Chapter 1

Introduction

As soon as an Analytical Engine exists,
it will necessarily guide the future course of the science.
Whenever any result is sought by its aid,
the question will then arise —
But what course of calculation can these
results be arrived at by the machine
in the shortest time?
Charles Babbage (1864)

1.1 Context

Since Charles Babbage, many computer scientists have been asking themselves the same question: *may I find a way to make the computer solve a problem in shorter time*? In the process of pursuing such a goal, several important computational problems have been identified as core problems in Computer Science. Among them it stands out the satisfiability problem (SAT), which is the problem of deciding if there exists a truth assignment that satisfies a propositional formula in conjunctive normal form (CNF formula). Such a problem is classified among combinatorial problems, which commonly imply finding values to a set of variables which are restricted by a set of constraints; in some cases the aim is to find a solution satisfying all the constraints (satisfaction problems), in other cases the aim is to find a solution satisfying as many constraints as possible (optimization problems).

In this thesis, we focus on an optimization version of SAT: the Maximum Satisfiability problem (MAX-SAT). This problem consists of finding a truth assignment that satisfies the maximum number of clauses in a CNF formula. We will see in the sequel that the MAX-SAT algorithms actually solve the equivalent problem of finding a truth assignment that falsifies the minimum number of clauses in a CNF formula, since this carries implementation benefits. Sometimes, we also consider a variant of MAX-SAT, called weighted MAX-SAT. In

weighted MAX-SAT, every clause has a weight and the problem consists of finding a truth assignment in which the sum of weights of violated clauses is minimal. While SAT is NP-complete [Coo71], both MAX-SAT and weighted MAX-SAT are NP-hard [GJ79].

We started our research on MAX-SAT in 2002, when SAT was —as it is nowadays— a central topic in Artificial Intelligence and Theoretical Computer Science. At that time, there were publicly available complete solvers such as Chaff [MMZ+01], GRASP [MSS99], Posit [Fre95], Relsat [BS97], and Satz [LA97a, LA97b], as well as local search solvers such as GSAT and WalkSAT [SK93, SKC94, SLM92]. There was also enough empirical evidence about the merits of the generic problem solving approach which consists of modeling NP-complete decision problems as SAT instances, solving the resulting encodings with a state-of-the-art SAT solver, and mapping the solution back into the original problem. This generic problem solving approach was competitive in a variety of domains, including hardware verification [MSG99, MMZ+01, VB03], quasigroup completion [AGKS00, KRA+01], planning [KS96, Kau06], and scheduling [BM00].

Despite the remarkable activity on SAT, there was a reduced number of papers dealing with the design and implementation of exact MAX-SAT solvers, and solving NP-hard problems by reducing them to MAX-SAT was not considered a suitable alternative for solving optimization problems. This is in contrast with what happened in the Constraint Programming community, where the Weighted Constraint Satisfaction Problem (Weighted CSP) was a problem attracting the interest of that community, which published a considerable amount of results about weighted CSP [MRS06].

We decided to start our research by designing and implementing branch and bound MAX-SAT solvers in the style of the exact solvers developed by Wallace and Freuder [WF96], and Borchers and Furman [BF99], which can be seen as an adaptation to MAX-SAT of the Davis-Logemann-Loveland (DLL) procedure [DLL62]. We thought that we could produce good performing MAX-SAT solvers by adapting to MAX-SAT the technology incorporated into the existing DLL-style SAT solvers, which were equipped with optimized data structures, clever variable selection heuristics, clause learning, non-chronological backtracking, randomization and restarts.

Before going into more technical details, let us introduce how works a basic branch and bound (BnB) algorithm for MAX-SAT: BnB explores the search space induced by all the possible truth assignments in a depth-first manner. At each node of the search tree, BnB compares the number of clauses falsified by the best complete assignment found so far —called *Upper Bound* (UB)— with the *Lower Bound* (LB), which is the number of clauses falsified by the current partial assignment plus an *underestimation* of the number of clauses that would become unsatisfied if the current partial assignment is extended to a complete assignment. Obviously, if $UB \leq LB$, a better assignment cannot be found from this point in the search, and BnB prunes the subtree below the current node and backtracks to a higher level in the search tree. If $UB > LB$, the current partial

assignment is extended by instantiating one more variable; which leads to create two branches from the current branch: the left branch corresponds to instantiate the new variable to false, and the right branch corresponds to instantiate the new variable to true. The solution to MAX-SAT is the value that UB takes after exploring the entire search tree.

At first sight, we observe two differences between BnB and DLL: unit propagation is not applied since it is unsound for MAX-SAT, and a lower bound has to be updated at each node of the search tree. On the one hand, the fact that unit propagation does not preserve the number of unsatisfied clauses led us to study new forms of inference for MAX-SAT. On the other hand, the fact of having to compute a lower bound at each node of the search tree led us to improve the existing lower bound computation methods.

Unit propagation is unsound for MAX-SAT, in the sense that the number of clauses falsified by an assignment is not preserved between a CNF formula ϕ and the formula obtained after applying unit propagation to ϕ. For example, if we apply unit propagation to $\phi = \{p_1, \neg p_1 \vee \neg p_2, \neg p_1 \vee p_2, \neg p_1 \vee \neg p_3, \neg p_1 \vee p_3\}$, we get two unsatisfied clauses. While assigning all the variables to false falsifies exactly one clause of ϕ. Therefore, if unit propagation is applied in BnB, non-optimal solutions can be obtained.

We devoted a part of this thesis to define sound and efficiently applied resolution rules for MAX-SAT that are, in a sense, the MAX-SAT counterpart of unit resolution. Let us see an example of inference rule: Given a MAX-SAT instance ϕ that contains three clauses of the form $l_1, l_2, \bar{l}_1 \vee \bar{l}_2$, where l_1, l_2 are literals, replace ϕ with the CNF formula

$$\phi' = (\phi \setminus \{l_1, l_2, \bar{l}_1 \vee \bar{l}_2\}) \cup \{\square, l_1 \vee l_2\}.$$

Note that the rule detects a contradiction from $l_1, l_2, \bar{l}_1 \vee \bar{l}_2$ and, therefore, replaces these clauses with an empty clause $\square$. In addition, the rule adds the clause $l_1 \vee l_2$ to ensure the equivalence between ϕ and ϕ'. An assignment that falsifies l_1 and l_2 then falsifies 2 of those 3 clauses, while any other asignment falsifies exactly 1 clause. The last clause added by the rule captures such a state.

The inference rules contribute by deriving new empty clauses, but it is also important to improve the underestimation of the number of clauses that will become unsatisfied if the current partial assignment is completed. Basically, when we started our research, the underestimations defined for MAX-SAT were based on counting the number of complementary unit clauses in the CNF formula under consideration.

We realized that a suitable underestimation is provided by the number of disjoint unsatisfiable subformulas which can be computed with an efficient procedure. In a first step [LMP05], we defined a powerful lower bound computation method based on detecting disjoint unsatisfiable subformulas by applying unit propagation. Once a contradiction is detected by unit propagation, we derive a unit resolution refutation, and the clauses involved in that refutation are taken as an unsatisfiable subformula. We repeat that process until we are not able to detect further unsatisfiable subformulas. We showed that this method, which

can be applied in time linear in the length of the CNF formula ($\mathcal{O}(ub \times |\phi|)$, where ub is an upper bound of the minimum number of unsatisfied clauses in $|\phi|$, and ϕ is the length of ϕ, leads to lower bounds of good quality. In a second step [LMP06], we enhanced our method by also applying failed literal detection. This way, we can detect resolution refutations (not necessarily unit resolution refutations) which can be computed efficiently and are beyond the reach of unit propagation.

Interestingly, we showed that inference techniques used in SAT cannot be applied to MAX-SAT because they can produce non-optimal solutions, but we applied such techniques to dramatically improve the computation of underestimations for lower bounds.

Besides defining original inference rules and good quality lower bounds, a constant concern of our work was to pay special attention to implementation issues and, in particular, to the definition of suitable data structures that allow one to perform as efficiently as possible the more common operations. Therefore, we defined optimized data structures for implementing the application of the inference rules and lower bounds, and investigated the use of very basic lazy data structures for implementing a simple and fast weighted MAX-SAT solver.

In the course of the thesis, we implemented four MAX-SAT solvers: AMP, Lazy, UP, and MaxSatz. The more sophisticated solver is MaxSatz, which incorporates the main contributions of our research. Further details about these solvers and the ideas behind can be found in the remaining chapters. At this point, we just would like to mention that MaxSatz was the best performing solver on all the sets of MAX-SAT instances that were solved in the *First MAX-SAT Evaluation*, a co-located event of the Ninth International Conference on Theory and Applications of Satisfiability Testing (SAT-2006).

1.2 Objectives

The general objective of our research is the design, implementation and evaluation of exact algorithms for MAX-SAT with the ultimate goal of improving, and converting into a suitable alternative for solving optimization problems, the generic problem solving approach consisting of modeling optimization problems as MAX-SAT instances, solving the resulting encodings with a MAX-SAT solver, and mapping the solution back to the original problem.

The particular objectives of the thesis can be summarized as follows:

- Define underestimations of good quality for lower bound computation methods that allow one to prune as soon as possible the parts of the search space that do not contain any optimal solution.

- Define sound resolution-style inference rules for MAX-SAT whose application allows one to derive empty clauses as early as possible during the exploration of the search space.

- Design algorithms that combine the computation of underestimations and the application of inference rules in such a way that both operations can be done efficiently and accelerate the search for an optimal solution.

- Design and implement MAX-SAT solvers, equipped with optimized data structures and clever variable selection heuristics, that incorporate the MAX-SAT solving techniques we defined.

- Conduct an empirical evaluation of the techniques developed in this thesis, and in particular of the underestimations and inference rules devised. Identifying their strengths and weaknesses should allow us to gain new insights for developing more powerful MAX-SAT solving techniques.

- Conduct an empirical comparison between our solvers and the best performing state-of-the-art MAX-SAT solvers. Knowing the performance profile of our *competitors* can help improve the performance of our solvers.

- Identify MAX-SAT techniques that can be naturally extended to weighted MAX-SAT, and analyze the implementation issues that should be reconsidered to develop fast weighted MAX-SAT solvers.

1.3 Contributions

The main contributions of this thesis can be summarized as follows:

- We defined lower bound computation methods that detect disjoint unsatisfiable subformulas by applying unit propagation, and subsume most of the existing MAX-SAT lower bounds. As a result, we defined three new lower bounds: UP, UP^S, and UP^*. The main difference among them is the data structures that implement for storing unit clauses. These data structures have an impact on the number of unit clauses of the MAX-SAT instance under consideration that are used in the derivation of unit resolution refutations.

- We enhanced the previous lower bounds with failed literal detection, giving rise to three additional lower bounds: UP_{FL}, UP^S_{FL}, and UP^*_{FL}.

- We defined a set of original inference rules for MAX-SAT which are sound and can be applied efficiently. All of them can be seen as unit resolution refinements adapted to MAX-SAT.

- We have analyzed the time complexity of the lower bounds and the inference rules proposed. Most of them can be applied in time linear in the size of the CNF formula. In particular, we defined an algorithm that combines the application of powerful lower bounds and inference rules in time $\mathcal{O}(ub \times |\phi|)$, where ub is an upper bound of the minimum number of unsatisfied clauses and $|\phi|$ is the size of the CNF formula ϕ.

- We have extended some of the previous lower bounds and inference rules to weighted MAX-SAT.
- We designed and implemented four MAX-SAT solvers:

  AMP : This was our first solver. It extends the solver of Borchers and Furman [BF99] by incorporating a different variable selection heuristic and a more powerful underestimation.

  Lazy : This was our second solver. It deals with very simple lazy data structures that allow to perform quickly the most common operations. It is based on a static variable selection heuristic and solves both MAX-SAT and weighted MAX-SAT instances.

  UP : This was our third solver. The main implementation issues were adaptations to MAX-SAT of the technology implemented in the SAT solver Satz. The most powerful component of UP is the detection of disjoint unsatisfiable subformulas by unit propagation.

  MaxSatz : This was our last solver. It applies both the underestimations and the inference rules defined in this thesis.
- We conducted an empirical evaluation of the underestimations and inference rules developed in the thesis. We observed that UP^*_{FL} is usually the best performing underestimation for the testbed used and, in combination with our inference rules, gives rise to the best performance profile.
- We conducted an empirical comparison between our solvers and the best performing state-of-the-art MAX-SAT solvers. We observed that MaxSatz is extremely competitive and outperforms the rest of MAX-SAT solvers up to several orders of magnitude on a significant number of instances.

1.4 Publications

Some of the results presented in this thesis have already been published in journals and conference proceedings. The articles are chronologically listed and classified according to the solver used on it:

AMP

- Teresa Alsinet, Felip Manyà, and Jordi Planes. Improved branch and bound algorithms for Max-2-SAT and weighted Max-2-SAT. In *Proceedings of the 6th Catalan Conference on Artificial Intelligence (CCIA 2003)*, volume 100 of *Frontiers in Artificial Intelligence and Applications*, pages 435–442, P. Mallorca, Spain, 2003. IOS Press.
- Teresa Alsinet, Felip Manyà, and Jordi Planes. Improved branch and bound algorithms for Max-SAT. In *Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing (SAT 2003)*, pages 408–415, Portofino, Italy, 2003.

– Jordi Planes. Improved branch and bound algorithms for Max-2-SAT and weighted Max-2-SAT. In Francesca Rossi, editor, *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of LNCS, page 991, Kinsale, Ireland, 2003. Springer.

Lazy

– Teresa Alsinet, Felip Manyà, and Jordi Planes. A Max-SAT solver with lazy data structures. In *Proceedings of the 9th Ibero-American Conference on Artificial Intelligence (IBERAMIA 2004)*, volume 3315 of LNAI, pages 334–342, Puebla, Mexico, 2004. Springer.

– Teresa Alsinet, Felip Manyà, and Jordi Planes. A Max-SAT solver with lazy data structures. In Le Thi Hoai An and Pham Dinh Tao, editors, *Modeling, Computation and Optimization in Information Systems and Management Sciences (MCO 2004)*, pages 491–498, Metz, France, 2004. Hermes publishing.

– Teresa Alsinet, Felip Manyà, and Jordi Planes. Improved exact solver for weighted Max-SAT. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT 2005)*, volume 3569 of LNCS, pages 371–377, St. Andrews, Scotland, 2005. Springer.

UP

– Chu Min Li, Felip Manyà, and Jordi Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In Peter van Beek, editor, *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, volume 3609 of LNCS, pages 403–414, Sitges, Spain, 2005. Springer.

MaxSatz

– Chu Min Li, Felip Manyà, and Jordi Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*, pages 86–91, Boston/MA, USA, 2006. AAAI Press.

After defending the doctoral dissertation, we published two journal articles containing its main contributions:

– Chu Min Li, Felip Manyà, and Jordi Planes. New Inference Rules for Max-SAT. In *Journal of Artificial Intelligence Research*, volume 30, pages 321–359, 2007.

– Teresa Alsinet, Felip Manyà, and Jordi Planes. An Efficient Solver for Weighted Max-SAT. In *Journal of Global Optimization*, volume 41, pages 61–73, 2008.

1.5 Overview

This section provides an overview of the thesis. We briefly describe the contents of each of the remaining chapters:

Chapter 2: Algorithms for SAT and MAX-SAT. In this chapter we provide an overview of techniques used in SAT and MAX-SAT solving. First, some basic concepts commonly used in satisfiability solving are introduced. Second, different techniques for SAT solving such as the DP algorithm and the DLL algorithm, recent efficient techniques in complete SAT solving, as well as representative local search algorithms are described. Third, the branch and bound algorithm to solve MAX-SAT is also introduced, with special attention to the techniques in lower bounds, inference rules and variables selection heuristics for MAX-SAT.

Chapter 3: Lower bounds. In this chapter we focus on computing lower bounds, the forecasting techniques for MAX-SAT. First, we review some state-of-the-art lower bounds. Second, we introduce the star rule, which is our first original lower bound computation method. Third, we define three original lower bounds that detect disjoint inconsistent subformulas by applying unit propagation. Fourth, we improve the previous lower bounds by adding failed literal detection. Finally, we report on the empirical evaluation of our lower bound computation methods.

Chapter 4: Inference rules. In this chapter we define a set of unit resolution refinements for MAX-SAT, describe an efficient way of implementing the application of the rules at each node of the search tree, and report on an experimental evaluation that provides empirical evidence that our rules can speed up a MAX-SAT solver several orders of magnitude.

Chapter 5: Implementing a weighted MAX-SAT solver. In this chapter we define a lower bound and a set of inference rules for weighted MAX-SAT, that are extensions of the MAX-SAT ones. We describe their implementation in an algorithm with a static variable selection heuristic and lazy data structures, and report on an experimental evaluation that provides empirical evidence of the good performance of the rules.

Chapter 6: Empirical comparison of MAX-SAT and weighted MAX-SAT solvers. In this chapter we first describe the best performing state-of-the-art MAX-SAT solvers, and the solvers we have designed and implemented in this thesis. Then, we report on an experimental comparison that provides empirical evidence that our solvers outperform the rest of the solvers in most of the solved instances.

Chapter 7: Conclusions. We briefly summarize the main contributions of the thesis, and point out some open problems and future research directions that we plan to tackle in the near future.

Finally, this thesis has an appendix, with enhanced inference rules that are provided as future work.