Applying Answer Set Programming and Metaheuristics to the Volunteer Assignment Problem with Preferences

Francisco R. Dutra[1], **Thiago P. Valente**[1], **Carlos V. Araújo**[2], **Alexandre M. Arruda**[3]

[1]Universidade Federal do Ceará (UFC)
62900-000, Cidade Universitária, Russas

[2]Universidade Estadual do Ceará (UECE)
60714-903, Itaperi, Fortaleza - CE

[3]Universidade Federal do Ceará (UFC)
63902-580, Cedro Novo, Quixadá

{francisco.riandutra, thiagop00}@alu.ufc.br,

carlos.vinicio@aluno.uece.br, alexandre.arruda@ufc.br

Abstract. *The Volunteer Assignment Problem, an NP-hard variant of the Generalized Assignment Problem, poses significant computational challenges for exhaustive search. This work evaluates three strategies to maximize a volunteer-preference-based function: exact search via Answer Set Programming (ASP), and two hybrid approaches combining a greedy heuristic or ASP itself with Simulated Annealing and Genetic Algorithm refinement. The exact method alone exceeded the time limit without proving optimality, whereas the hybrid strategies delivered high-quality solutions efficiently. The ASP-initialized GA achieved the best overall performance, outperforming the constructive heuristic and maximizing allocation quality*

1. Introduction

The organization of large-scale academic events requires careful logistical planning, particularly regarding volunteer allocation. In the context of the University Meetings of the Federal University of Ceará – Russas Campus, this task is carried out mostly manually, taking into account constraints such as volunteer availability and preferences. This process is prone to errors and becomes progressively more complex as the number of participants grows.

Given this scenario, the following problem arises: how to efficiently allocate volunteers in real-world events while respecting multiple operational constraints and individual preferences, without incurring high computational costs or yielding low-quality solutions? Although exact and heuristic approaches exist for allocation problems, the practical application of these techniques to real-world instances still presents challenges, especially in NP-hard problems, where obtaining optimal solutions may become infeasible in polynomial time [Garey and Johnson 1979, Talbi 2009].

The roots of this problem, known as the Volunteer Assignment Problem (VAP), can be traced back to [Sampson 2006], who first formalized the distinction between traditional labor assignment and volunteer assignment. Mathematically, the VAP can be modeled as an extension of the Generalized Assignment Problem (GAP)

[Burkard et al. 2012], belonging to the NP-hard class [Garey and Johnson 1979]. Unlike paid employees on fixed schedules, volunteers have irregular availability, heterogeneous skills, and strong personal preferences, in addition to operating under a constrained labor pool without the binding labor costs typical of traditional assignment. These characteristics require a fundamentally different modeling approach to ensure engagement and operational efficiency.

For NP-hard problems, heuristics and metaheuristics are widely employed: heuristics use problem-specific rules to explore feasible solutions quickly, while metaheuristics such as Simulated Annealing (SA) [Kirkpatrick et al. 1983] and Genetic Algorithm (GA) [Goldberg 1989] guide the search through iterative improvement. Exact methods such as Mixed Integer Linear Programming (MILP) [Wolsey and Nemhauser 1999] and Constraint Programming (CP) [Rossi et al. 2006] can obtain optimal solutions but often scale poorly on large instances. Similar hybrid refinement and decomposition strategies, combining heuristics with exact logic solvers, have recently succeeded in other combinatorial domains, such as vehicle routing with Max-SAT [Ferreira and Arruda 2025, Silva et al. 2026]. A complementary alternative is Answer Set Programming (ASP) [Brewka et al. 2011], a declarative paradigm whose solver Clingo [Gebser et al. 2012] can obtain exact or high-quality initial solutions for complex combinatorial instances, combining logic-based expressiveness with competitive solving performance.

This work proposes a formal model of the Volunteer Assignment Problem using the ASP language and evaluates exact and hybrid approaches for the VAP on instances based on the University Meetings Russas Campus, comparing: (i) exact method via ASP; (ii) greedy heuristic + SA/GA; and (iii) ASP + SA/GA. The goal is to identify the strategy that best balances solution quality and computational cost.

2. Theoretical Background

2.1. The Volunteer Assignment Problem (VAP)

The VAP consists of assigning v volunteers $I = \{1, \ldots, v\}$ to f roles $J = \{1, \ldots, f\}$ over s time slots $T = \{1, \ldots, s\}$, potentially encompassing multiple constraints and penalties [da Silva Neto et al. 2024].

However, for the scope of this work, a relaxed version of the original problem was adopted. The objective focuses exclusively on maximizing overall satisfaction, ensuring that volunteers are assigned to their preferred roles within the time slots for which they indicated availability. The simplified objective function maximizes:

$$\text{Maximize } Z = \sum_{i \in I} \sum_{t \in T} \sum_{j \in J} P_{ij} \cdot x_{itj} \tag{1}$$

where P_{ij} is the preference weight of volunteer i for role j, and $x_{itj} \in \{0, 1\}$ indicates whether i was assigned to role j in time slot t. The model is subject to operational constraints regarding minimum engagement, uniqueness of role per time slot, minimum and maximum demand coverage per role, and individual availability.

2.2. Answer Set Programming (ASP)

Answer Set Programming is a declarative paradigm aimed at combinatorial search problems [Brewka et al. 2011]. In answer set programming, a solution is given by a *stable model*, which consists of a set of atoms that satisfy the defined rules and facts. For assignment problems, the language uses aggregators to restrict tasks and optimization statements to select, among the possible models, the one that best satisfies the objective function.

Processing is carried out by the *Clingo* system [Gebser et al. 2012], which integrates the grounding (variable instantiation) and solving (solution search) stages. An ASP program is composed of:

- **Facts:** Represent input data and *a priori* true knowledge (e.g., João is a volunteer);
- **Choice Rules:** Define the search space, allowing the solver to generate different combinations of solutions (e.g., assigning volunteers to tasks);
- **Integrity Constraints:** Rules that eliminate models violating problem conditions (e.g., a volunteer cannot be assigned to two simultaneous tasks).

2.3. Simulated Annealing (SA)

Simulated Annealing is a single-trajectory metaheuristic inspired by metallurgical annealing [Kirkpatrick et al. 1983]. Starting from an initial solution π, the algorithm generates neighbors via volunteer *swap* and accepts inferior solutions with probability

$$P(\Delta F, T) = \exp\left(\frac{\Delta F}{T}\right), \tag{2}$$

where ΔF is the fitness variation between the candidate and current solution. The temperature T follows a geometric cooling schedule $T_{k+1} = \alpha T_k$, with $\alpha = 0.99$, starting at $T_0 = 100$ and terminating after L iterations without improvement.

2.4. Genetic Algorithm (GA)

The Genetic Algorithm [Goldberg 1989] evolves a population of $N = 100$ solutions over $G = 1000$ generations via roulette-wheel selection, period-based single-point crossover, which exchanges complete time-slot assignments between two parents (rate p_c), and cyclic mutation, which rotates role assignments among three randomly selected volunteers (rate $p_m = 0.05$). The fitness of each individual, to be maximized, is computed as:

$$\text{Fitness}(\pi) = \sum_{k=1}^{m} \sum_{i=1}^{n} P(t_k, v_i, f), \tag{3}$$

where $P(t_k, v_i, f)$ is the preference of volunteer v_i for role f to which they were assigned in time slot t_k.

3. Related Work

The work of [Reis 2024] investigated volunteer allocation in a family reunification program run by an international NGO, using Microsoft Excel Solver to maximize the number

of volunteers assigned to weekly shifts. The model considered individual availability and operational constraints, such as a maximum of 20 weekly hours, at least one shift per volunteer, and a parity constraint ensuring male volunteers per shift were at least equal to female volunteers. The study highlighted Solver's cost-free availability and ease of use, requiring no advanced programming knowledge, but noted software limitations, including difficulties and excessive processing time when solving the model with a maximum-volunteers-per-shift constraint.

The work of [Sousa 2019] proposed an initial ASP-based approach to the Crew Scheduling Problem in urban public transportation. The main advantage was ASP's readability and its ability to model labor constraints and complex business rules straightforwardly, eliminating the need for the extensive variable glossary required by traditional ILP models. The Clingo solver found feasible solutions within reasonable time for small-scale instances, covering planning horizons of up to three days, though the author concluded that ASP's computational performance was inferior to ILP for large-scale instances.

The study of [da Silva Neto et al. 2024] developed a Mixed Integer Linear Programming model for scheduling volunteers in technical support and multimedia teams of an NGO, aiming to assign volunteers to tasks matching their highest skill levels while meeting technical needs. The objective function maximized allocation weights based on volunteer preferences, with penalties preventing excessive workload and consecutive-week assignments, alongside constraints on scheduling frequency, role coverage, and reported availability. The model found optimal, infeasibility-free solutions in under 40 seconds, compared to manually produced schedules that took over 10 minutes and contained allocation errors requiring further correction.

The present work differs from the aforementioned studies by combining ASP as an initial solution generator for metaheuristics, systematically investigating the influence of initialization quality on the performance of SA and GA.

4. Methodology

4.1. Instances

The instances were synthetically generated from the real parameters of the University Meetings, preserving volunteer privacy. Ten instances were created varying in: (i) number of volunteers (36 to 60) and (ii) availability density (restricted to abundant). The number of roles was fixed at 5 across all instances. Each volunteer is described by an ID, an ordered preference list, and binary availability per time slot.

The roles and their capacity bounds per time slot are: Section Coordinator (cs): $[12, 12]$; Cultural Section Coordinator (csc): $[10, 10]$; Floater (vol): $[4, 4]$; Event Registration (re): $[4, 8]$; Accreditation (cre): $[6, 8]$. Preferences follow a scale from 5 (most preferred) to 1 (least preferred).

The first five instances presented in Table 1 concentrate the largest portion of the experimental set, with 52 to 60 volunteers and availability densities ranging from 66.2% to 83.1%. Instance 1 is the most constrained: 20.7% of volunteers have only two available time slots, reducing the feasible space for maximum-preference assignments. Instance 4 presents the most uniform profile, with all volunteers available in exactly three or four

time slots. Instances 3 and 5, with the highest densities (83.1% and 77.0%) and the largest volunteer totals (58 and 60), represent the cases with the largest search space.

Instances 6 through 10 form the smaller group, with 36 to 48 volunteers, yet with higher availability densities ranging from 79.2% to 100.0%. Instance 10 represents the extreme case: all 36 volunteers are available across all five time slots, resulting in maximum density and a completely homogeneous profile. Paradoxically, this abundance of availability does not translate into greater flexibility for preference-based optimization. The determining factor is availability slack, defined as the difference between the total available slots ($N \times 5$) and the minimum capacity required by the event (36 slots per time slot $\times$ 5 time slots $=$ 180 slots). In Instance 10, the slack is zero: every available slot is strictly required to cover the minimum demand, leaving no margin to assign volunteers according to their preferences. Instance 8, with 48 volunteers and a density of 79.2%, most closely resembles the profile of Instances 1–5 and, consistently, yields the best results within the second group.

Table 1. Instance characteristics

Inst.	N	Dens.	Avg. p./vol.	Availability dist.			
				2p	3p	4p	5p
1	58	66.2%	3.31	12	23	16	7
2	52	80.0%	4.00	2	8	30	12
3	58	83.1%	4.16	0	1	47	10
4	55	74.9%	3.75	0	14	41	0
5	60	77.0%	3.85	0	11	47	2
6	38	96.8%	4.84	0	0	6	32
7	42	86.7%	4.33	0	5	18	19
8	48	79.2%	3.96	0	7	36	5
9	46	83.0%	4.15	0	4	31	11
10	36	100.0%	5.00	0	0	0	36

Below, an example of a line from one of the instance files, illustrating the input format:

```
João; Section Coordinator, Cultural Section
Coordinator, Floater, Accreditation, Event
Registration; yes; yes; yes; yes; no
```

4.2. ASP Modeling

The problem was modeled using the declarative approach of Answer Set Programming (ASP) and structured to be processed by the Clingo solver, divided into facts (instance) and logical rules (knowledge). The flexibility of the modeling allows adaptation to different editions of the event, parameterizing elements such as the set of roles, volunteer assignment bounds, among others. The problem is described by three main domains. For the experimental instances used in this work, the sets were defined as:

- **Volunteers**: $V = \{v_1, v_2, \ldots, v_n\}$;

- **Roles**: $F = \{cs, csc, vol, cre, re\}$;
- **Time slots**: $P = \{quaM, quaT, quiM, quiT, sexM\}$.

The predicates used to represent the instance and the decision variables were:

- **period(P)**: indicates a valid event time slot P;
- **available(V, P)**: volunteer V is available in time slot P;
- **preference(V, F, S)**: volunteer V has satisfaction S for role F;
- **min_bound(F, L)**: role F has lower bound L;
- **max_bound(F, K)**: role F has upper bound K;
- **assign(V, F, P)**: decision variable indicating that V was assigned to role F in time slot P.

In the instances used in this work, the satisfaction values S, as well as the minimum (L) and maximum (K) assignment bounds, were standardized to ensure experimental consistency. Preferences follow a decreasing scale from 5 to 1, where 5 represents the most preferred role and 1 the least preferred.

The volunteer concept is derived indirectly from their availability:

```
volunteer(V) :- available(V, _).
```

The choice rule establishes the problem's search space. The directive uses a cardinality constraint to define that, for each volunteer V available in a time slot P, the solver must choose to assign them to at most one role F. The generation is conditioned on the existence of a declared preference:

```
0 { assign(V, F, P) : preference(V, F, S) } 1
:- available(V, P).
```

The hard constraints eliminate models that violate the event's rules. Three main rules were implemented: **Minimum Engagement:** To prevent excessive team fragmentation and ensure commitment, every volunteer considered in the model must be assigned to at least two distinct time slots:

```
:- volunteer(V), #count { P : assign(V, F, P) } < 2.
```

Capacity Bounds (Minimum and Maximum): For each role F and time slot P, the count of assigned volunteers must strictly respect the lower (L) and upper (K) bounds defined in the instance:

```
:- min_bound(F, L), period(P), #count
{ V : assign(V, F, P) } < L.
:- max_bound(F, K), period(P), #count
{ V : assign(V, F, P) } > K.
```

The VAP objective in this context is to maximize the overall volunteer satisfaction. This is modeled by an optimization directive that instructs Clingo to search, among the feasible models, for the one that yields the highest sum of satisfaction values S associated with each assignment carried out:

```
#maximize { S,V,F,P : assign(V, F, P),
preference(V, F, S) }.
```

4.3. Greedy Heuristic

To generate feasible initial solutions, a greedy heuristic was developed in Python with a worst-case time complexity of $O(N^2 \cdot P^2 \cdot F)$, where N, P, and F represent the total number of volunteers, time slots, and roles, respectively. The algorithm performs ordered iterations over volunteers and their respective availability; at each assignment, a dual prioritization criterion is adopted: (1) filling roles whose minimum bounds have not yet been met, and (2) the preference order declared by the individual. This strategy ensures full coverage of the event's critical demands without violating the maximum allowed capacities.

4.4. Metaheuristics and Hybrid Strategies

Six hybrid strategies were evaluated: (H1) Heuristic + SA; (H2) Heuristic + GA; (H3) Heuristic + SA + GA; (H4) ASP + SA; (H5) ASP + GA; (H6) ASP + SA + GA. Parameters were defined empirically from preliminary calibration experiments. For SA, an initial temperature of $T_0 = 100$, a cooling factor of $\alpha = 0.99$, and swap-based perturbation were used. For GA, a population of 100 individuals, 1000 generations, a mutation rate of 5%, and period-based single-point crossover were adopted.

4.5. Computational Environment

Experiments were run on an AMD Ryzen 7 5825U (16 cores), 16 GB RAM, Windows 11 64-bit. The heuristic, SA and GA were implemented in Python 3.10.19; the exact method uses Clingo 5.8 via CLI with 16 threads, seed 2022, a timeout of 3600s (exact) and 300s (initial solution). It is worth noting that the average time taken by Clingo to find a feasible initial solution was only 0.0263 seconds. Being negligible, this value was not added to the execution time of the hybrid approaches. The source code and detailed results are publicly available at Github[1].

5. Results

5.1. Evaluation Metrics

Method performance was assessed using three main metrics. **Execution time** (in seconds) captures computational cost. **Solution quality** is expressed by the objective function value. For stochastic approaches (SA and GA), each configuration was run 100 times per instance, reporting the **best cost** (C) and **average cost** (M) — the former reflecting the approach's maximum potential, the latter its stability, reducing the effect of atypically favorable runs.

For the exact method (ASP/Clingo), the solver also provides bounds [lb; ub], where ub is the best feasible (primal) value found and lb is the dual bound from relaxing the problem, such that $\text{lb} \leq x^* \leq \text{ub}$. The optimality gap, $(\text{ub} - \text{lb}) \,/\, \text{ub}$, quantifies the maximum distance from the global optimum; a positive gap at the time limit indicates optimality was not proven.

[1] https://github.com/FRianOD/VAP-Artigo

5.2. Exact Method: ASP

Table 2 reports the results obtained from the native execution of the exact method via Clingo. The solver exhausted the 3600-second time limit on all 10 instances without proving global optimality. The lower and upper bounds obtained delimit the search space for each instance within the time limit.

Table 2. Exact method results (ASP / Clingo)

Instance	Time (s)	Bounds	Quality
instance 1	3600	[-944; -750]	750
instance 2	3600	[-1739; -964]	964
instance 3	3600	[-2430; -909]	909
instance 4	3600	[-1589; -965]	965
instance 5	3600	[-2157; -1005]	1005
instance 6	3600	[-2041; -690]	690
instance 7	3600	[-2006; -668]	668
instance 8	3600	[-1543; -849]	849
instance 9	3600	[-1949; -739]	739
instance 10	3600	[-1679; -654]	654

5.3. Hybrid Approaches: Heuristic + Metaheuristics

The results of the hybrid approaches initialized by the greedy heuristic are presented in Table 3.

Table 3. Heuristic + Metaheuristic Approaches.

	H+SA			H+GA			H+SA+GA		
Inst.	**C**	**M**	**T**	**C**	**M**	**T**	**C**	**M**	**T**
inst. 1	853	838.41	0.48	882	879.50	59.70	883	879.15	45.26
inst. 2	864	849.30	0.54	898	894.86	60.14	898	894.83	45.11
inst. 3	878	861.37	0.56	913	909.12	60.96	913	909.09	45.80
inst. 4	879	857.82	0.57	907	903.57	60.06	907	903.78	44.84
inst. 5	875	860.63	0.59	911	908.54	61.07	912	907.46	45.39
inst. 6	836	822.20	0.53	866	862.97	45.32	867	863.44	39.41
inst. 7	823	807.05	0.56	849	846.48	46.52	850	846.41	39.18
inst. 8	849	831.31	0.58	876	872.23	46.39	876	871.59	39.77
inst. 9	845	830.54	0.57	879	874.69	46.61	878	873.61	39.31
inst. 10	820	804.86	0.56	847	844.32	45.14	848	844.61	38.52

Note: C = best cost, M = average cost, T = average time (s).

5.4. Hybrid Approaches: ASP + Metaheuristics

Table 4 presents the performance data obtained by the metaheuristics when initialized from a solution generated by Clingo.

Table 4. ASP + Metaheuristic Approaches.

	ASP+SA			ASP+GA			ASP+SA+GA		
Inst.	**C**	**M**	**T**	**C**	**M**	**T**	**C**	**M**	**T**
inst. 1	880	860.66	0.81	909	903.74	48.36	910	905.09	47.50
inst. 2	928	910.75	0.86	970	966.22	52.38	970	966.64	46.17
inst. 3	962	943.28	0.90	1006	1001.38	52.57	1006	1001.84	51.44
inst. 4	939	919.01	0.87	982	977.50	51.17	982	977.85	43.94
inst. 5	952	934.02	0.93	998	994.61	51.64	1000	994.86	49.33
inst. 6	841	826.31	0.79	870	866.80	43.66	871	867.53	39.87
inst. 7	831	816.24	0.79	861	856.44	49.60	860	856.77	41.82
inst. 8	873	854.06	0.87	900	896.81	42.99	901	897.19	39.44
inst. 9	878	858.40	0.85	910	905.47	46.13	911	906.01	41.67
inst. 10	821	804.63	0.81	849	844.75	49.66	848	844.29	41.16

Note: C = best cost, M = average cost, T = average time (s).

6. Discussion of Results

The experimental data allow the analysis to be organized along four axes: the behavior of the exact method, the systematic superiority of ASP-based initialization, the relationship between instance complexity and solution quality, and the quality-time efficiency frontier.

6.1. Behavior of the Exact Method

Clingo exhausted the 3600-second time limit on all 10 instances without proving optimality, with substantial gaps between the reported lower and upper bounds. The most revealing aspect, however, is not the timeout itself, but the practical outcome: the average quality of solutions found by the exact method is only 819.3 points, a value lower than all six evaluated hybrid approaches, including the simplest one (H+SA: 852.2 points, delivering solutions in under 1 second). This phenomenon is expected in NP-hard problems: by prioritizing optimality proof, the solver distributes computational effort toward exploring the dual space (bound tightening) at the expense of searching for high-quality primal solutions [Talbi 2009, Wolsey and Nemhauser 1999]. The result confirms that, in the VAP, optimality certification is computationally prohibitive and that strategies oriented toward primal quality are more effective in contexts with real time constraints.

6.2. Systematic Superiority of ASP-Based Initialization

The most robust finding is that ASP-based initialization outperforms greedy heuristic initialization on all 10 instances for both metaheuristics, without exception. The average gain is 38.3 points for SA and 42.7 for GA (best cost over 100 runs), with a similar pattern for average cost (+36.4 and +41.7). This consistency across the result distribution [Sarhani et al. 2023] indicates the effect is systematic, not attributable to metaheuristic stochasticity.

6.3. Relationship Between Instance Complexity and Solution Quality

When analyzing instances by group, an important pattern emerges: the benefit of ASP-based initialization is considerably more pronounced in larger instances (1–5: 52–60 vol-

unteers) than in smaller ones (6–10: 36–48 volunteers). For SA, the average gain on instances 1–5 is 62.4 points, compared to 14.2 points on instances 6–10; GA follows an analogous pattern, 70.8 versus 14.6 points. This asymmetry is directly related to the availability slack of each group, defined as the difference between the total available slots and the minimum event capacity. Instances 1–5 present an average slack of 35.6 (ranging from 12 to 61), while instances 6–10 have an average slack of only 5.4 (ranging from 0 to 10). In instances with reduced slack, nearly all available slots are already required to cover the minimum demand, restricting the margin for preference-based optimization regardless of the initial solution. In instances with greater slack, the search space is broader and the quality of the starting point more decisively determines which basins of attraction will be explored [Blum and Roli 2003, Hoos and Stützle 2005].

6.4. Quality-Time Efficiency Frontier

The joint analysis of both metrics exposes two dominant points on the Pareto frontier of the evaluated approaches. The first is ASP+SA: the average best cost per instance is 890.5 points at an average time of 0.85 seconds, outperforming the exact method by 8.7% and the H+GA approach by 7.7 absolute points. This implies that, for scenarios where response speed is critical, ASP+SA represents a dominant strategy: it delivers superior quality to the most expensive heuristic alternative (H+GA) at a fraction of the computational cost — a roughly 63$\times$ reduction in refinement time relative to H+GA (0.85 s vs. 53.2 s). The second standout point is ASP+GA: the best absolute quality among all approaches, with an average of 925.5 points, outperforming the exact method on 9 out of 10 instances, with an average gain of +13.0% at approximately 74$\times$ lower time (48.8 seconds versus 3600 seconds). The only instance in which the exact method outperforms ASP+GA (instance 5: 1005 vs. 998 points) represents a margin of 0.7%, and even in that case ASP+SA+GA achieved 1000 points, less than 0.5% from the exact value. Finally, the ASP+SA+GA combination does not show consistent gains over ASP+GA alone, suggesting that the preliminary SA phase adds no relevant value to the refinement pipeline when GA already operates on a high-quality initial solution.

6.5. Summary

The results consolidate three empirically supported conclusions: (i) the exact ASP method is infeasible standalone for real-scale VAP instances within practical time limits; (ii) the quality of the initial solution has a direct, universal, and instance-difficulty-dependent impact on metaheuristic performance; and (iii) the ASP+GA approach represents the best quality–computational cost trade-off, while ASP+SA offers competitive performance in virtually zero time, making it the rational choice in latency-sensitive contexts.

7. Conclusion

This work proposed a formal ASP model of the Volunteer Assignment Problem (VAP) and evaluated six hybrid strategies combining two initial solution generators, greedy heuristic and ASP, with two metaheuristic refiners, SA and GA. Experiments on ten synthetic instances derived from real parameters confirm three central findings: (i) the exact ASP method alone is insufficient for real-scale instances within practical time limits, producing lower-quality solutions than any hybrid approach even at 3600 seconds; (ii) ASP-based initialization systematically outperforms the constructive heuristic across all instances and

both metaheuristics, with average gains of 38.3 points for SA and 42.7 for GA, amplifying with instance availability slack; and (iii) ASP+GA offers the best absolute quality, averaging 13.0% gain over the exact method at 74× lower time, while ASP+SA delivers competitive results in under 1 second, making it the rational choice under time constraints.

Limitations. The results were obtained on synthetic instances of small to medium scale (36–60 volunteers, 5 roles, 5 time slots), which limits scalability claims. The objective function adopted is a simplified version of the model from [da Silva Neto et al. 2024], without penalties for excessive workload or consecutive time slots. Furthermore, the absence of formal statistical tests (e.g., Wilcoxon) prevents claims of superiority with controlled significance, which is recommended in broader comparative studies.

Future work includes six natural extensions: **(1) Validation on real editions:** applying ASP+GA to real schedules to compare computational quality against volunteers' perceived satisfaction. **(2) Scalability:** testing instances with 100+ volunteers and multiple tracks, where ASP initialization should benefit more as availability slack grows. **(3) Alternative refiners:** evaluating Tabu Search and ant colony algorithms, whose diversification mechanisms may explore different basins from ASP-generated solutions. **(4) Full objective function:** reintroducing overload and consecutive-slot penalties to test whether ASP+GA dominance holds under a richer evaluation function. **(5) Automatic parameter tuning:** using machine learning or Bayesian optimization to replace empirical SA/GA calibration. **(6) Integration as a lower bound:** feeding refined solutions back to Clingo as a *warm start*, reducing the initial gap and potentially proving optimality on medium-scale instances.

References

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

Brewka, G., Eiter, T., and Truszczyński, M. (2011). Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103.

Burkard, R. E., Dell'Amico, M., and Martello, S. (2012). *Assignment Problems, Revised Reprint*. SIAM.

da Silva Neto, V. S., França, R. L. d. M., Silva, K. K. N., da Costa, L. C. A., and Kramer, H. H. F. R. (2024). A mixed-integer linear programming model for scheduling volunteers in technical support teams in Non-Governmental Organizations. *Socioeconomic Analytics*, 2(1):19–36.

Ferreira, P. and Arruda, A. (2025). Post-improving the capacitated vehicle routing problem using a max-sat solver. In *Anais do VI Workshop Brasileiro de Lógica*, pages 32–39, Porto Alegre, RS, Brasil. SBC.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2012). *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence. Morgan & Claypool Publishers.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Hoos, H. and Stützle, T. (2005). *Stochastic Local Search: Foundations and Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Reis, E. R. d. S. (2024). Alocação otimizada de voluntários em programa de restabelecimento de laços familiareas por uma ONG Internacional.

Rossi, F., van Beek, P., and Walsh, T. (2006). *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. Elsevier Science.

Sampson, S. E. (2006). Optimization of volunteer labor assignments. *Journal of Operations Management*, 24(4):363–377. Operations Management in Not-For-Profit, Government and Public Services: Innovative Applications and Case Studies.

Sarhani, M., Voß, S., and Jovanovic, R. (2023). Initialization of metaheuristics: comprehensive review, critical analysis, and research directions. *International Transactions in Operational Research*, 30(6):3361–3397.

Silva, P., Silva, F., Dantas, R., Araújo, C., and Arruda, A. (2026). A clustering-based maxsat hybrid approach for the capacitated vehicle routing problem with time windows. In *Anais do XI Encontro de Teoria da Computação*, pages 1–5, Porto Alegre, RS, Brasil. SBC.

Sousa, F. Y. d. (2019). Uma abordagem inicial em answer set programming para o problema de programação de tripulação.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley.

Wolsey, L. and Nemhauser, G. (1999). *Integer and Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley.