UAlg
UNIVERSIDADE DO ALGARVE

Metaheuristics, 2025/2026: MAX-SAT 3

Diogo Paiva Almeida	Ricardo Vicente	Fernando Lobo
Student number: 81477	Student number: 81450	Responsible Lecturer

Introduction

In this assignment, we implemented and evaluated two metaheuristics, Particle Swarm Optimisation (PSO), Simulated Annealing (SA) and compared their performance against four hillclimbing based methods: Next Ascent Hillclimbing (NAHC), Variable Neighbourhood Hillclimbing (VNH) and their multistart versions. We computed statistics for each algorithm, such as the mean best solution and the evaluations count used. Finally, we assessed statistical significance using a Kruskal–Wallis test to detect global differences across all algorithms, followed by pairwise Mann–Whitney tests to determine which specific algorithm pairs differed in performance.

The problem: MAX-SAT

The Maximum Satisfiability problem aims to find a truth assignment to n variables x_i … x_n that maximises the number of satisfied clauses (fitness). For example, if problem A has two variables x_1 and x_2 (n = 2), and one clause $(x_1\ and\ x_2)$, to achieve maximum satisfiability (fitness = 1 in this case), both variables must be assigned True, as any other combination of True or False would result in the clause being false.

Algorithms

Particle Swarm Optimisation (PSO)

PSO is a populational and multi-state method, inspired in the movement of flocks of birds or school of fish. It uses a swarm of particles where each particle is a solution that moves through the search space. Particles adjust their position with their velocity v. Each particle x_i velocity is calculated based on its own best fitness found x^*, the global best fitness found x^+ and its own neighbour's fitness $x^!$, by the formula:

$$v_i = \alpha v_i + \beta(x_i^* - x_i) + \gamma(x_i^+ - x_i) + \delta(x_i^! - x_i)$$

The values $\alpha, \beta, \delta, \gamma$ are algorithm parameters.

21/11/2025

Simulated Annealing

SA is a single-state method, inspired by the annealing process, where a heated material is slowly cooled to reach a stable crystalline structure. The algorithm starts from an initial random assignment of 1's and 0's (True and False) to the variables x_i ... x_n, this solution's fitness is evaluated and compared iteratively against a neighbouring solution. If the neighbour solution is better, it is accepted; if it is worse, it may still be accepted with a probability that depends on a temperature parameter. As the temperature decreases according to a cooling schedule, the algorithm becomes less likely to accept worse moves, allowing wide exploration early on and more focused search later.

Default Setup

For every tested method we utilised a default configuration of parameters, this facilitates replication of results and guarantees comparability between algorithms and instances.

MAX-SAT Instances

Three instances of the MAX-SAT problem were tested:

1. uf20-01.cnf, 20 variables, 91 clauses;
2. uf100-01.cnf, 100 variables, 430 clauses
3. uf250-01.cnf, 250 variables, 1065 clauses

Each algorithm was executed independently 30 times per instance, utilising a limit of maximum function evaluations of 1000000 evaluations. Each run returned the solution, best fitness achieved and the total of evaluations used.

Particle Swarm Optimisation Parameters

PSO parameters were configured to rely more heavily on informants. This choice comes from preliminary tests showing that a strong global-best component made the swarm collapse too quickly, therefore resulting in little exploration of the search space.

File	Num Particles	Informants	α	β	γ	δ	Max V
uf20	30	6	0.73	1.496	0.75	1.496	0.4
uf100	75	6	0.73	1.496	0.75	1.496	0.4
uf250	350	6	0.73	1.496	0.75	1.496	0.4

The parameter selection was based on a combination of theoretical grounding, stability and empirical behaviour observed during preliminary testing. We initially adopted the Clerc-Kennedy constriction factor (Clerc & Kennedy, 2002), a well-established configuration known to ensure a stable exploration. However, early experiments showed limited exploration, likely due to the structure of MAX-SAT highly rugged search space. To address this, we reduced the global-best coefficient to 0.75, shifting the algorithm towards an informant-based search. We also reduced the initial maximum velocity from 4 to 0.4, these adjustments significantly improved exploration, reduced premature convergence, and led to an average increase of 4.86% in the best fitness found across all instances. (see calculations on results.xlsx, sheet PARAMS)

For the swarm size, we started with a relatively small, arbitrary value (20 particles for each instance) and then progressively doubled the number of particles. This process revealed a clear trend: the larger the search space is, the more beneficial it is to use a larger swarm. A small number of particles is sufficient for small instances, but for medium and large instances it tends to converge too quickly and fails to explore the search space. Also, very large swarms can be computationally wasteful on small instances, but they perform much better on larger instances, achieving broader exploration, resulting in higher quality solutions.

The results of García-Nieto and Alba, who demonstrated that neighbourhood sizes of about six provide an excellent balance between exploration and exploitation in PSO, served as the basis for the decision to use six informants for each particle. Their results indicate that too few informants slow down information propagation and limit the swarm's ability to coordinate promising search directions, while too many informants effectively collapse the topology into a global-best structure, accelerating convergence at the cost of diversity. In order to promote wider exploration, six informants establish a well connected neighbourhood. In our experiments, this configuration led to more consistent improvements (from the initial 4 informants) and a more robust search, which is consistent with our goal of prioritising informant-based search over global-best attraction.

Simulated Annealing Parameters

For Simulated Annealing, the parameter configuration was selected to ensure a long, controlled cooling schedule compatible with the evaluation budget.

File	Minimum Temperature	Maximum Temperature	Cooling α
uf20	0.000001	1.0	0.9999930923
uf100	0.000001	1.0	0.9999930923
uf250	0.000001	1.0	0.9999930923

The maximum temperature was set to 1.0, providing enough freedom for the early acceptance of worse moves and ensuring broad exploration of the search space. The minimum temperature was defined as 10^{-6}, forcing the algorithm to gradually shift into a greedy regime as the search goes on. The cooling factor α = 0.9999930923 was computed so that the temperature decays smoothly from 1.0 to the minimum threshold over the full one million evaluations. These three values were chosen both to guarantee that the cooling process aligns precisely with the evaluation budget.

This configuration provides a balanced trade-off: high exploration in the early phase, casually accepting worse solutions to avoid local optima, followed by progressively more exploitation, and a final deterministic phase that stabilises the search near global optima.

Hillclimb Parameters

All hillclimbing algorithms share the same basic parameters, with the exception of Variable Neighbourhood Hillclimbing and its multistart version, which introduce an additional structural parameter: the maximum neighbourhood size k. In our implementation, this value was fixed at k = 3 in accordance with the assignment requirements. This parameter controls how many bits can be flipped simultaneously when the algorithm attempts to escape local optima, distinguishing VNH and MS-VNH from the single-bit hillclimbers.

Random Seed Selection

To ensure reproducibility while preserving independence between runs, each algorithm was executed 30 times per instance with explicitly controlled random seeds. For run r, with $r \in \{0,\ldots,29\}$, we set both the Python pseudorandom generator (random.seed) and NumPy's generator (np.random.seed) to the value r.

Performance Assessment

In this section we compare the performance of all algorithms on the three MAX-SAT instances in terms of solution quality and computational cost.

Solution Quality (Fitness)

The goal for all algorithms is to maximise the number of satisfied clauses. The global optimum for each instance is known:

File	Global Optimum (Fitness)
uf20	91
uf100	430
uf250	1065

Computational Cost (Evaluations)

The goal here is for all algorithms to minimise the number of function evaluations while achieving good solution quality. A function evaluation corresponds to computing the fitness of a candidate solution.

Stochastic Algorithms

The selection of the initial solution, the neighbour selection, mutations or movements in the search space depend on random number generators. As a consequence, two executions of the same algorithm with the same parameters may produce different results.

The stochastic behaviour present in these metaheuristics make it necessary to perform multiple independent runs and apply non-parametric tests to compare their performance reliably.

Kruskal-Wallis

Kruskal-Wallis was the non-parametric statistical test chosen.

The null hypothesis of the Kruskal-Wallis test states that all algorithms come from the same distribution (in other words, they have equal median performance). A low p-value indicates that at least one of the algorithms differs significantly from the others.

The alternative hypothesis is the opposite of the null hypothesis: At least one of the groups (algorithms), comes from a different distribution, meaning that at least one algorithm has a significantly different median performance compared to others. (It does not say which one, therefore later we run Mann-Whitney pairwise tests.)

Kruskal-Wallis Results

For each instance, the Kruskal–Wallis test was applied to both solution quality and computational cost, considering all six algorithms simultaneously. The null hypothesis states that all algorithms have equal median performance. The table below summarises the Kruskal-Wallis statistic *H* and the corresponding *p*-value.

Solution Quality:

Instance	H	p	Statistical Difference?
uf20	121	1.76×10^{-24}	Yes
uf100	153	2.78×10^{-31}	Yes
uf250	150	8.87×10^{-31}	Yes

Computational Cost:

Instance	H	p	Statistical Difference?
uf20	130	1.68×10^{-26}	Yes
uf100	159	1.50×10^{-32}	Yes
uf250	165	6.60×10^{-34}	Yes

The results were very clear, here's what we found:

1. All *p*-values are extremely small, between 6.60^{-34} and 1.76^{-24} (well below $p \leq 0.05$), for every instance and both metrics, so we reject the null hypothesis that all algorithms have the same median performance.
2. There are statistically significant differences between algorithms in terms of solution quality and computational cost.
3. The large Kruskal-Wallis statistic ($H \approx$ 120-165) indicate that these differences aren't small, the rank distributions are clearly separated.
4. Since there are statistical differences, It is appropriate to perform pairwise Mann-Whitney U tests to identify which specific algorithms differ from each other.

Pairwise Statistical Comparisons (Mann-Whitney U Tests)

We must apply pairwise tests because Kruskal-Wallis just tells us there are statistical differences between groups, but it doesn't tell us which groups. With pairwise statistical comparisons we can find exactly which groups differ from others, and which of the two is better.

To investigate the differences in performance among the six metaheuristics- PSO, SA, and four hillclimbers, we will perform nine targeted pairwise comparisons using the Mann-Whitney U test: one between PSO and SA, four between PSO and each hillclimber, and four between SA and each hillclimber.

In order to account for the multiple comparisons and control the family-wise error rate, we adopt a Bonferroni correction, which controls the significance level ($\alpha = 0.05$) by dividing it by 9, achieving an adjusted threshold of approximately 0.0056, only p-values below the threshold are considered significant.

We'll apply a two-sided Mann Whitney U test, this will determine if there's a difference and how significant It is.

For pairs that exhibit a significant difference, we will identify which algorithm is better by examining descriptive statistics, namely the medians and quartiles of their performance metrics, assuming maximisation for solution quality, such that a higher median is indicative of superiority, and minimisation for computational cost, such that a lower median is indicative of superiority.

Metrics

Solution Quality (fitness):

For each pair of algorithms A and B, for each problem instance:

- Null Hypothesis H_0: The distributions of fitness values produced by algorithms A and B are equal.
- Alternative Hypothesis H_1: The distributions of fitness values produced by algorithms A and B are different.

For solution quality, higher fitness is better.

Computational Cost (evaluations):

For each pair of algorithms A and B, for each problem instance:

- Null Hypothesis H_0: The distributions of evaluation values produced by algorithms A and B are equal.
- Alternative Hypothesis H_1: The distributions of evaluation values produced by algorithms A and B are different.

For computational cost, lower number of evaluations is better.

Particle Swarm Optimisation vs. Simulated Annealing

For the uf20 instance:

Metric	U-value	p-value	Statistical Difference?
Fitness	150	9.82×10^{-8}	Yes
Evaluations	792	2.67×10^{-7}	Yes

For both metrics, the *p*-value is below the 0.0056 threshold, therefore we reject the null hypothesis H_0, there are statistical differences between PSO and SA. For fitness, the U-value is 150, meaning PSO has smaller ranks compared to SA, therefore smaller fitness values. For evaluations, the U-value is 792, meaning PSO has higher ranks compared to SA, therefore higher evaluations values. This can be confirmed based on the descriptive statistics of each algorithms:

Algorithm	Avg Fitness	Std Deviation	Median	Q3 Quartile	Maximum
PSO	90	0.718	90	91	91
SA	91	0	91	91	91

Algorithm	Avg Evaluations	Std Deviation	Median	Q3 Quartile	Maximum
PSO	669804	467048	1000 000	1000000	1 000000
SA	1366	0	1105	2052.75	4450

For uf20, comparing Particle Swarm Optimisation vs. Simulated Annealing, we reject H_0 with 99.44% confidence, and conclude that PSO and SA differ significantly. Based on descriptive statistics, SA achieves higher fitness values and uses fewer evaluations.

For the uf250 instance:

Metric	U-value	p-value	Statistical Difference?
Fitness	0.0	1.21×10^{-11}	Yes
Evaluations	735	3.46×10^{-7}	Yes

For both metrics, we reject H_0 with 99.44% confidence, there are significant statistical differences between PSO and SA, which we can confirm based on the descriptive statistics:

Algorithm	Avg Fitness	Std Deviation	Median	Q3 Quartile	Maximum
PSO	1041	5.06	1041	1044	1049
SA	1065	0.48	1065	1065	1065

Algorithm	Avg Evaluations	Std Deviation	Median	Q3 Quartile	Maximum
PSO	1000000	0	1000000	1000000	1000000
SA	665798	314729	668558	1000000	1000000

For uf250, we reject H_0 with 99.44% confidence, and conclude that PSO and SA differ significantly. Based on descriptive statistics, SA achieves higher fitness values and uses fewer evaluations.

Particle Swarm Optimisation vs. Hillclimbers

Although we'd like to explore every pairwise comparison, this would take too much space, therefore we created a table with all the data from every Mann-Whitney U test done for PSO vs Hillclimber methods.

For uf20, fitness metric:

Algorithm	PSO: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
PSO	-------	-------	-------	90	90	91	91
NAHC	768	1.02×10^{-6}	Rejected	89	89	89	91
MS_NAHC	150	9.82×10^{-8}	Rejected	91	91	91	91
VNH	414	5.72×10^{-1}	Accepted	90	90	91	91
MS_VNH	150	9.82×10^{-8}	Rejected	91	91	91	91

PSO had no significant statistical difference from VNH, it had higher solution quality compared to NAHC but smaller compared to the multistart hillclimbers.

For uf20, evaluations metric:

Algorithm	PSO: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
PSO	-------	-------	-------	1000000	1000000	1000000	1000000
NAHC	900	1.26×10^{-11}	Rejected	52	51	61	72
MS_NAHC	836	6.33×10^{-9}	Rejected	470	414	630	1493
VNH	794	2.80×10^{-7}	Rejected	1142	1402	1541	2693
MS_VNH	717	5.94×10^{-5}	Rejected	4232	2909	5564	21521

For uf20 (evaluations), all p-values are below 0.0056, so H_0 is rejected in every comparison. PSO shows higher computational cost than all hillclimbing algorithms.

For uf100, fitness metric:

Algorithm	PSO: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
PSO	-------	-------	-------	421	422	423	425
NAHC	617	1.31×10^{-2}	Accepted	419	419	421	423
MS_NAHC	0	1.61×10^{-11}	Rejected	428	428	428	429
VNH	153	1.04×10^{-5}	Rejected	424	425	426	428
MS_VNH	14	8.10×10^{-11}	Rejected	426	426	427	428

NAHC shows no significant difference vs PSO (p = 0.0131 > 0.0056), while MS-NAHC, VNH and MS_VNH significantly outperform PSO in solution quality.

For uf100, evaluations metric:

Algorithm	PSO: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
PSO	-------	-------	-------	1000000	1000000	1000000	1000000
NAHC	900	1.26×10^{-12}	Rejected	437	436	474	676
MS_NAHC	450	6.33×10^{0}	Accepted	1000000	1000000	1000000	1000000
VNH	900	2.80×10^{-12}	Rejected	252472	237066	297619	460788
MS_VNH	450	5.94×10^{0}	Accepted	1000000	1000000	1000000	1000000

PSO differs significantly from NAHC and VNH, but not from MS-NAHC or MS-VNH ($p > 0.0056$). PSO uses far more evaluations than NAHC and VNH, but shows no statistical difference against the multistart variants.

For uf250, fitness metric:

Algorithm	PSO: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
PSO	-------	-------	-------	1041	1041	1044	1049
NAHC	391	3.82×10^{-1}	Accepted	1042	1042	1046	1050
MS_NAHC	0	2.49×10^{-11}	Rejected	1054	1054	1055	1056
VNH	5	4.60×10^{-11}	Rejected	1053	1053	1055	1059
MS_VNH	12	8.76×10^{-11}	Rejected	1052	1052	1054	1059

PSO shows no significant difference vs NAHC ($p = 0.382 > 0.0056$), but MS-NAHC, VNH, and MS-VNH all achieve higher fitness.

For uf250, evaluations metric:

Algorithm	PSO: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
PSO	-------	-------	-------	1000000	1000000	1000000	1000000
NAHC	900	1.21×10^{-12}	Rejected	1254	1236	1373	1547
MS_NAHC	450	1.0×10^{0}	Accepted	1000000	1000000	1000000	1000000
VNH	900	1.21×10^{-12}	Rejected	352382	318382	381910	601203
MS_VNH	450	1.0×10^{0}	Accepted	1000000	1000000	1000000	1000000

For uf250 (evaluations), PSO differs significantly from NAHC and VNH ($p < 0.0056$), but not from MS-NAHC or MS-VNH ($p = 1$). NAHC and VNH use fewer evaluations than PSO.

Simulated Annealing vs. Hillclimbers

For uf20, fitness metric:

Algorithm	SA: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
SA	-------	-------	-------	91	91	91	91
NAHC	885	2.92×10^{-12}	Rejected	89	89	89	91
MS_NAHC	450	1.0×10^{0}	Accepted	91	91	91	91
VNH	705	1.94×10^{-6}	Rejected	90	90	91	91
MS_VNH	450	1.0×10^{0}	Accepted	91	91	91	91

SA differs significantly from NAHC and VNH ($p < 0.0056$), both of which achieve worse fitness. No significant differences are found vs MS-NAHC or MS-VNH ($p = 1$).

For uf20, evaluations metric:

Algorithm	SA: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
SA	-------	-------	-------	1000000	1000000	1000000	1000000
NAHC	900	3.01×10^{-11}	Rejected	52	51	61	72
MS_NAHC	669	1.27×10^{-3}	Accepted	470	414	630	1493
VNH	496	5.01×10^{-1}	Accepted	1142	1402	1541	2693
MS_VNH	145	6.74×10^{-6}	Rejected	4232	2909	5564	21521

SA differs significantly from NAHC and MS-VNH ($p < 0.0056$), both of which use far fewer evaluations. No significant difference is found vs MS-NAHC or VNH ($p > 0.0056$).

For uf100, fitness metric:

Algorithm	SA: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
SA	-------	-------	-------	429.8	430	430	430
NAHC	900	6.00×10^{-12}	Rejected	419	419	421	423
MS_NAHC	897	4.38×10^{-12}	Rejected	428	428	428	429
VNH	900	5.73×10^{-12}	Rejected	424	425	426	428
MS_VNH	900	4.33×10^{-12}	Rejected	426	426	427	428

All p-values are far below 0.0056, so H_0 is rejected in every comparison. SA significantly outperforms all hillclimbing algorithms, achieving higher fitness across all runs.

For uf100, evaluations metric:

Algorithm	SA: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
SA	-------	-------	-------	340740	179716	233340	1000000
NAHC	900	1.26×10^{-11}	Rejected	437	436	474	676
MS_NAHC	90	6.33×10^{-9}	Rejected	1000000	1000000	1000000	1000000
VNH	310	2.80×10^{-2}	Accepted	252472	237066	297619	460788
MS_VNH	90	5.94×10^{-9}	Rejected	1000000	1000000	1000000	1000000

SA differs significantly from NAHC, MS-NAHC, and MS-VNH ($p < 0.0056$), but not from VNH ($p = 0.028 > 0.0056$). NAHC uses far fewer evaluations than SA, while the multistart variants use the full evaluation budget.

For uf250, fitness metric:

Algorithm	SA: U	p-value	H_0	Avg Fitness	Median	Q3	Maximum
SA	-------	-------	-------	1064.6	1065	1065	1065
NAHC	900	1.23×10^{-11}	Rejected	1042	1042	1046	1050
MS_NAHC	900	1.04×10^{-11}	Rejected	1054	1054	1055	1056
VNH	900	1.18×10^{-11}	Rejected	1053	1053	1055	1059
MS_VNH	900	1.19×10^{-11}	Rejected	1052	1052	1054	1059

All p-values are far below 0.0056, so H_0 is rejected in every comparison. All hillclimbing algorithms achieve significantly lower fitness than SA.

For uf250, evaluations metric:

Algorithm	SA: U	p-value	H_0	Avg Evals	Median	Q3	Maximum
SA	-------	-------	-------	665798	668558	1000000	1000000
NAHC	900	2.62×10^{-11}	Rejected	1254	1236	1373	1547
MS_NAHC	165	3.45×10^{-7}	Rejected	1000000	1000000	1000000	1000000
VNH	839	8.34×10^{-9}	Rejected	352382	318382	381910	601203
MS_VNH	165	3.45×10^{-7}	Rejected	1000000	1000000	1000000	1000000

All p-values are below 0.0056, H_0 is rejected in every comparison. SA uses significantly more evaluations than NAHC and VNH, and shows no improvement over the multistart variants, which consume the full evaluation budget.

Future Work and Conclusion:

Despite the limited evaluation budget, the study's results showed consistent performance disparities between the approaches, which were validated using Mann-Whitney tests at a 99.44% confidence level, offering significant statistical support for the observed patterns.

For future improvements, one promising direction is to incorporate the population-sizing method of Lobo and Harik (1999) from the Parameter-less Genetic Algorithm, which automatically adjusts the population size during the search, this could strengthen PSO. Further improvements include parameter tuning, specially for PSO. From an implementation perspective, compiling evaluation and neighbourhood functions with Numba could significantly reduce execution time.

The evaluation budget of 1 million, which was lower than ideal, limited the performance of some algorithms, the multistart VNH was the most affected, as it was unable to complete a full neighbourhood cycle requiring over 2 million evaluations, and the PSO.

References

1. CLERC, Maurice; KENNEDY, James – The particle swarm — explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 2002, vol. 6, n.º 1, p. 58–73. DOI: 10.1109/4235.985692.
2. GARCÍA-NIETO, José; ALBA, Enrique – Restart particle swarm optimization with velocity modulation: a simple and efficient algorithm for continuous optimization. Engineering Optimization, 2011, vol. 43, n.º 10, p. 979–999. DOI: 10.1080/0305215X.2010.538683.
3. HARIK, Georges R.; LOBO, Fernando G. – A parameter-less genetic algorithm. In BANZHAF, Wolfgang et al. (eds.) – Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99): Orlando, Florida, USA, 13–17 July 1999. San Francisco: Morgan Kaufmann, 1999, p. 258–265. ISBN 1-55860-611-4.