Extended Conjunctive Normal Form and An Efficient Algorithm for Cardinality Constraints

Zhendong Lei[1,2] , **Shaowei Cai**[1,2*] and **Chuan Luo**[3]
[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, China
[3]Microsoft Research, China
{leizd, caisw}@ios.ac.cn

Abstract

Satisfiability (SAT) and Maximum Satisfiability (MaxSAT) are two basic and important constraint problems with many important applications. SAT and MaxSAT are expressed in CNF, which is difficult to deal with cardinality constraints. In this paper, we introduce Extended Conjunctive Normal Form (ECNF), which expresses cardinality constraints straightforward and does not need auxiliary variables or clauses. Then, we develop a simple and efficient local search solver *LS-ECNF* with a well designed scoring function under ECNF. We also develop a generalized Unit Propagation (UP) based algorithm to generate the initial solution for local search. We encode instances from Nurse Rostering and Discrete Tomography Problems into CNF with three different cardinality constraint encodings and ECNF respectively. Experimental results show that *LS-ECNF* has much better performance than state of the art MaxSAT, SAT, Pseudo-Boolean and ILP solvers, which indicates solving cardinality constraints with ECNF is promising.

1 Introduction

Satisfiability problem (SAT) has gained a considerable attention with the advent of a new generation of solvers able to solve large instances from real world problems. Maximum Satisfiability (MaxSAT) is an optimization version of SAT which also achieves great success thanks to the techniques used in SAT. These impressive progress led to more and more combinatorial optimization problems in real world situations encoded into these propositional logic languages and solved by applying a suitable SAT or MaxSAT solver.

Given a propositional formula expressed in the Conjunctive Normal Form (CNF), SAT is concerned with satisfying all clauses, while MaxSAT is concerned with satisfying as many clauses as possible. Partial MaxSAT (PMS) is a generalization of MaxSAT, whose clauses are divided into hard and weighted soft clauses and the goal is to find an assignment that satisfies all the hard clauses and maximize the total weight of satisfied soft clauses. There are many efficient PMS solvers, including SAT-based solvers [Narodytska and Bacchus, 2014; Martins *et al.*, 2014; Berg *et al.*, 2019; Nadel, 2019; Joshi *et al.*, 2019; Demirovic and Stuckey, 2019] and local search solvers [Cai *et al.*, 2016; Luo *et al.*, 2017; Cai *et al.*, 2017; Lei and Cai, 2018; Guerreiro *et al.*, 2019].

One of the most important drawbacks of these logical languages is the difficulty to deal with cardinality constraints. Indeed, cardinality constraints arise frequently in the encoding of many real world situations such as scheduling, logic synthesis or verification, product configuration and data mining. For the above reasons, many works have been done on finding an efficient encoding of cardinality constraints in CNF formulas [Sinz, 2005; Asín *et al.*, 2009; Hattad *et al.*, 2017; Boudane *et al.*, 2018; Karpinski and Piotrów, 2019].

Even though modern SAT and MaxSAT solvers are powerful and the cardinality constraint encodings have also been improved, MaxSAT and SAT solvers still do not perform well when solving problems with cardinality constraints. Many encoding methods have been proposed, but their performance vary among different domains. Indeed, experiment results from a few problems [Demirovic and Musliu, 2014; Demirovic *et al.*, 2019] indicate that none of these encoding methods are obviously dominant.

In this work, we propose another solution to deal with cardinality constraints. Instead of finding a better encoding method, we propose Extended Conjunctive Normal Form (ECNF), which is an extension of CNF. In SAT(MaxSAT) of CNF, a clause is satisfied if it has at least one true literal, while in SAT(MaxSAT) of ECNF, a clause is satisfied if it contains a certain amount of true literals. Encoding cardinality constraints into ECNF is natural and simple without introducing any additional variable or clause. The most important thing is that techniques used in SAT or MaxSAT solvers can be easily adapted to this model with minor modifications.

We define the ECNF versions of SAT and MaxSAT problems. Then we develop a local search solver *LS-ECNF* for solving SAT and MaxSAT of ECNF formulas. A new scoring function is used in *LS-ECNF* to adapt to the new model. We also design a generalized Unit Propagation (UP) based algorithm to get an initial solution for *LS-ECNF*. To study the effectiveness of our method, we encode instances from Nurse Rostering and Discrete Tomography Problems into MaxSAT(SAT) of CNF with three different cardinal-

*Corresponding author

ity constraint encodings respectively. We also encode these instances into ECNF for comparison. Experimental results show that *LS-ECNF* has great improvement in the encoding efficiency, memory usage, speed and quality over the state of the art MaxSAT, SAT, Pseudo-Boolean and even Integer Linear Programming (ILP) solvers.

The remainder of this paper is structured as follows. Section 2 introduces preliminary knowledge. Section 3 presents the definition of ECNF and encoding rules from cardinality constraints to ECNF. Section 4 presents the local search and generalized UP-based algorithms. Experimental results are presented in Section 5. Finally, we conclude the paper.

2 Preliminary

Given a set of n Boolean $variables$ $\{x_1, x_2, ..., x_m\}$, a $literal$ is either a variable x_i or its negation $\neg x_i$. A $clause$ is a disjunction of literals which can be represented as a set of literals. A conjunctive normal form (CNF) formula $F = C_1 \wedge C_2 \wedge ... \wedge C_m$ is a conjunction of clauses which can be represented as a multiset of clauses. A complete truth assignment is a mapping that assigns to each variable either 0 or 1.

In CNF, a clause is satisfied if it has at least one true literal, and falsified otherwise. SAT problem is concerned with satisfying all clauses, while MaxSAT problem is concerned with satisfying as many clauses as possible. Given a CNF formula, the Partial MaxSAT (PMS) problem, in which some clauses are declared to be hard and the rest are declared to be soft, is the problem of finding an assignment such that all hard clauses are satisfied and maximizes the total number of satisfied soft clauses. In Weighted PMS (WPMS), each soft clause is associated with a positive integer as its weight, and the goal is to satisfy all hard clauses and maximize the total weight of satisfied soft clauses.

Note that, if the problem contains only hard (resp. soft) constraints, the obtained problem is SAT (resp. MaxSAT). If the problem contains both hard and soft constrains, the resulting problem is Partial MaxSAT.

A *unit clause* is a clause containing only one literal. The unit propagation technique is quite simple: For a given CNF formula, we collect all unit clauses in it, and then assume that variables are set to satisfy these unit clauses. That is, if the unit clause $\{v_i\}$ appears in the formula, we set v_i to true; and if the unit clause $\{\neg v_i\}$ appears in the formula, we set v_i to false. We then condition the formula on these settings. The iterative application of this rule until no more unit clauses remain is called *unit propagation* (UP).

2.1 Cardinality Constraints

Cardinality constraints impose limits on the truth values assigned to literals. These are $atLeast\_k[x_i : x_i \in X]$, $atMost\_k[x_i : x_i \in X]$ and $exactly\_k[x_i : x_i \in X]$, which constraint that at least, at most and exactly k literals out of the specified ones must or may be assigned to true. For clarity and consistency, and as the constraints $atMost\_k[x_i : x_i \in X]$ can be equivalently rewritten as an $atLeast\_(m-k)[\neg x_i : x_i \in X]$ and $exactly\_k[x_i : x_i \in X]$ can be rewritten as $atLeast\_k[x_i : x_i \in X]$ and $atLeast\_(m-k)[\neg x_i : x_i \in X]$, we only consider the $atLeast\_k$ constraints. Usually, each cardinality constraint c has a penalty multiplier $\lambda(c)$.

We differentiate hard and soft cardinality constraints. Hard cardinality constraints are the ones that must be satisfied. Soft cardinality constraints are the ones that can be falsified with a penalty to the cost. In our case, the penalty is proportional to the severity of the violation. Specifically, the penalty is the product of the multiplier λ and the degree of violation (which is the minimum number of variables in the clause that need to be flipped in order to meet the cardinality constraint). For example, for the soft constraints $atLeast\_3[x_1, x_2, x_3, x_4]$ with penalty multiplier λ, the assignment (x_1, x_2, x_3, x_4)=(1, 1, 0, 1) would incur no penalty, while assignment (1, 1, 0, 0) and (0, 1, 0, 0) would incur a penalty of 1*λ and 2*λ respectively.

2.2 Encoding Cardinality Constraints into CNF

Different methods of encoding cardinality constraints into CNF have been studied. We choose three most famous encoding methods in recent years: cardinality networks, sequential encoding and pigeon-hole encoding.

Sequential encoding: The main idea behind the sequential encoding is to encode the sum of the considered variables and then forbid certain output values. It needs $O(k * m)$ auxiliary clauses and variables where k is the cardinality of the constraint and m is the number of the variables of the constraint. Generalized arc consistency is maintained by unit propagation. For more details, we refer the reader to [Boudane *et al.*, 2018; Sinz, 2005].

Cardinality networks encoding: Cardinality networks generate helper variables that are used to sort all the considered truth assignments and then insert clauses which forbid certain outputs. Generalized arc consistency is maintained by unit propagation. It needs $O(m * log_2^2(k))$ auxiliary clauses and variables. For more details on the encoding, we refer the reader to [Boudane *et al.*, 2018; Asín *et al.*, 2009].

Pigeon-hole encoding: The semantic of the cardinality constraint can be equivalently expressed as the problem of putting k pigeons into n holes. We use the improved version obtained by breaking the symmetries and this version also maintains generalized arc consistency. It needs $O(k*(m-k))$ auxiliary clauses and variables. More details can be found in [Boudane *et al.*, 2018; Hattad *et al.*, 2017].

These encondings are proposed to encode hard cardinality constraints into SAT. Actually, they can also be applied to encode soft cardinality constraints. In [Demirovic and Musliu, 2014; Demirovic *et al.*, 2019], they showed how to encoding soft constraints into MaxSAT and we take the same methods. The number of auxiliary variables and clauses used for encoding a soft cardinality constraint is almost the same as the ones used for a hard cardinality constraint.

3 Extended Conjunctive Normal Form

In this section, we introduce Extended Conjunctive Normal Form. We first give the definitions and notations below.

Extended Conjunctive Normal Form. ECNF is extended from CNF as follows. (1) In ECNF, each clause is associated with a positive integer $cardinality[c]$ as its cardinality. (2)

In ECNF, all cardinality constraints are expressed in the form of $atLeast\_k$ constraints. (3) A clause is satisfied iff it has at least $cardinality[c]$ true literals. (4) In addition, as with CNF, we differentiate hard and soft clauses. Each soft clauses s is associated with an integer as its penalty multiplier $\lambda[s]$.

SAT of ENCF formulas. Given an ECNF formula, the definition of SAT remains the same, i.e., to decide whether all clauses can be satisfied, and provide an satisfying assignment if the answer yes.

PMS of ENCF formulas. Given a ECNF formula and an assignment α to its variables, let $num\_true\_l[s]$ denote the number of true literals in clause s, the penalty of a falsified soft clause s is defined as

$$penalty[s] = \lambda[s] \cdot (cardinality[s] - num\_true\_l[s]).$$

Note that we adopt penalty of the linear form, which is adopted in most works dealing with cardinality constraints. But the penalty can be defined using other forms of functions, according to the original problems.

The PMS problem of ECNF formulas, is to find an assignment to satisfy all hard clauses, and to minimize the total penalty of falsified clauses. An assignment α is feasible if all hard clauses are satisfied, and the cost of α, denoted by $gcost(\alpha)$, is the total penalty of falsified soft clauses under α.

For convenience, we denote SAT and Partial MaxSAT problems for ECNF formulas as ESAT and EPMS respectively. If all cardinality of clauses in an ECNF formula is 1, it becomes a CNF formula.

Encoding method. We describe our method of encodng cardinality constraints into ECNF as follows. In our method, we rewrite $atMost\_k$ and $exactly\_k$ constraints as $atLeast\_k$ constraints (described in Section 2.1), and thus we only need to consider the encoding of $atLeast\_k$ constraints.

- For a hard cardinality constraint $atLeast\_k[x_i : x_i \in X]$, generate a hard clause $c := \{x_i : x_i \in X\}$ with $cardinalty[c] = k$.
- For a soft cardinality constraint $atLeast\_k[x_i : x_i \in X]$ and the penalty multiplier λ, generate a soft clause $s := \{x_i : x_i \in X\}$ with $cardinalty[s] = k$ and $\lambda[s] = \lambda$.

4 Local Search Algorithm for EPMS

We introduce a simple local search algorithm *LS-ECNF* and then propose a generalized unit propagation based decimation algorithm to generate the initial solution. Note that, *LS-ECNF* can be used to solve both ESAT and EPMS problems.

Clause weighting techniques are very common in local search algorithms for SAT and MaxSAT. We propose a new weighting scheme for EPMS and the weight maintained by clause weighting techniques is denoted by $w(c)$[1].

The clause weighting scheme is named as Weighting-EPMS, and works as follows: For each hard clause c, we associate an integer number as its weight which is initialized to 1; for each soft clause s, we use $\lambda[s]$ as its initial weight. Whenever a "stuck" situation is observed, that is, we cannot decrease the cost by flipping any variable, the clause weights are updated as follows.

[1]This $w(c)$ is introduced by our algorithm, and is not a part of the instance.

- with probability $1 - sp$: for each falsified hard clause c, $w(c) := w(c) + 1$; for each falsified soft clause c, $w(c) := w(c) + 1$ if $w(c) < \zeta$ ($\zeta = 1000$ in our experiment).
- with probability sp (smoothing probability): for each satisfied clause c, $w(c) := w(c) - 1$;

Local search algorithms use scoring functions and heuristics to guide the search. The most common used scoring function is perhaps the score of variables. In local search for MaxSAT, the score of a variable x is the increase of total weight of satisfied clauses caused by flipping x. In EPMS, the penalty differs according to the severity of the violation, so the traditional scoring functions in local search for MaxSAT are not suitable for EPMS.

Definition 1. ***(Penalty weight of falsified clauses)*** *For each clause c in EPMS (either hard or soft clause), if $num\_true\_l[c] < cardinality[c]$, it incurs a penalty of $w(c) \cdot (cardinality[c] - num\_true\_l[c])$.*

Definition 2. ***(Score in* LS-ECNF*)*** *In* LS-ECNF*, the score of a variable x, denoted by $score(x)$, is the decrease of total penalty weight of falsified clauses caused by flipping x.*

Here is an example. Consider a hard clause $c = \{x_1, x_2, x_3, x_4, x_5\}$ with $cardinality[c] = 3$, and suppose its current weight is 2, given the assignment $(x_1, x_2, x_3, x_4, x_5) = (1, 0, 0, 0, 0)$, $score(x_2) = 2$ and $score(x_1) = -2$ according to our scoring function in *LS-ECNF*. For another assignment $(1, 0, 1, 1, 0)$, $score(x_2)$ is 0 as the number of true literals $num\_true\_l[c] = cardinality[c]$.

4.1 The *LS-ECNF* Algorithm

Our local search algorithm for EPMS (Partial MaxSAT under ECNF), denoted by *LS-ECNF*, is presented below. In the beginning, *LS-ECNF* generates a complete assignment α, and the best found solution α^* is initialized as $\emptyset$.

Algorithm 1: LS-ECNF

Input: EPMS instance F, *cutoff*
Output: A feasible assignment α of F and its cost, or "no feasible assignment found"

```
1  begin
2      α := an initial complete assignment; α* := ∅;
3      while elapsed time < cutoff do
4          if ∄ falsified hard clauses & gcost(α) < gcost*
             then α* := α; gcost* := gcost(α) ;
5          if D := {x|score(x) > 0} ≠ ∅ then
6              v := a variable in D with the highest score;
7          else
8              update weights of clauses by Weighting-EPMS;
9              if ∃ falsified hard clauses then
10                 c := a random falsified hard clause;
11             else c := a random falsified soft clause;
12             v :=the variable with highest score in c;
13         α := α with v flipped;
14     if α* is feasible then return (gcost*, α*);
15     else return "no feasible assignment found";
```

After that, a loop (lines 3-13) is executed to iteratively modify α until a given time limit is reached. During the search, whenever a better feasible solution is found, the best feasible solution α^*, and $gcost^*$, are updated accordingly (line 4). At each iteration, if the set D of the variables with score bigger than zero is not empty, *LS-ECNF* picks the variable with the greatest score in D breaking ties by time stamp.

If there is no such variable, which means the search got stuck, *LS-ECNF* updates clause weights according to the Weighting-EPMS scheme, and picks a variable from a falsified clause. Specifically, it chooses a clause randomly from falsified hard clauses if any, and from falsified soft clauses otherwise. Then, *LS-ECNF* picks the variable with the greatest score in the clause and flips it.

Finally, when the loop terminates upon reaching the time limit, *LS-ECNF* returns $gcost^*$ and α^* if α^* is a feasible solution. Otherwise, it returns "no feasible assignment found".

4.2 UP-based Algorithm for EPMS

Unit Propagation (UP) has been used in SAT and MaxSAT solvers. Actually, UP can also be applied to EPMS with some modifications. Following the line of UP-based decimation [Cai *et al.*, 2017], we integrate UP to design a decimation algorithm for EPMS, and use it to produce initial assignments for *LS-ECNF*.

Definition 3. ***Generalized Unit Clause*** *For a hard (resp. soft) clause c, if its cardinality is bigger or equal than the number of literals in c, then c is a generalized hard (resp. soft) unit clause.*

Generalized Unit Propagation: For a given ECNF formula, we collect all generalized unit clauses in it, and all the literals in these generalized unit clauses are set to true. Then the formula is simplified accordingly.

The GUP-based decimation algorithm is outlined in Algorithm 2 , and is described below. The algorithm works iteratively by assigning variables one by one (line 2-9) until no unassigned variables are left. In each step, it is divided into three situations. (1) Firstly, if there exist a generalized hard unit clause c, the algorithm picks a hard unit clause randomly and performs a generalized unit propagation step using c. (2) Secondly, if there is no generalized hard unit clause but soft ones, the algorithm picks a soft unit clause randomly and works in the same manner as in the first situation. (3) Lastly, if there is no generalized unit clause, then the algorithm picks an unassigned variable x, assigns it randomly, and simplifies the formula accordingly.

Algorithm 2: GUP-Decimation

```
   Input: EPMS instance F
   Output: An assignment of variables in F
1  begin
2      while ∃ unassigned variables do
3          if ∃ generalized hard unit clauses then
4              pick a generalized hard unit clause randomly
                 and perform generalized unit propagation ;
5          else if ∃ generalized soft unit clauses then
6              pick a generalized soft unit clause randomly
                 and perform generalized unit propagation ;
7          else
8              x := pick an unassigned variable randomly;
9              assign x with a random value v, simplify F
                 accordingly;
10     return the resulting assignment to variables of F;
```

5 Experimental Evaluation

We model Nurse Rostering and Tomography Problem into ECNF formulas. We also encode these two problems into CNF formula with three famous encoding types (sequential encoding, cardinality networks encoding and pigeon-hole encoding), Pseudo-Boolean (PB) and Integer Liner Programming (ILP) for comparison. Note that, we encode the original instances as PB or ILP instances directly which is fair for comparison. These two problems are described as follows.

Nurse-Rostering Problem (NRP): These are well-known problem instances from the second international nurse rostering competition (INRC2) [Ceschia *et al.*, 2015]. We chose to focus on this specific problem formulation as it provides a number of instances that include challenging and realistic scheduling problems, while still being intuitive and straightforward to use. The overall goal is to find an optimal roster for a number of nurses, shift types, where every nurse may either work a single shift or have a day off on each day of a given scheduling period. Of course, that this arrangement should satisfy a set of constraints including both hard and soft ones. Finally, the objective function of a candidate solution is defined as the sum of penalty of all violated soft constraints. In this work, the penalty multiplier of each soft constraint is set to 1. The size of NRP instances varies according to the number of nurses and the length of scheduling period. In our experiments, the number of nurses ranges from 30 to 120 and the length of scheduling period ranges from 12 to 24 weeks.

Discrete Tomography Problems: This set of benchmarks we have used is the one introduced in [Bailleux and Boufkhad, 2003]. The idea is to first generate an $N * N$ grid in which some cells are filled and some others are not. The problem consists in finding out which are the filled cells using only the information of how many filled cells there are in each row, column and diagonal. For that purpose, variables x_{ij} are used to indicate whether cell (i, j) is filled. And cardinality constraints impose how many filled cells there are in each row, column and diagonal. We generate 170 benchmarks (10 instances for each size $N = 10, 20...90, 100, 150, ..., 400$). Note that, the discrete tomography problem only contains hard cardinality constraints, and thus is a SAT problem.

We compare our algorithm against five state of the art algorithms including SAT, MaxSAT, Pseudo-Boolean and ILP solvers.

- SATLike: [Lei and Cai, 2018] is the best local search solver for WPMS and won two unweighted categories of incomplete track in MSE (MaxSAT Evaluation) 2018.
- Loandra: [Berg *et al.*, 2019] won two unweighted categories, and was ranked 2nd in two weighted categories of incomplete track in MSE 2019.

Instance size	*LS-ECNF*	*Loandra*			*Open-WBO*			
		car	seq	pig	car	seq	pig	pb
	cmin	cmin	cmin	cmin	cmin	cmin	cmin	cmin
30*12	**103**	360	191	188	804	432	460	2099
30*16	**131**	N/A	806	1253	922	799	1356	15798
30*20	**182**	990	659	379	1393	913	1039	21649
30*24	**195**	2143	918	775	1592	1188	1377	24455
40*12	**67**	753	421	293	988	631	656	19871
40*16	**100**	1185	702	382	1505	844	1045	25936
40*20	**126**	2223	1084	1535	2073	1211	1483	32684
40*24	**148**	4059	1279	1581	2482	1654	2007	39089
50*12	**64**	2213	824	555	1804	1002	926	25039
50*16	**144**	3424	1236	1137	2515	1437	1816	33239
50*20	**122**	N/A	1426	2390	4266	1832	3706	41403
50*24	**225**	N/A	2098	3016	N/A	2182	4231	50032
60*12	**185**	2740	1052	793	2173	1323	1143	32153
60*16	**86**	5202	1599	2588	5948	1684	2506	39480
60*20	**326**	N/A	2137	2999	N/A	2286	3996	52754
60*24	**131**	N/A	2740	N/A	N/A	3086	5770	60130
80*12	**188**	4472	1330	2206	2829	1692	2006	34481
80*16	**222**	5003	2097	2318	3750	2113	3674	45573
80*20	**324**	N/A	2779	3895	N/A	2729	4924	57075
80*24	**319**	N/A	N/A	N/A	N/A	3116	6373	68923
100*12	**82**	N/A	1623	2428	N/A	1521	1932	46241
100*16	**75**	N/A	N/A	4243	N/A	2354	4384	61607
100*20	**140**	N/A	N/A	N/A	N/A	N/A	5734	76860
100*24	**112**	N/A	N/A	N/A	N/A	N/A	7197	92128
120*12	**89**	N/A	N/A	4659	N/A	N/A	5416	58755
120*16	**131**	N/A	N/A	N/A	N/A	N/A	N/A	79248
120*20	**159**	N/A	N/A	N/A	N/A	N/A	N/A	N/A
120*24	**203**	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 1: Experiment results of comparing *LS-ECNF* with the state of the art solvers on nurse-rostering problem. Time limit is 300s

LS-ECNF	*Loandra*			*Open-WBO*				*Gurobi*	
	car	seq	pig	car	seq	pig	pb	comp	heur
cmin	cmin	cmin	cmin	cmin	cmin	cmin	cmin	cmin	cmin
100	144	121	161	487	376	421	899	**80**	**80**
125	174	149	160	600	460	552	1419	**115**	**115**
168	245	230	238	869	631	753	2624	**136**	**136**
189	274	308	260	1165	886	908	3219	**174**	**174**
58	136	100	123	679	446	504	1993	**49**	73
92	208	179	187	968	751	785	3229	**76**	83
121	293	346	285	1599	1096	752	6730	N/A	N/A
142	343	385	359	1858	1497	941	39089	**123**	N/A
59	136	149	120	1123	695	813	3720	**52**	N/A
140	433	296	230	1794	1043	1374	6271	N/A	N/A
114	1021	632	335	2603	1832	1582	41403	N/A	N/A
210	2154	1463	429	3025	2182	2259	50032	N/A	N/A
172	419	240	271	1959	1145	1024	6293	**164**	N/A
82	490	285	182	2573	1431	2103	39480	N/A	N/A
313	3506	1068	617	3581	2286	1923	52754	**280**	N/A
123	3796	1894	871	3581	2699	3263	60130	N/A	N/A
183	396	275	286	2425	1692	1463	8490	**172**	175
214	898	1884	1385	3756	2113	2198	45573	**198**	211
308	2296	1680	697	4657	2729	3263	57075	N/A	N/A
319	4662	2238	1588	4603	3116	3327	68923	N/A	N/A
75	1137	486	126	3410	1521	1668	46241	**72**	75
72	2913	1476	633	4781	2354	3065	61607	N/A	N/A
130	7949	2662	N/A	N/A	3833	5734	76860	N/A	N/A
105	9682	3444	5283	N/A	4151	4083	92128	N/A	N/A
83	4567	1753	638	5869	2788	3054	58755	N/A	N/A
127	10200	2973	2101	14643	3663	4633	79248	N/A	N/A
154	16325	N/A	5630	17283	5118	9696	98526	N/A	N/A
185	N/A	N/A	7803	20410	5960	11504	117653	N/A	N/A

Table 2: Experimental results of comparing *LS-ECNF* with the state of the art solvers on nurse-rostering problem. Time limit is 3600s

- TT-Open-WBO-inc: [Nadel, 2019] won two weighted categories of incomplete track in MSE 2019. We compare with the best version *Open-WBO-Inc-BMO*, and we denote it as Open-WBO for convenience. In addition, Open-WBO can also be used to solve Pseudo-Boolean problems. We mark this version as pb.
- CADICAL: [SAT Race 2019] ranked first in the SAT (satisfiable instances) track and second in the SAT+UNSAT track (of the SAT Race 2019).
- Gurobi: [Gurobi Optimization, 2019] is one of the most powerful mathematical optimization solvers. We use both the complete and heuristic versions which are denoted as $comp$ and $heur$ respectively.

LS-ECNF is implemented in C++ and compiled by g++ with -O3 option. Our experiments were conducted on a server using Intel Xeon E7-8850 v2 @2.30GHz, 2048GB RAM, running Ubuntu 16.04.5 Linux operation system. We adopt various time limits, including 300 seconds and 3600 seconds on nurse rostering problem, and the time limit is 3600 seconds on discrete tomography problems. For nurse rostering problem, in each run, the solver prints successively the best solution it has found so far. For each instance, $cmin$ is the cost of the best feasible solution found. Discrete tomography problems are decision problems, so we show the $time$ of finding the feasible solution of each instance. For each SAT and MaxSAT solver, we use three cardinality encodings marked as car, seq, pig respectively. In bold we present the best results for each instance.

5.1 Experiment Results on NRP

The experiment results on nurse rostering benchmarks are presented in Tables 1 and 2. When the time limit is 300 seconds, *Gurobi* (both complete and heuristic versions) fail to find feasible solutions on all these instances, so we do not report their results. *LS-ECNF* is the best solver, finding better solutions than other solvers, and usually much faster. When the time limit is 3600 seconds (Table 2), MaxSAT and Pseudo-Boolean algorithms still perform much worse than *LS-ECNF*. In the meantime, *Gurobi* can find a good feasible solution for the small instances. However, as the scale of the problem increases, the probability to find a feasible solution drops significantly. When the number of nurses increases to 100, *Gurobi* can hardly find any feasible solution.

Note that, *CaDiCaL* is a SAT solver, so we do not report its results on this optimization problem. Also, *SATLike* fails to find feasible solutions on almost all instances, so its results are not reported, either.

5.2 Experiment Results on DTP

Experiment results on discrete tomography problem are presented in Table 3. The MaxSAT version of Open-WBO is

Instance	*LS-ECNF*	*SATLike*			*Loandra*			*Open-WBO*	*CADICAL*			*Gurobi*	
		car	seq	pig	car	seq	pig	pb	car	seq	pig	comp	heur
N*N	time	time	time	time	time	time	time	time	time	time	time	time	time
10*10	<**0.01**	0.05	0.05	0.06	0.07	0.05	0.05	<0.01	<0.01	<0.01	<0.01	0.03	0.03
20*20	<**0.01**	21.48	N/A	N/A	3.99	0.64	0.64	8.08	1.05	0.6	1.92	0.16	0.26
30*30	<**0.01**	66.81	N/A	N/A	17.21	4	N/A	328.30	74.91	19.38	4.84	0.45	1.29
40*40	<**0.01**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	76.13	N/A	1.96	2.42
50*50	**0.01**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	11.81	7.56
60*60	**0.01**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.87	3.40
70*70	**0.03**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2.01	3.21
80*80	**0.03**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.63	4.74
90*90	**0.05**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.57	6.10
100*100	**0.06**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.64	7.73
150*150	**0.32**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	27.90	33.62
200*200	**3.46**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	71.43	114.06
250*250	**5.60**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	158.90	238.78
300*300	**17.63**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	319.20	332.98
350*350	**64.62**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	537.99	866.60
400*400	**114.43**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1339.50	1159.98

Table 3: Experimental Results on Discrete Tomography Problems.

much poorer than the PB version, so we only report the results of PB version. The results are impressive. All MaxSAT, SAT and PB solvers can only solve small instances with size $N \leq 40$, while *LS-ECNF* solves all these instances very quickly. *LS-ECNF* finds solutions in 1 second when the size $N \leq 150$. Though *Gurobi* (both complete and heuristic versions) can also solve all these instances, the running time of *Gurobi* is more than 10 times that of *LS-ECNF*.

In addition, we also compared our algorithm with local search SAT solver *CCAnr* [Cai *et al.*, 2015], complete SAT solver *MapleLCMDistChronoBT-DL* [SAT Race 2019] and PB solver *RoundingSAT* [Elffers and Nordström, 2018] on discrete tomography problems. All these solvers fail to solve instances with size $N > 40$, so we did not report their results.

5.3 Experiment Analysis

In our experiments, GUP-based algorithm can improve the performance over *LS-ECNF* in more than 70% instances on NRP. And for DTP, using GUP-based algorithm can save more than 20% running time. Generally, techniques from SAT and MaxSAT can be beneficial for ESAT and EPMS solvers. Indeed, we can develop a DPLL framework for ECNF problems. During the propagation, a variable x is pick to be assigned. For a clause c s.t. $l \in c$ where l is x or its negation, if l is true, than $num\_true\_l[c]$ is increased by 1, otherwise, l is removed from c. If $|c| = cardinality[c]$, then c is a generalized unit clause and all the rest literals in c should be set to true. If $|c| < cardinality[c]$, which means there is a conflict, then backtracking is executed.

5.4 Relationship Between ECNF with CNF and PB

ECNF extends CNF by introducing the concept of $cardinality$. One can encode cardinality constraints into ECNF without any auxiliary variables and clauses. In Table 4, we present the average number of variables and clauses of each encoding. We can see that, encoding cardinality constraints into ECNF is more compact and efficiency. Because that ECNF and CNF share almost the same data structure, techniques used for problems of CNF (e.g. SAT) can be used for problems of ECNF (e.g. ESAT). ECNF can be considered as a special case of Pseudo-Boolean Optimisation (PBO). In PBO, each literal of each constraint is associated with an integer number as its weight (e.g. $2x_1 + 3x_2 - 4x_3 > 2$). If all the weight of literals is 1, it becomes an ECNF formula.

Benchmark	*ECNF*	*CNF*		
		car	seq	pig
variables				
NRP	$1.2 * 10^5$	$6.7 * 10^6$	$5.3 * 10^6$	$1.9 * 10^6$
DTP	$1.1 * 10^4$	$3.9 * 10^6$	$5.6 * 10^6$	$2.8 * 10^6$
clauses				
NRP	$2.8 * 10^5$	$1.0 * 10^7$	$2.1 * 10^7$	$4.0 * 10^6$
DTP	$8.9 * 10^2$	$5.9 * 10^6$	$2.4 * 10^7$	$5.7 * 10^6$

Table 4: Averaged number of variables and clauses of each encoding

6 Conclusion

We proposed an extension of CNF formulas, which can easily deal with cardinality constraints without any auxiliary variable and clause. We also designed an efficient local search algorithm and a generalized UP-based algorithm for ECNF. Experiments results showed that our solver is much better than all state of the art SAT, MaxSAT, PB and ILP solvers for nurse rostering problems and discrete tomography problems. The strong results show that we opened a promising direction to solve cardinality constraints. We would like to encode other combinatorial problems with cardinality constraints into ECNF and design a more efficient solver.

Acknowledgements

This work is supported by Beijing Academy of Artificial Intelligence (BAAI) and Youth Innovation Promotion Association, Chinese Academy of Sciences [No. 2017150].

References

[Asín *et al.*, 2009] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks and their applications. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 167–180, 2009.

[Bailleux and Boufkhad, 2003] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, pages 108–122, 2003.

[Berg *et al.*, 2019] Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete maxsat. In Louis-Martin Rousseau and Kostas Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings*, volume 11494 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2019.

[Boudane *et al.*, 2018] Abdelhamid Boudane, Saïd Jabbour, Badran Raddaoui, and Lakhdar Sais. Efficient sat-based encodings of conditional cardinality constraints. In *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, pages 181–195, 2018.

[Cai *et al.*, 2015] Shaowei Cai, Chuan Luo, and Kaile Su. Ccanr: A configuration checking based local search solver for non-random satisfiability. In Marijn Heule and Sean A. Weaver, editors, *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, volume 9340 of *Lecture Notes in Computer Science*, pages 1–8. Springer, 2015.

[Cai *et al.*, 2016] Shaowei Cai, Chuan Luo, Jinkun Lin, and Kaile Su. New local search methods for partial maxsat. *Artif. Intell.*, 240:1–18, 2016.

[Cai *et al.*, 2017] Shaowei Cai, Chuan Luo, and Haochen Zhang. From decimation to local search and back: A new approach to maxsat. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 571–577, 2017.

[Ceschia *et al.*, 2015] Sara Ceschia, Nguyen Dang Thi Thanh, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. Second international nurse rostering competition (INRC-II) - problem description and rules -. *CoRR*, abs/1501.04177, 2015.

[Demirovic and Musliu, 2014] Emir Demirovic and Nysret Musliu. Modeling high school timetabling as partialweighted maxsat. In *LaSh 2014: The 4th Workshop on Logic and Search (a SAT/ICLP workshop at FLoC 2014), July 18, Vienna, Austria*, 2014.

[Demirovic and Stuckey, 2019] Emir Demirovic and Peter J. Stuckey. Techniques inspired by local search for incomplete maxsat and the linear algorithm: Varying resolution and solution-guided search. In Thomas Schiex and Simon de Givry, editors, *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*, volume 11802 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2019.

[Demirovic *et al.*, 2019] Emir Demirovic, Nysret Musliu, and Felix Winter. Modeling and solving staff scheduling with partial weighted maxsat. *Annals OR*, 275(1):79–99, 2019.

[Elffers and Nordström, 2018] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-boolean solving. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1291–1299. ijcai.org, 2018.

[Guerreiro *et al.*, 2019] Andreia P. Guerreiro, Miguel Terra-Neves, Inês Lynce, José Rui Figueira, and Vasco M. Manquinho. Constraint-based techniques in stochastic local search maxsat solving. In Thomas Schiex and Simon de Givry, editors, *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*, volume 11802 of *Lecture Notes in Computer Science*, pages 232–250. Springer, 2019.

[Gurobi Optimization, 2019] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019.

[Hattad *et al.*, 2017] Soukaina Hattad, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Enhancing pigeon-hole based encoding of boolean cardinality constraints. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017*, pages 299–307, 2017.

[Joshi *et al.*, 2019] Saurabh Joshi, Prateek Kumar, Sukrut Rao, and Ruben Martins. Open-wbo-inc: Approximation strategies for incomplete weighted maxsat. *J. Satisf. Boolean Model. Comput.*, 11(1):73–97, 2019.

[Karpinski and Piotrów, 2019] Michal Karpinski and Marek Piotrów. Encoding cardinality constraints using multiway merge selection networks. *Constraints*, 24(3-4):234–251, 2019.

[Lei and Cai, 2018] Zhendong Lei and Shaowei Cai. Solving (weighted) partial maxsat by dynamic local search for SAT. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1346–1352, 2018.

[Luo *et al.*, 2017] Chuan Luo, Shaowei Cai, Kaile Su, and Wenxuan Huang. CCEHC: an efficient local search algorithm for weighted partial maximum satisfiability. *Artif. Intell.*, 243:26–44, 2017.

[Martins *et al.*, 2014] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-wbo: A modular maxsat solver,. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 438–445, 2014.

[Nadel, 2019] Alexander Nadel. Anytime weighted maxsat with improved polarity selection and bit-vector optimization. In Clark W. Barrett and Jin Yang, editors, *2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019*, pages 193–202. IEEE, 2019.

[Narodytska and Bacchus, 2014] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided maxsat resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2717–2723, 2014.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 827–831, 2005.