-Ibánez et al. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43 – 58.

[10] James Montgomery and Marcus Randall. 2002. Anti-pheromone as a Tool for Better Exploration of Search Space. In *ANTS 2002 – 3rd International Workshop on Ant Algorithms (Lecture Notes in Computer Science)*, Marco Dorigo, Gianni Di Caro, and Michael Sampels (Eds.), Vol. 2463. Springer Berlin Heidelberg, 100–110.

[11] Teddy Nurcahyadi and Christian Blum. 2021. Adding Negative Learning to Ant Colony Optimization: A Comprehensive Study. *Mathematics* 9, 4 (2021), 361.

[12] Jiehui Pan and Tian-Ming Bu. 2019. A Fast Greedy Algorithm for Finding Minimum Positive Influence Dominating Sets in Social Networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 360–364.

[13] Amir Afrasiabi Rad and Morad Benyoucef. 2011. Towards detecting influential users in social networks. In *International Conference on E-Technologies*. Springer, 227–240.

[14] Hassan Raei, Nasser Yazdani, and Masoud Asadpour. 2012. A new algorithm for positive influence dominating set in social networks. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 253–257.

[15] Feng Wang, Erika Camacho, and Kuai Xu. 2009. Positive influence dominating set in online social networks. In *International Conference on Combinatorial Optimization and Applications*. Springer, 313–321.

[16] Feng Wang, Hongwei Du, Erika Camacho, Kuai Xu, Wonjun Lee, Yan Shi, and Shan Shan. 2011. On positive influence dominating sets in social networks. *Theoretical Computer Science* 412, 3 (2011), 265–269.

[17] Guangyuan Wang. 2014. *Domination problems in social networks*. Ph.D. Dissertation. University of Southern Queensland.