A MaxSAT-Based Approach to the Team Composition Problem in a Classroom

Felip Manyà[1(✉)], Santiago Negrete[1,2], Carme Roig[3], and Joan Ramon Soler[1]

[1] Artificial Intelligence Research Institute (IIIA, CSIC), Campus UAB, 08193 Bellaterra, Spain
felip@iiia.csic.es
[2] Universidad Autónoma Metropolitana (DCCD, Cuajimalpa), Mexico City, Mexico
[3] INS Torres i Bages, Hospitalet de Llobregat, Barcelona, Spain

Abstract. Given a classroom containing a fixed number of students and a fixed number of tables that can be of different sizes, as well as a list of preferred classmates to sit with for each student, the team composition problem in a classroom (TCPC) is the problem of finding an assignment of students to tables in such a way that preferences are maximally-satisfied. In this paper, we formally define the TCPC, prove that it is NP-hard and define a MaxSAT model of the problem. Moreover, we report on the results of an empirical investigation that show that solving the TCPC with MaxSAT solvers is a promising approach.

1 Introduction

Given a classroom containing a fixed number of students and a fixed number of tables that can be of different sizes, as well as a list of preferred classmates to sit with for each student, the team composition problem in a classroom (TCPC) is the problem of finding an assignment of students to tables in such a way that preferences are maximally-satisfied. Our motivation behind this work is to solve a problem posed by the director of studies of a secondary school in the area of Barcelona, though this problem may be found in a wide range of situations and institutions.

In this paper, we formally define the TCPC, prove that it is NP-hard and define a MaxSAT model of the problem. Moreover, we report on the results of an empirical investigation that show that solving the TCPC with MaxSAT solvers is a promising approach.

To tackle the TCPC we use a MaxSAT-based problem solving approach, which is an active area of research in Artificial Intelligence, (see e.g. [2,5,7–12,15–17,20,21] and the references therein for previous and related work). MaxSAT-based problem solving is a generic problem solving approach for optimization problems which consists on first defining a MaxSAT model for instances of the problem to be solved, and then derive solutions to the encoded instances of the problem using an off-the-shelf MaxSAT solver. By a MaxSAT model we mean a representation of the problem using the language of Boolean propositional logic. It is a declarative approach: we only need to define a model and from that model an optimal solution is automatically derived. Furthermore, the method is highly efficient because we may take advantage of the extremely efficient MaxSAT solvers which are publicly available.

It is commonly assumed that designing an algorithm to work directly on the original problem encoding should outperform approaches that require a translation via a generic intermediate formalism, such as a CSP, SAT or MaxSAT. However, this line of reasoning ignores the fact that generic solvers can benefit from many years of development by a broad research community. It is not easy to replicate this kind of effort in other domains.

In the present formulation of the problem, we consider the preferences of the students. Nevertheless, our approach could also be easily adapted to take into account other factors that can be relevant to the performance of a team such as personality, expertise, competence, competitiveness and human formation [4,6].

The rest of the paper is organized as follows: Sect. 2 defines the TCPC formally and proves that it is NP-hard. Section 3 gives some background on MaxSAT. Section 4 defines a MaxSAT model of the TCPC. Section 5 reports on the empirical investigation conducted. Section 6 gives some conclusions and future work.

2 The Team Composition Problem in a Classroom

Depending on the activity to be performed in a classroom at a given moment, the distribution of the students may need to be different. In the general case, we consider there is a fixed number of students and there is a list of preferred classmates to sit with for each student. Then, the goal is to partition students into teams, which may have different sizes, in such a way that the preferences of the students are maximally-satisfied.

The version of the TCPC that we use as a case study in this paper has the following constraints:

- The classroom has n students.
- The classroom has tables of 2 and 3 students with a combined capacity for n students.
- Each student has provided a list of classmates she would prefer to sit with.

The objective is to find an assignment of students to tables such that preferences are maximally-satisfied. Notice that the first two constraints are hard whereas the last one is soft. We will say that a solution is *fully-satisfied* if, and only if, all the students in the same table have the rest of the students of the table in their list of preferences. We will say that a solution is *maximally-satisfied* if, and only if, the number of students who have their preferences satisfied is maximized. Note that a fully-satisfied solution is also a maximally-satisfied solution.

Proposition 1. *Given n students, a classroom that has tables of 2 and 3 students with a combined capacity for n students, and a list of preferred classmates* *to sit with for each student, the problem of deciding if there is a fully-satisfied solution is NP-complete.*

Proof. This problem belongs to NP: we can check, in polynomial time, whether or not an assignment of students to tables is a fully-satisfied solution by inspecting the lists of preferences of the students.

We now prove that this problem is NP-hard by reducing the problem of partitioning a graph into triangles to it.

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, that verifies that $|V| = 3q$ for some integer q, the partition of V into triangles consists on finding a partition of V formed by $V_1, \ldots, V_q$, each containing exactly 3 vertices, such that for each $V_i = \{u_i, v_i.w_i\}$, $1 \leq i \leq q$, the edges $\{u_i, v_i\}$, $\{u_i, w_i\}$ and $\{v_i, w_i\}$ belong to E. This problem is NP-complete [14].

That problem can be reduced to an instance of our problem without loss of generality by considering a classroom with $3q$ students, 0 tables of 2 and q tables of 3. For each edge $\{u, v\}$ on graph V, establish a preference of student u for student v and a preference of student v for student u. Note that this reduction takes polynomial time. Then, the problem of partitioning the vertices of a graph into triangles has a solution if, and only if, all the students in the classroom can be sat in such a way that all students preferences are fully-satisfied. □

Corollary 1. *The TCPC is NP-hard.*

Proof. This follows from the fact that every fully-satisfied solution is also a maximally-satisfied solution.

3 The MaxSAT Problem

We assume readers have some familiarity with basic concepts of Boolean propositional logic. The most well-know problem of propositional logic is SAT: given a formula ϕ in Conjunctive Normal Form (CNF), decide whether there is a truth assignment that satisfies ϕ.

Reminder: a literal is a propositional variable or a negated propositional variable, a clause is a disjunction of literals, a CNF formula is a conjunction of clauses, and a truth assignment is a mapping that assigns 0 (false) or 1 (true) to each propositional variable. A CNF is satisfied by an assignment if it is true under the usual truth-functional interpretation of $\vee$ and $\wedge$ and the truth-values assigned to the variables.

An optimization variant of SAT is MaxSAT: given a CNF formula ϕ, MaxSAT consists of finding a truth assignment that maximizes the number of satisfied clauses of ϕ. However, in this paper we use the term MaxSAT in a broad sense: we allow to distinguish between hard and soft clauses, and allow to associate a weight to soft clauses (formally, hard clauses have an infinite weight). This more general formulation of MaxSAT is technically known as weighted partial MaxSAT [15], which is formally defined in the remaining of this section.

We start by defining a more general notion of clause. A weighted clause is a pair (c, w), where c is a clause and w, its weight, is a positive integer or infinity. A clause is hard if its weight is infinity; otherwise it is soft.

A weighted partial MaxSAT instance is a multiset of weighted clauses

$$\phi = \{(h_1, \infty), \ldots, (h_k, \infty), (c_1, w_1), \ldots, (c_m, w_m)\},$$

where the first k clauses are hard and the last m clauses are soft. For simplicity, in what follows, we omit infinity weights, and write $\phi = \{h_1, \ldots, h_k, (c_1, w_1), \ldots, (c_m, w_m)\}$. A soft clause (c, w) is equivalent to having w copies of the clause $(c, 1)$, and $\{(c, w_1), (c, w_2)\}$ is equivalent to $(c, w_1 + w_2)$.

Weighted partial MaxSAT for an instance ϕ is the problem of finding an assignment that satisfies all the hard clauses and minimizes the sum of the weights of the falsified soft clauses; such an assignment is called optimal assignment.

4 The MaxSAT Encoding

We show how the TCPC can be represented as a weighted partial MaxSAT instance. In other words, we show how to model the TCPC in the weighted partial MaxSAT formalism. To illustrate how to model the problem, we will consider that the classroom has 28 students and there are 8 tables of 2 students and 4 tables of 3 students. This is a typical classroom distribution in secondary schools of the area of Barcelona.

First of all, we define the set of Boolean variables of our encoding:

$$\{x_{ij} | 1 \leq i < j \leq 28\} \cup \{x_{ijk} | 1 \leq i < j < k \leq 28\} \cup \{y_i | 1 \leq i \leq 28\}$$

These variables have the following intended meaning: x_{ij} is true iff students i and j sit together in a table of 2; x_{ijk} is true iff students i, j and k sit together in a table of 3; and y_i is true if student i sits in a table of 2 and is false if student i sits in a table of 3.

Using the previous Boolean variables, we create a Weighted Partial MaxSAT instance that encodes the constraints of the problem. The proposed encoding has the following hard clauses:

1. For each student i, where $1 \leq i \leq 28$, the encoding contains a set of hard clauses that encode the following cardinality constraint:
   (a) If $i = 1$, then

$$\sum_{j=2}^{28} x_{1j} + \sum_{j=2}^{27} \sum_{k=j+1}^{28} x_{1jk} = 1$$

   (b) If $2 \leq i \leq 27$, then

$$\sum_{j=1}^{i-1} x_{ji} + \sum_{j=i+1}^{28} x_{ij} + \sum_{j=1}^{i-1} \sum_{k=i+1}^{28} x_{jik} + \sum_{j=i+1}^{27} \sum_{k=j+1}^{28} x_{ijk} = 1$$

(c) If $i = 28$, then

$$\sum_{j=1}^{27} x_{j28} + \sum_{j=1}^{26} \sum_{k=j+1}^{27} x_{jk28} = 1$$

This cardinality constraint states that student i sits exactly in one table, and the table is either of 2 or of 3.

2. For each variable x_{ij}, the encoding contains the hard clauses $\neg x_{ij} \vee y_i$ and $\neg x_{ij} \vee y_j$. Note that $(\neg x_{ij} \vee y_i) \wedge (\neg x_{ij} \vee y_j)$ is equivalent to $x_{ij} \rightarrow y_i \wedge y_j$. This clause states that if x_{ij} is true, then students i and j sit in a table of 2.
3. For each variable x_{ijk}, the encoding contains the hard clauses $\neg x_{ijk} \vee \neg y_i$, $\neg x_{ijk} \vee \neg y_j$ and $\neg x_{ijk} \vee \neg y_k$. Note that $(\neg x_{ijk} \vee \neg y_i) \wedge (\neg x_{ijk} \vee \neg y_j) \wedge (\neg x_{ijk} \vee \neg y_k)$ is equivalent to $x_{ijk} \rightarrow \neg y_i \wedge \neg y_j \wedge \neg y_k$. This clause states that if x_{ijk} is true, then students i, j and k sit in a table of 3.
4. The encoding contains a set of hard clauses that encode the following cardinality constraints: $\sum_{i=1}^{28} y_i = 16$ and $\sum_{i=1}^{28} \neg y_i = 12$. These cardinality constraints state that there are 16 students sitting in tables of 2 and 12 students sitting in tables of 3.
In practice, it is sufficient to add either the constraint $\sum_{i=1}^{28} y_i = 16$ or the constraint $\sum_{i=1}^{28} \neg y_i = 12$ because if there are exactly 16 (12) variables y_i, $1 \leq i \leq 28$, that evaluate to true (false), then the remaining 12 (16) variables must evaluate to false.

The encoding of a cardinality constraint of the form $x_1 + \ldots + x_n = k$ has $\mathcal{O}(n)$ clauses if one uses the encoding based on counters and defined in [22]. Other efficient encodings of cardinality constraints are described and analyzed in [1]. In our empirical investigation, we encode the previous cardinality constraints using PBLib[1], which is a C++ tool for efficiently encoding pseudo-Boolean constraints to CNF.

Since we considered two sizes of tables, we just need one variable y_i for each student. If we consider n different sizes, then we need $\lceil \log_2 n \rceil$ variables for each student. For example, for four different sizes, we need two variables (y_i, y'_i) and each size is represented by one of the following conjunctions: $y_i \wedge y'_i$, $\neg y_i \wedge y'_i$, $y_i \wedge \neg y'_i$ and $\neg y_i \wedge \neg y'_i$.
The soft clauses of our encoding are the following weighted unit clauses:

1. For each variable x_{ij}, $1 \leq i < j \leq 28$, the encoding contains the weighted unit clause (x_{ij}, w_{ij}).
2. For each variable x_{ijk}, $1 \leq i < j < k \leq 28$, the encoding contains the weighted unit clause (x_{ijk}, w_{ijk}).

Let us explain how weights are assigned to the variables of the form x_{ij} and x_{ijk}. First of all, we build a directed graph $G = (V, E)$, where V contains a vertex i for each student i in the classroom, and E contains an edge (i, j) if student i wants to seat with student j. The weight associated with each student i in G,

[1] http://tools.computational-logic.org/content/pblib.php.

denoted by $w(i)$, is the out-degree of the vertex i of G.[2] The weight associated with the variable x_{ij}, denoted by w_{ij}, is $2(w(i) \times w(j))$, where $w(i)$ and $w(j)$ are the weights associated with vertices i and j, respectively, in the subgraph of G induced by the set of vertices $\{i, j\}$ (i.e.; the weight of student i and j in $G(\{i, j\})$). The weight associated with the variable x_{ijk}, denoted by w_{ijk}, is $3(w(i) \times w(j) \times w(k)/8)$, where $w(i)$, $w(j)$ and $w(k)$ are the weights associated with vertices i, j and k, respectively, in $G(\{i, j, k\})$. The value of $w(i) \times w(j)$ ranges from 0 to 1 and the value of $w(i) \times w(j) \times w(k)$ ranges from 0 to 8. This explains the fact that $w(i) \times w(j) \times w(k)$ is divided by 8. Moreover, we multiply the weights by 2 in the tables of 2 and by 3 in the tables of 3. In this way, we maximize the number of satisfied students.[3]

In the previous encoding, if the weight associated with a variable is 0, then the negation of this variable is added as a unit clause in the hard part. Moreover, an optimal solution corresponds to a fully-satisfied solution iff all the satisfied soft clauses of the form (x_{ij}, w_{ij}) and (x_{ijk}, w_{ijk}) have weight 2 and 3, respectively.

Observe that, for fully-satisfied instances, if we add to the hard part the negation of x_{ij} (i.e., the unit hard clause $\neg x_{ij}$) for each variable x_{ij} whose associated weight is different from 2 and the negation of x_{ijk} (i.e., the unit hard clause $\neg x_{ijk}$) for each variable x_{ijk} whose associated weight is different from 3, then we do not need to add any soft clause. Moreover, any satisfying assignment of the hard part allows us to derive a fully-satisfied solution. This case can be solved either with a SAT solver or with a MaxSAT solver fed with a MaxSAT instance that only contains hard clauses.

If there is no fully-satisfied solution, the objective is to find a solution that satisfies students as much as possible. Because of that, in the general case, we add the clauses (x_{ij}, w_{ij}) and (x_{ijk}, w_{ijk}) such that $w_{ij} \neq 0$ and $w_{ijk} \neq 0$ in the soft part of the encoding. In this way, we provide a solution that maximizes the number of satisfied students. In this case, we say that we have a maximally-satisfied solution.

Finally, it is worth mentioning that it is possible to define a MaxSAT encoding of the TCPC using the set of propositional variables $\{x_i^t | 1 \leq i \leq 28, 1 \leq t \leq 12\}$, where the intended meaning of x_i^t is that x_i^t is true iff student i sits at table t. However, all the experiments performed with encodings using this set of variables did not outperform the experiments performed with the encoding proposed in this section.

5 Experimental Results

We conducted an empirical investigation to assess how the MaxSAT-based approach to the TCPC works in practice on fully-satisfied instances. In the experiments, in order to analyze the scaling behavior, we considered different sizes of

[2] The out-degree of a vertex is the number of edges going out of a vertex in a directed graph.

[3] Since most of the MaxSAT solvers deal with weights that are positive integers, in the experiments we multiply the weights by 100 and take the integer part.

classrooms: the rows always have 2 tables of 2 and 1 table of 3, and the numbers of rows ranges from 1 to 20. So, the numbers of students per classroom ranges from 7 to 140. Besides, we assumed that each student gives a list of students she would like to sit with. We generated the preferences at random in such a way that we can guarantee that the generated instances have fully-satisfied solutions. We generated 50 different TCPC instances for each size of classroom, encoded them to weighted partial MaxSAT, and solved the resulting encodings with the exact MaxSAT solver WPM3 [7] using a cutoff time of 1 h. All the experiments were performed in a 2.3 GHz Intel PC with 1 GB RAM. The results obtained are shown in Tables 1 and 2.

Table 1 shows the results for the encoding that only contains hard clauses. Besides the hard clauses of Sect. 4, we add the unit hard clause $\neg x_{ij}$ for each variable x_{ij} whose associated weight is different from 2 and the unit hard clause

Table 1. Experimental results for the encoding without soft clauses: Students: number of students; Clauses: mean number of clauses per instance; Variables: mean number of variables per instance; and Time: mean time, in seconds, needed to solve an instance. The number of solved instances, within a cutoff time of 3600 s, is shown in parentheses.

Students	Clauses	Variables	Time
7	178	96	0.01 (50)
14	947	607	0.01 (50)
21	2675	1857	0.01 (50)
28	5888	4260	0.01 (50)
35	10841	8155	0.01 (50)
42	18036	13732	0.01 (50)
49	27685	21381	0.01 (50)
56	40282	31508	0.02 (50)
63	56130	44490	0.02 (50)
70	75187	60606	0.04 (50)
77	98640	80288	0.05 (50)
84	126205	103807	0.08 (50)
91	158597	131484	0.12 (50)
98	195963	163685	0.16 (50)
105	239214	200667	0.21 (50)
112	288402	242780	0.27 (50)
119	343816	290432	0.61 (50)
126	405198	343887	1.23 (50)
133	475062	403623	1.94 (50)
140	551134	469835	2.73 (50)

$\neg x_{ijk}$ for each variable x_{ijk} whose associated weight is different from 3. Table 2 shows the results for the encoding that has the hard clauses of the previous encoding but also the soft clauses of the form $(x_{ij}, 2)$ and $(x_{ijk}, 3)$.

Table 2. Experimental results for the encoding with soft clauses: Students: number of students; Clauses: mean number of clauses per instance; Soft clauses: mean number of soft clauses per instance; Variables: mean number of variables per instance; and Time: mean time, in seconds, needed to solve an instance. The number of solved instances, within a cutoff time of 3600s, is shown in parentheses.

Students	Clauses	Soft clauses	Variables	Time
7	178	7	96	0.01 (50)
14	947	18	607	0.01 (50)
21	2675	32	1857	0.01 (50)
28	5888	51	4260	0.05 (50)
35	10841	75	8155	3.26 (50)
42	18036	100	13732	346 (49)
49	27685	133	21381	1273 (10)

The empirical results show that the encoding without soft constraints finds optimal solutions quickly and scales well in practice. However, the encoding with soft constraints only finds optimal solutions quickly when the number of students is not greater than 35. In summary, the results show that MaxSAT allows one to find fully-satisfied solutions quickly using suitable encodings. For the TCPC, it is decisive to use efficient encodings for cardinality constraints.

6 Concluding Remarks

We have developed a method to encode the TCPC as a weighted partial MaxSAT problem, proved its NP-hardness, and carried out experiments to evaluate our approach using an exact MaxSAT solver. The results show that our method is useful because it does not need a dedicated algorithm; it is declarative, hence all stakeholders can be involved and understand the way the problem is specified; it is flexible because different classroom configurations can be solved with it; and it is efficient because it provides optimal solution in a reasonable amount of time. In the future, we plan to conduct a more exhaustive empirical investigation, model the problem using MinSAT [18,19] instead of MaxSAT, and explore the possibility of using our method to encode similar team composition problems. In practice, our method could be combined with profiling techniques [13] to solve the group formation problem in *Computer Supported Collaborative Learning* applications. Our contributions could also be applied to other projects have taken a different approach to solve related problems using other AI techniques (see [3,4,6] and the references therein for further details).

Acknowledgements. This work was supported by the Generalitat de Catalunya grant AGAUR 2014-SGR-118, and the MINECO-FEDER project RASO TIN2015-71799-C2-1-P. The second author is supported by grant PSPA-PNB-CGVyDI.324.2016 from Universidad Autónoma Metropolitana (Cuajimalpa), Mexico.

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: A parametric approach for smaller and better encodings of cardinality constraints. In: Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming, CP, Uppsala, Sweden, pp. 80–96 (2013)
2. Abramé, A., Habet, D.: On the resiliency of unit propagation to max-resolution. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, Buenos Aires, Argentina, pp. 268–274 (2015)
3. Alberola, J.M., del Val, E., Sánchez-Anguix, V., Palomares, A., Teruel, M.D.: An artificial intelligence tool for heterogeneous team formation in the classroom. Knowl. Based Syst. **101**, 1–14 (2016)
4. Alberola, J.M., del Val, E., Sanchez-Anguix, V., Julian, V.: Simulating a collective intelligence approach to student team formation. In: Pan, J.-S., Polycarpou, M.M., Woźniak, M., de Carvalho, A.C.P.L.F., Quintián, H., Corchado, E. (eds.) HAIS 2013. LNCS (LNAI), vol. 8073, pp. 161–170. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40846-5_17
5. Alsinet, T., Manyà, F., Planes, J.: An efficient solver for weighted Max-SAT. J. Glob. Optim. **41**, 61–73 (2008)
6. Andrejczuk, E., Rodríguez-Aguilar, J.A., Roig, C., Sierra, C.: Synergistic team composition. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS, São Paulo, Brazil, pp. 1463–1465 (2017)
7. Ansótegui, C., Didier, F., Gabàs, J.: Exploiting the structure of unsatisfiable cores in MaxSAT. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI, Buenos Aires, Argentina, pp. 283–289 (2015)
8. Ansótegui, C., Gabàs, J., Malitsky, Y., Sellmann, M.: MaxSAT by improved instance-specific algorithm configuration. Artif. Intell. **235**, 26–39 (2016)
9. Argelich, J., Li, C.M., Manyà, F., Planes, J.: The first and second Max-SAT evaluations. J. Satisfiability Boolean Model. Comput. **4**, 251–278 (2008)
10. Argelich, J., Li, C.M., Manyà, F., Planes, J.: Analyzing the instances of the MaxSAT evaluation. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 360–361. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21581-0_29
11. Argelich, J., Manyà, F.: Exact Max-SAT solvers for over-constrained problems. J. Heuristics **12**(4–5), 375–392 (2006)
12. Bonet, M.L., Levy, J., Manyà, F.: A complete calculus for Max-SAT. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 240–251. Springer, Heidelberg (2006). https://doi.org/10.1007/11814948_24
13. Costaguta, R.: Algorithms and machine learning techniques in collaborative group formation. In: Lagunas, O.P., Alcántara, O.H., Figueroa, G.A. (eds.) MICAI 2015. LNCS (LNAI), vol. 9414, pp. 249–258. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27101-9_18
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)

15. Li, C.M., Manyà, F.: MaxSAT, hard and soft constraints. In: Biere, A., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, pp. 613–631. IOS Press (2009)
16. Li, C.M., Manyà, F., Planes, J.: New inference rules for Max-SAT. J. Artif. Intell. Res. **30**, 321–359 (2007)
17. Li, C.M., Manyà, F., Soler, J.R.: A clause tableaux calculus for MaxSAT. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI-2016, New York City, NY, USA, pp. 766–772 (2016)
18. Li, C.M., Zhu, Z., Manyà, F., Simon, L.: Minimum satisfiability and its applications. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, Barcelona, Spain, pp. 605–610 (2011)
19. Li, C.M., Zhu, Z., Manyà, F., Simon, L.: Optimizing with minimum satisfiability. In: Artificial Intelligence, vol. 190, pp. 32–44 (2012)
20. Martins, R., Joshi, S., Manquinho, V., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: O'Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 531–548. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_39
21. Morgado, A., Heras, F., Liffiton, M.H., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: a survey and assessment. Constraints **18**(4), 478–534 (2013)
22. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 827–831. Springer, Heidelberg (2005). https://doi.org/10.1007/11564751_73