Scuttle: A System for Multi-Objective MaxSAT

Christoph Jabs
Department of Computer Science, University of Helsinki, Finland

Jeremias Berg
Department of Computer Science, University of Helsinki, Finland

Matti Järvisalo
Department of Computer Science, University of Helsinki, Finland

Abstract

We describe the Scuttle system for multi-objective combinatorial optimization. Scuttle accepts multi-objective instances where the constraints are declared either as propositional clauses or pseudo-Boolean constraints, and implements a range of multi-objective maximum satisfiability algorithms (including ones for enumerating all Pareto-optimal and leximax-optimal solutions). Pseudo-Boolean constraints are translated to clauses, allowing for applying any of the implemented algorithms on both the clausal and the pseudo-Boolean level. Scuttle also includes tightly integrated preprocessing (both core boosting and liftings of SAT preprocessing techniques) for multi-objective instances and can provide proof certificates for selected algorithms.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Constraint and logic programming

Keywords and phrases Multi-objective combinatorial optimization, maximum satisfiability, pseudo-Boolean optimization

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.37

Category Tool Paper

Supplementary Material *Software*: `https://bitbucket.org/coreo-group/scuttle`
archived at `swh:1:rel:ff1eaf27737ef8081842013b04353b56d395d849`

Funding *Christoph Jabs*: Funded by Research Council of Finland grant 356046.
Jeremias Berg: Funded by Research Council of Finland grant 362987.
Matti Järvisalo: Funded by Research Council of Finland grant 356046.

Acknowledgements The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

1 Introduction

Significant advances in maximum satisfiability (MaxSAT) [4] solvers have made MaxSAT a truly viable approach to efficiently finding provably optimal solutions to various types of real-world combinatorial optimization problem instances. Motivated by this and the fact that often real-world problems come with multiple conflicting objectives to optimize [14, Chapter 1], the challenge of lifting the success of MaxSAT solving to multi-objective combinatorial optimization, i.e., multi-objective MaxSAT (MO-MaxSAT), has received increasing attention [42, 44, 10, 11, 18, 25, 29, 26, 24].

We describe Scuttle version 1.0, a system for multi-objective combinatorial optimization. Scuttle offers implementations of a range of algorithmic approaches capturing different notions of sets of "optimal" solutions in the multi-objective setting, as well as preprocessing, optimality certification (proof logging), and various search space parameter options. In terms of flavors of multi-objective optimization, Scuttle supports both the notion of Pareto optimization [14, Chapter 2] (sometimes referred to as Pareto efficiency), and

lexicographic max-ordering optimization (or leximax optimization in short) [14, Chapter 5]. The former refers to the enumeration of one (or all) representative solution(s) to each so-called non-dominated point in the solution space and is arguably perhaps the most central optimality notion in multi-objective optimization. The latter refers to a more restricted enumeration of specific Pareto-optimal solutions exhibiting in a sense minimal trade-offs among objective values across the objectives. Specifically, for a multi-objective MaxSAT instance $(F, (O_1, \ldots, O_p))$ consisting of constraints represented as a propositional formula F in conjunctive normal form (CNF) and a set of p linear objective functions over Boolean variables, a solution α to F is Pareto-optimal [14, Chapter 2] if there is no solution β such that $O_i(\beta) \leq O_i(\alpha)$ for all $i = 1, \ldots, p$ with $O_i(\beta) < O_i(\alpha)$ for at least one i, i.e., no other solution can be better in one objective without being worse in another. A non-dominated point is the tuple of objective values of a Pareto-optimal solution and the non-dominated set, the set of all non-dominated points of an instance. On the other hand, a solution α is leximax-optimal if there is no other solution for which the list of objective values sorted in decreasing order is lexicographically smaller than the sorted list of objective values of α. Thereby the definition of leximax optimality aims for a kind of "fair" balance between the objectives.

In terms of algorithms, Scuttle offers from-scratch implementations of state-of-the-art representative approaches to MO-MaxSAT, including P-minimal [42, 34, 30], multi-objective implicit hitting set search combining SAT and integer programming [28], lower-bounding search [11], and the bi-objective algorithm BiOptSat [29], as well as different instantiations of SAT-based leximax optimization [10]. Additionally, Scuttle also provides an implementation of MipPd [43], an integer programming-based approach instantiated for MO-MaxSAT for the first time. We provide empirical data to support the claim that Scuttle offers state-of-the-art implementations of the various MO-MaxSAT algorithms. In terms of input formats, as we describe, Scuttle accepts both a natural extension of the WCNF DIMACS clausal format, with support for multiple objectives, as well as constraints expressed in the pseudo-Boolean level OPB format. To handle pseudo-Boolean constraints, Scuttle automatically translates them into clausal constraints, allowing the use of the offered algorithms to solve multi-objective pseudo-Boolean optimization instances. Certification (proof logging) in the VeriPB 3 format [17, 9] is currently provided for a selection of the algorithms as well as for the PB-to-clausal translation, allowing for verifying that the enumerated solutions are exactly the sought-after optimal solutions [24]. Finally, Scuttle offers tightly-coupled integration of both liftings of SAT preprocessing techniques [25] and the recent so-called core boosting preprocessing/reformulation approach [26] for MO-MaxSAT. The Scuttle system is implemented in Rust and is available in open source at `https://bitbucket.org/coreo-group/scuttle`.

This description focuses on the main intended use-case of Scuttle as a standalone executable. Additionally, Scuttle can also be used via a prototype Rust API; more details on how to use the API are included in the solver repository.

2 Input Instances and File Formats

Internally Scuttle works on instances consisting of a CNF formula F describing the solution space and a tuple of linear objective functions $(O_1, \ldots, O_p)$ over Boolean variables in F. Figure 1 (right) shows an example instance. As input formats, Scuttle supports both MCNF, a custom file format extending the WCNF format introduced for the MaxSAT Evaluation 2022 [6] standard to single-objective MaxSAT solvers, and an extension of the pseudo-Boolean OPB file format [41] that allows for multiple objective lines and expressing pseudo-Boolean constraints, which are internally translated to CNF.

```
h 1 6 0
h 2 7 0
h 3 8 0
h 4 9 0
o 1 1 -1 0
o 1 2 -2 0
o 1 1 -3 0
o 1 1 -4 0
o 1 1 -5 0
o 2 1 -6 0
o 2 1 -7 0
o 2 2 -8 0
o 2 1 -9 0
o 2 1 5 6 0
```

```
min: +1 x1 +2 x2 +1 x3 +1 x4 +1 x5;
min: +1 x6 +1 x7 +2 x8 +1 x9 +1 x10;
+1 x1 +1 x6 >= 1;
+1 x2 +1 x7 >= 1;
+1 x3 +1 x8 >= 1;
+1 x4 +1 x9 >= 1;
+1 x5 +1 x6 +1 x10 >= 1;
```

$$\begin{aligned}\mathcal{I} =&(F,(O_1,O_2))\\ F =&(x_1 \vee x_6) \wedge (x_2 \vee x_7)\\ &\wedge (x_3 \vee x_8) \wedge (x_4 \vee x_9)\\ &\wedge (x_5 \vee x_6 \vee x_{10})\\ O_1 =&x_1 + 2x_2 + x_3 + x_4 + x_5\\ O_2 =&x_6 + x_7 + 2x_8 + x_9 + x_{10}\end{aligned}$$

Figure 1 Left: An example MCNF file representing an instance $\mathcal{I}$. Middle: A multi-objective OPB file depicting the instance obtained after the blocking literal transformation on $\mathcal{I}$. Right: The representation of $\mathcal{I}$ that Scuttle uses internally.

The MCNF Format. An example of an MCNF file is shown in Figure 1 (left), corresponding to the instance in Figure 1 (right). Similarly to single-objective weighted partial MaxSAT, in this format each clause is either *hard* (i.e., must always be satisfied), or *soft* (incurring a cost when not satisfied). In the multi-objective setting, each soft clause needs to specify both the objective it belongs to and the cost of falsifying the clause. In the MCNF file format, hard clauses are specified in the same way as in the WCNF file format. A hard clause is a line that starts with the tag `h`, followed by space-separated non-zero integers specifying the literals of the clause (where variables are indexed starting from 1, and negative numbers represent a negated variable), and terminated by `0`. Soft clause lines extend the soft clause format in WCNF in that they are prefixed by the character `o` followed by a positive integer specifying the index of the objective the soft clause belongs to. Everything after the objective index is identical to a soft clause in WCNF, i.e., a positive integer specifying the cost of not satisfying the clause followed by the clause described in the same format as hard clauses.

When processing MCNF input, Scuttle applies the well-known blocking literal transformation [4] to soft clauses. A unit soft clause (ℓ) with weight w is converted to the term $w \cdot \neg\ell$ in the respective objective. A non-unit soft clause C of weight w is augmented with a fresh variable x_C to form the hard clause $(C \vee x_C)$ and the term $w \cdot x_C$ in the objective. For example, consider the last soft clause in Figure 1 (left) which is extended with the fresh variable x_{10} included as a term in O_2 in the instance in Figure 1 (right)

The OPB Format Extended to MO. Scuttle also supports multi-objective pseudo-Boolean optimization instances expressed in a multi-objective extension of the OPB format [41] also used as the input format in some of the other multi-objective MaxSAT solvers [10, 11]. Concretely, Scuttle supports OPB files with variable names of form `x<idx>` where `<idx>` is the index of the variable. Objectives – either minimization (`min:`) or maximization (`max:`), which is converted to minimization by negating the objective – and constraints are allowed to contain negated variables, written as `~x<idx>`. An example of a multi-objective OPB file is shown in Figure 1 (middle).

Beyond clauses, Scuttle supports arbitrary pseudo-Boolean constraints. These are converted internally to clauses using RustSAT [23]. RustSAT detects PB constraints that can be expressed as clauses also in cases where the right-hand side is not `>= 1`, such as in `-1 x1 -1 x2 >= -1`, which is equivalent to $(\neg x_1 \vee \neg x_2)$, and encodes ones that do not correspond

Table 1 Algorithms implemented in Scuttle and the tasks the individual algorithms support. The "Enumeration" column refers to support for enumerating either all optimal solutions or all ParetoMCSes. "Certificates" refers to support for generating a certificate of correctness in the VeriPB format alongside solving.

Algorithm	# Objectives	Optimality notion	Enumeration	Certificates
`p-minimal` [42]	any	Pareto	yes	yes
`bioptsat` [29]	2	Pareto	yes	yes
`lower-bounding` [11]	any	Pareto	yes	yes
`pareto-ihs` [28]	any	Pareto	no	no
`mip-pd` [43]	any	Pareto	no	no
`leximax-sat-unsat` [10]	any	Leximax	no	no
`leximax-msu3` [10]	any	Leximax	no	no

to clauses with the (generalized) totalizer encoding [5, 32]. With this pseudo-Boolean to CNF translation, Scuttle has been recently shown to be competitive with, and in some cases even outperform, multi-objective pseudo-Boolean optimizers that rely on native pseudo-Boolean reasoning [27].

3 Algorithmics

We shortly overview the algorithms and preprocessing techniques implemented in Scuttle.

3.1 Solving Algorithms

Scuttle implements a collection of algorithms for different multi-objective optimization tasks. The algorithm to use is selected via the command line interface of Scuttle; see Section 5. Table 1 gives an overview of the offered algorithms and the multi-objective optimization tasks the algorithms support. Scuttle also integrates its additional features, such as solving pseudo-Boolean input instances via translation and preprocessing, to each reimplementation. Further, `p-minimal`, `bioptsat` and `lower-bounding` can produce certificates (see Section 6) and support enumerating either all Pareto-optimal solutions or all so-called ParetoMCSes [44].[1]

For all algorithms that iteratively employ a SAT solver as an oracle (i.e., all except `mip-pd`), Scuttle currently uses CaDiCaL v2.2.1 [8].[2] PB constraints over the objectives are encoded to CNF via the (generalized) totalizer encoding [5, 32] as implemented in RustSAT [23]. Additionally, the encoding structure is incrementally reused when changing bounds [37].

***P*-minimal.** Scuttle provides a reimplementation of the solution-improving `p-minimal` algorithm [42, 34] based on enumeration of so-called P-minimal models. (The approach has been independently also referred to as the guided-improvement algorithm [30].) `p-minimal`

[1] In short, in our terminology, a ParetoMCS is a set of objective literals for which it holds that a solution that assigns precisely these literals to 1 is Pareto-optimal. Viewing the assignment of an objective literal to 1 as removing some constraints, ParetoMCSes represent minimal sets of constraints whose removal from the instance leaves the remainder satisfiable. Each ParetoMCS corresponds to a non-dominated point; a non-dominated point may correspond to several ParetoMCSes.

[2] Scuttle uses CaDiCaL via its extended IPASIR interface and the proof tracer interface for proof logging. No custom modifications to CaDiCaL are necessary.

incrementally finds solutions and adds exclusion constraints to the SAT oracle to block these solutions, and any solutions dominated by them. Additionally, temporary constraints (realized via the SAT solver's assumption interface) are used to heuristically guide the search. Without enumeration, the exclusion constraints are directly added to the SAT oracle. In order to facilitate enumeration, the exclusion constraints are first enforced via reversible assumptions, and subsequently made permanent only after enumeration is completed.

BiOptSat. Scuttle includes a reimplementation of the SAT-UNSAT variant – using solution-improving search for both objectives – of the `bioptsat` [29] algorithm. `bioptsat` supports bi-objective optimization following the lexicographic method. First, one objective is minimized. Then, fixing the value of the first-minimized objective to its optimal value, the other objective is minimized. By doing so, a non-dominated point is found and `bioptsat` proceeds by adding a constraint requiring the second objective to be better than the just-found conditional optimum, and iterating to find the next point.

Lower Bounding. Selectable as `lower-bounding`, Scuttle includes a reimplementation of the lower-bounding search algorithm from [11]. The `lower-bounding` algorithm executes the `p-minimal` algorithm within heuristically set upper bounds on the objectives. The bounds are incrementally increased until the full search space has been explored. In addition to being a from-scratch reimplementation of `p-minimal`, the Scuttle implementation differs from the original in the SAT oracle employed, the pseudo-Boolean encoding used for objective constraints, and the ability to produce certificates.

ParetoIHS. Scuttle includes the first implementation of the recent `pareto-ihs` [28] algorithm, extending the implicit hitting set approach to MO-MaxSAT. `pareto-ihs` performs single-objective implicit hitting set search on a linear combination of the objectives. Whenever an optimal solution is found, a so-called PD cut is enforced in the hitting set solver to block it and all solutions it dominates. As the hitting set solver, Scuttle supports the fully open-source HiGHS [19] mixed-integer programming (MIP) solver and Gurobi as a potentially more performant (but commercial) option. The encoding of PD cuts varies slightly depending on the hitting set solver used: the reification variables required are implemented with either the indicator constraint API (in Gurobi), or the big-M method (in HiGHS).

MipPd. Scuttle provides (to the best of our knowledge) the first implementation of the `mip-pd` algorithm described in [43] applied to MO-MaxSAT. `mip-pd` is the only implemented algorithm that does not make use of a SAT oracle. `mip-pd` optimizes a linear combination of the objectives by employing a MIP solver while iteratively blocking optimal solutions with Pareto dominance cuts. As the underlying MIP solver, HiGHS [19] and Gurobi can be used.

Leximax. Scuttle version 1.0 also includes from-scratch reimplementations of two leximax optimization algorithms from [10]. The first algorithm, `leximax-sat-unsat`, follows the solution-improving paradigm where a sequence of iteratively improving (in the leximax sense) solutions are found, while the second algorithm, `leximax-msu3`, finds a leximax optimum with iterative core-guided search based on the MSU3 MaxSAT algorithm [36]. The Scuttle implementation of these algorithms differs from the original in the SAT oracle used and the pseudo-Boolean encodings employed.

3.2 Preprocessing

Scuttle offers tightly-coupled preprocessing techniques in conjunction with the implemented algorithms in the form of core boosting and liftings of SAT preprocessing techniques.

Core Boosting. Core boosting [26] is an MO-MaxSAT preprocessing technique that shrinks the search space by reformulating each individual objective with the help of a single-objective core-guided MaxSAT algorithm. It has been found to speed-up solution-improving style solving algorithms in particular [26]. In Scuttle, core boosting is implemented for all algorithms except for `mip-pd` and turned on by default for all except for `pareto-ihs`. (In contrast to the other algorithms, core boosting has not been found to speed up `pareto-ihs` [28].) Concretely, Scuttle performs core boosting by executing its own implementation of the OLL [38] single-objective MaxSAT algorithm with stratification [2] individually for each objective. Towards finding a minimum-cost solution for a single-objective MaxSAT instance, the OLL algorithm iteratively relaxes inconsistencies (unsatisfiable cores) of the instance and reformulates the objective to obtain a smaller value range and higher constant term. The instance reformulation is implemented using the totalizer [5] encoding. An MO-MaxSAT solver of choice is then executed on the reformulated instance. Core boosting is tightly integrated with the solving algorithms. In more detail, firstly, the SAT oracle used during core boosting is reused for solving, preserving lemmas and simplifications learned during core boosting. Secondly, for algorithms employing pseudo-Boolean constraint encodings over the objectives, the totalizer structures built during OLL are reused as parts of the (generalized) totalizer encodings employed in the solving algorithm. Lastly, for `pareto-ihs`, several modes of adding information about the instance reformulation built by core boosting into the hitting set solver can be selected.

Lifted SAT Preprocessing Techniques. The MaxSAT preprocessor MaxPre (version 2) [33] has been recently extended with support for multi-objective MaxSAT [25]. MaxPre implements preprocessing techniques lifted from SAT (e.g., bounded variable elimination [13] and blocked clause elimination [31]) as well as MaxSAT-specific techniques such as subsumed label elimination [7]. Scuttle is compiled with multi-objective MaxPre support by specifying `--features maxpre` at compilation time. With MaxPre preprocessing enabled, the input instance is automatically processed by MaxPre and the solutions found to the preprocessed instance reconstructed before reporting the solutions to the user.

4 Performance of Scuttle Compared to Earlier Implementations

As a short overview of the performance of the solving algorithms as implemented in Scuttle, Figure 2 shows a pairwise comparison between the algorithms implemented in Scuttle (with their default settings) and previous implementations of the same algorithms. Complementing the pairwise comparisons, Figure 3 shows the number of solved instances over time for each of the algorithm implementations. The experiments were run with a 1-hour time and 32-GB memory limit on machines with Intel Xeon Gold 6248 CPUs and 377 GB of RAM. As the instance set, we used the 320 instances under 2–7 objective (120 of which have two objectives) used most recently for evaluating the `pareto-ihs` algorithm [28]. In short, the reimplementations that Scuttle provides tend to outperform the earlier counterparts in all cases.

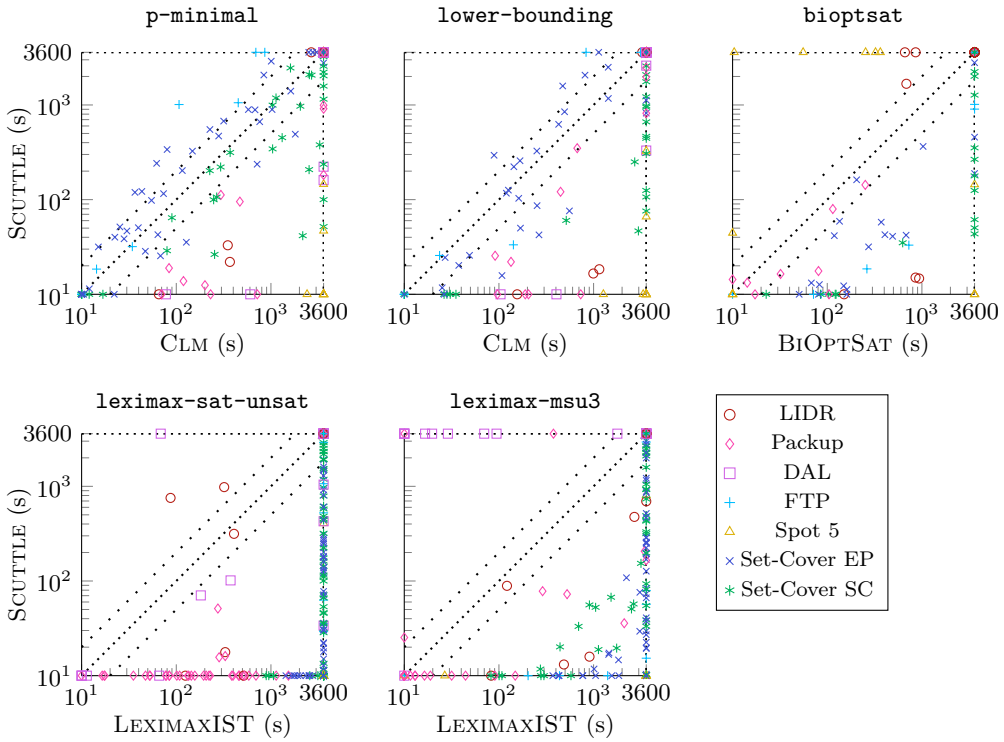

Figure 2 Pairwise comparisons of algorithms implemented in Scuttle with previous implementations of the same algorithms in terms of solving time. The previous implementations are Clm [11], BiOptSat [29], and LexiMaxIST [10]. The legend refers to benchmark domains as described in [28].

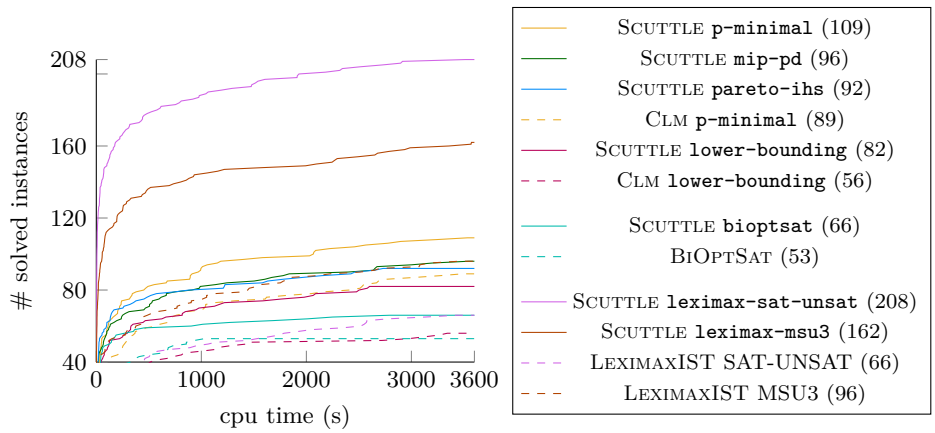

Figure 3 Number of solved instances over time of all algorithms implemented in Scuttle and previously available implementations.

5 Command Line Options

Executing Scuttle with `scuttle <algorithm> <instance>` will run the selected algorithm with reasonable default options. Here we cover a selection of additional command line options Scuttle exposes for controlling algorithm execution. A full list of options for each algorithm (including algorithm-specific options) is available via `scuttle <algorithm> --help`.

Enumeration options. By default, Scuttle does not perform enumeration of solutions at each non-dominated or leximax-optimal point. With the option `--enumeration` enumeration of either all Pareto-optimal solutions (`--enumeration=solutions`) or ParetoMCSes (`--enumeration=pareto-mcs`) can be enabled for the `p-minimal`, `lower-bounding`, and `bioptsat` algorithms.

Limiting search. Scuttle provides options for limiting search in various ways, causing the solver to terminate once a limit is reached and therefore making the search incomplete. This can be useful in applications where available computing resources are insufficient for computing the entire non-dominated set, while e.g. a subset of k non-dominated points can still be of interest (`--non-dominated-point-limit=k`). More options for limiting search are described in the "Solver limits" section of the help output. In addition to limits set via command line options, Scuttle can be terminated via the POSIX `SIGTERM` signal, which allows for enforcing resource limits, e.g., via the coreutils `timeout` command, enabling anytime usage of the solver.

Preprocessing options. Core boosting and MaxPre-based preprocessing are controlled via the Boolean `--core-boosting` and `--maxpre-preprocessing` flags, respectively, which can be set to either `true` or `false`. For core boosting, core minimization and exhaustion can additionally be enabled via the `--oll-core-minimization` or `--oll-core-exhaustion` options. For MaxPre, the exact techniques used can be specified via `--maxpre-techniques`, with the default being `[[uvsrgc]VRTG]` which includes unit propagation, bounded variable elimination [13], subsumption elimination, self-subsuming resolution [39, 13], group subsumed label elimination [7], binary core removal [16, 33], TrimMaxSAT [40], failed literal elimination [45, 15, 35], and intrinsic at-most-one constraint extraction [20, 21].

ParetoIHS-specific options. Specific options only apply to `pareto-ihs`. The MIP solver is selected via the `--hitting-set-solver` command line option. The algorithm itself has two configuration parameters: the strategy for choosing scalarization constants can be selected via the `--scalarization-constants` option, and `--precompute-lexicographic` controls how many lexicographic optima are precomputed before the main algorithm. Related to single-objective IHS refinements, the `--seeding` flag controls constraint seeding [12] into the hitting set solver, `--ihs-core-extraction` controls the strategy in which cores are extracted each iteration, and `--reduced-cost-fixing` controls reduced cost fixing [3]. Lastly, `--ihs-cb-treatment` controls the integration of core boosting into `pareto-ihs`.

Output options. The `--verbose` (`-v`) option increases logging verbosity during search and can be repeated to increase verbosity further. With the `--log-non-dominated` option, logging of non-dominated points as they are found can be enabled. Similar options are available for candidate solutions encountered during search and optimal solutions. With the `--print-solutions` option, the binary assignment of all optimal solutions found is printed as a `v` line in the format used in the MaxSAT evaluation [6]. All other options related to solver output are described in the "Output options" section of the help output.

6 Certificates

Scuttle supports generating a machine-checkable certificate (proof log) in the VeriPB 3 format [17, 9] as a witness for the facts that all discovered solutions are correct solutions, that all discovered solutions that are not dominated by another discovered solution are

Pareto-optimal, and that at termination, for each non-dominated point at least one solution has been discovered [24]. Support for generating such certificates is currently implemented for `p-minimal`, `bioptsat`, and `lower-bounding`. To produce a certificate, Scuttle is invoked with an additional parameter `scuttle <algorithm> <instance> [proof]` where `[proof]` specifies the path to write the VeriPB proof log to.

While the VeriPB proof system can be used to certify that at least one solution per non-dominated point was enumerated, the VeriPB checker does not currently support parsing multi-objective input files. To circumvent this limitation and check the proofs generated by Scuttle, the VeriPB checker currently needs to be supplied with an input file that contains only the constraints (after the blocking literal transformation) of the input instance. The objectives are baked into the partial order that is loaded at the beginning of the proof itself. For OPB input files, the required constraint-only file is obtained by simply removing all objective lines (starting with `min:`) from the instance file. For generating a similar input file when working with MCNF instances, Scuttle is supplied with an additional input parameter. Scuttle will then output an OPB instance file containing all constraints after the blocking literal transformation. For this use-case, Scuttle is invoked as `scuttle <algorithm> <instance> [proof] [opb-constr]`, followed by the VeriPB checker as `veripb <opb-constr> <proof>`.

For OPB input files with non-clausal PB constraints, the encoding of pseudo-Boolean constraints to clauses that Scuttle generates internally is also certified in the generated proof following [27]. This allows for end-to-end verification from the multi-objective OPB input file to the computed result.

7 Conclusions

We presented Scuttle, a system for multi-objective combinatorial optimization via MO-MaxSAT. Scuttle supports both propositional and pseudo-Boolean constraint level input, offers state-of-the-art implementations of a range of MO-MaxSAT solving algorithms for Pareto and leximax optimization with tightly integrated preprocessing, as well correctness certification (proof logging) for selected algorithms. Enabling VeriPB certificates for `pareto-ihs` and the leximax optimization algorithms – building on recent advances in certifying IHS [22] and the VeriPB proof system [1] – as well as stable C and Python APIs remain as future work.

References

1 Markus Anders, Bart Bogaerts, Benjamin Bogø, Arthur Gontier, Wietze Koops, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, Adrian Rebola-Pardo, and Yong Kiam Tan. Faster certified symmetry breaking using orders with auxiliary variables. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor, editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20–27, 2026*, pages 14140–14148. AAAI Press, 2026. doi:10.1609/aaai.v40i17.38426.

2 Carlos Ansótegui, Maria Luisa Bonet, Joel Gabàs, and Jordi Levy. Improving SAT-based weighted MaxSAT solvers. In Michela Milano, editor, *Principles and Practice of Constraint Programming – 18th International Conference, CP 2012, Québec City, QC, Canada, October 8–12, 2012. Proceedings*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2012. doi:10.1007/978-3-642-33558-7_9.

3 Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing for maximum satisfiability. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden*, pages 5209–5213. ijcai.org, 2018. doi:10.24963/ijcai.2018/723.

4 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 929–991. IOS Press, 2021. doi:10.3233/FAIA201008.

5 Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming – CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 – October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003. doi:10.1007/978-3-540-45193-8_8.

6 Jeremias Berg, Matti Järvisalo, Ruben Martins, Andreas Niskanen, and Tobias Paxian. MaxSAT Evaluation 2024: Rules. https://maxsat-evaluations.github.io/2024/rules.html.

7 Jeremias Berg, Paul Saikko, and Matti Järvisalo. Subsumed label elimination for maximum satisfiability. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 – 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands – Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 630–638. IOS Press, 2016. doi:10.3233/978-1-61499-672-9-630.

8 Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification – 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part I*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024. doi:10.1007/978-3-031-65627-9_7.

9 Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *Journal of Artificial Intelligence Research*, 77:1539–1589, 2023. doi:10.1613/jair.1.14296.

10 Miguel Cabral, Mikolás Janota, and Vasco Manquinho. SAT-based leximax optimisation algorithms. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, Haifa, Israel, August 2–5, 2022*, volume 236 of *LIPIcs*, pages 29:1–29:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.29.

11 João Cortes, Inês Lynce, and Vasco Manquinho. New core-guided and hitting set algorithms for multi-objective combinatorial optimization. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22–27, 2023, Proceedings, Part II*, volume 13994 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2023. doi:10.1007/978-3-031-30820-8_7.

12 Jessica Davies and Fahiem Bacchus. Exploiting the power of MIP solvers in MAXSAT. In Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability Testing – SAT 2013 – 16th International Conference, Helsinki, Finland, July 8–12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013. doi:10.1007/978-3-642-39071-5_13.

13 Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In Fahiem Bacchus and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19–23, 2005, Proceedings*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005. doi:10.1007/11499107_5.

14 Matthias Ehrgott. *Multicriteria Optimization (2. ed.)*. Springer, 2005. doi:10.1007/3-540-27659-9.

15 Jon William Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, USA, 1995.

16 James F. Gimpel. A reduction technique for prime implicant tables. In *5th Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, New Jersey, USA, November 11–13, 1964*, pages 183–191. IEEE Computer Society, 1964. doi:10.1109/SWCT.1964.4.

17 Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-Boolean proofs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pages 3768–3777. AAAI Press, 2021. doi:10.1609/aaai.v35i5.16494.

18 Andreia P. Guerreiro, João Cortes, Daniel Vanderpooten, Cristina Bazgan, Inês Lynce, Vasco Manquinho, and José Rui Figueira. Exact and approximate determination of the Pareto front using minimal correction subsets. *Computers & Operations Research*, 153:106153, 2023. doi:10.1016/j.cor.2023.106153.

19 Qi Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10:119–142, 2018. doi:10.1007/s12532-017-0130-5.

20 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11:53–64, 2019. doi:10.3233/SAT190116.

21 Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning – 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8–10, 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 75–94. Springer, 2022. doi:10.1007/978-3-031-10769-6_6.

22 Hannes Ihalainen, Dieter Vandesande, André Schidler, Jeremias Berg, Bart Bogaerts, and Matti Järvisalo. Efficient and reliable hitting-set computations for the implicit hitting set approach. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor, editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20–27, 2026*, pages 14251–14260. AAAI Press, 2026. doi:10.1609/aaai.v40i17.38439.

23 Christoph Jabs. RustSAT: A library for SAT solving in rust. In Jeremias Berg and Jakob Nordström, editors, *28th International Conference on Theory and Applications of Satisfiability Testing, SAT 2025, Glasgow, Scotland, August 12–15, 2025*, volume 341 of *LIPIcs*, pages 15:1–15:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SAT.2025.15.

24 Christoph Jabs, Jeremias Berg, Bart Bogaerts, and Matti Järvisalo. Certifying pareto optimality in multi-objective maximum satisfiability. In Arie Gurfinkel and Marijn Heule, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 31st International Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3–8, 2025, Proceedings, Part II*, volume 15697 of *Lecture Notes in Computer Science*, pages 108–129. Springer, 2025. doi:10.1007/978-3-031-90653-4_6.

25 Christoph Jabs, Jeremias Berg, Hannes Ihalainen, and Matti Järvisalo. Preprocessing in SAT-based multi-objective combinatorial optimization. In Roland H. C. Yap, editor, *29th International Conference on Principles and Practice of Constraint Programming, CP 2023, Toronto, Canada, August 27–31, 2023*, volume 280 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CP.2023.18.

26 Christoph Jabs, Jeremias Berg, and Matti Järvisalo. Core boosting in SAT-based multi-objective optimization. In Bistra Dilkina, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research – 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28–31, 2024, Proceedings, Part II*, volume 14743 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2024. doi:10.1007/978-3-031-60599-4_1.

27 Christoph Jabs, Jeremias Berg, and Matti Järvisalo. Engineering and evaluating multi-objective pseudo-Boolean optimizers. In Giovanni Casini, Besik Dundua, and Temur Kutsia, editors, *Logics in Artificial Intelligence – 19th European Conference, JELIA 2025, Kutaisi, Georgia, September 1–4, 2025, Proceedings, Part I*, volume 16093 of *Lecture Notes in Computer Science*, pages 115–134. Springer, 2025. doi:10.1007/978-3-032-04587-4_8.

28 Christoph Jabs, Jeremias Berg, and Matti Järvisalo. Multi-objective maximum satisfiability by single-objective implicit hitting set optimization. In Tias Guns, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research—23rd International Conference, CPAIOR 2026, Rabat, Morocco, May 26–29, 2026*, volume 16595 of *Lecture Notes in Computer Science*. Springer, 2026. URL: https://media.christophjabs.info/papers/JabsEtAl2026MultiObjectiveMaximumSatisfiability.pdf.

29 Christoph Jabs, Jeremias Berg, Andreas Niskanen, and Matti Järvisalo. From single-objective to bi-objective maximum satisfiability solving. *Journal of Artificial Intelligence Research*, 80:1223–1269, 2024. doi:10.1613/jair.1.15333.

30 Daniel Jackson, H.-Christian Estler, and Derek Rayside. The guided improvement algorithm for exact, general-purpose, many-objective combinatorial optimization. Technical report, Compute Science and Artificial Intelligence Laboratory, 2009. URL: http://hdl.handle.net/1721.1/46322.

31 Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20–28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2010. doi:10.1007/978-3-642-12002-2_10.

32 Saurabh Joshi, Ruben Martins, and Vasco Manquinho. Generalized totalizer encoding for pseudo-Boolean constraints. In Gilles Pesant, editor, *Principles and Practice of Constraint Programming – 21st International Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 2015. doi:10.1007/978-3-319-23219-5_15.

33 Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. MaxPre: An extended MaxSAT preprocessor. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017 – 20th International Conference, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 449–456. Springer, 2017. doi:10.1007/978-3-319-66263-3_28.

34 Miyuki Koshimura, Hidetomo Nabeshima, Hiroshi Fujita, and Ryuzo Hasegawa. Minimal model generation with respect to an atom set. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Proceedings of the 7th International Workshop on First-Order Theorem Proving, FTP 2009, Oslo, Norway, July 6–7, 2009*, volume 556 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL: https://ceur-ws.org/Vol-556/paper06.pdf.

35 Daniel Le Berre. Exploiting the real power of unit propagation lookahead. *Electronic Notes in Discrete Mathematics*, 9:59–80, 2001. doi:10.1016/S1571-0653(04)00314-2.

36 João Marques-Silva and Jordi Planes. On using unsatisfiability for solving maximum satisfiability. *Computing Research Repository*, abs/0712.1097, 2007. arXiv:0712.1097.

37 Ruben Martins, Saurabh Joshi, Vasco Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming – 20th International Conference, CP 2014, Lyon, France, September 8–12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2014. doi:10.1007/978-3-319-10428-7_39.

38 António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming – 20th International Conference, CP 2014, Lyon, France, September 8–12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014. doi:10.1007/978-3-319-10428-7_41.

39 Richard Ostrowski, Éric Grégoire, Bertrand Mazure, and Lakhdar Sais. Recovering and exploiting structural knowledge from CNF formulas. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming – CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9–13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2002. doi:10.1007/3-540-46135-3_13.

40 Tobias Paxian, Pascal Raiola, and Bernd Becker. On preprocessing for weighted MaxSAT. In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *Verification, Model Checking, and Abstract Interpretation – 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17–19, 2021, Proceedings*, volume 12597 of *Lecture Notes in Computer Science*, pages 556–577. Springer, 2021. doi:10.1007/978-3-030-67067-2_25.

41 Olivier Roussel. General OPB format. https://www.cril.univ-artois.fr/PB24/OPBgeneral.pdf.

42 Takehide Soh, Mutsunori Banbara, Naoyuki Tamura, and Daniel Le Berre. Solving multiobjective discrete optimization problems with propositional minimal model generation. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming – 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 596–614. Springer, 2017. doi:10.1007/978-3-319-66158-2_38.

43 John Sylva and Alejandro Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158:46–55, 2004. doi:10.1016/S0377-2217(03)00255-8.

44 Miguel Terra-Neves, Inês Lynce, and Vasco Manquinho. Multi-objective optimization through Pareto minimal correction subsets. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden*, pages 5379–5383. ijcai.org, 2018. doi:10.24963/ijcai.2018/757.

45 Ramin Zabih and David A. McAllester. A rearrangement search strategy for determining propositional satisfiability. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence, St. Paul, MN, USA, August 21–26, 1988*, pages 155–160. AAAI Press / The MIT Press, 1988. URL: http://www.aaai.org/Library/AAAI/1988/aaai88-028.php.