A Backbone-Based Co-evolutionary Heuristic for Partial MAX-SAT

Mohamed El Bachir Menaï[1] and Mohamed Batouche[2]

[1] Laboratoire d'Intelligence Artificielle, Université de Paris8,
2 rue de la liberté, 93526 Saint-Denis, France
menai@ai.univ-paris8.fr
[2] Laboratoire LIRE, Département d'Informatique,
Université Mentouri, 25000 Constantine, Algérie
batouche@wissal.dz

Abstract. The concept of backbone variables in the satisfiability problem has been recently introduced as a problem structure property and shown to influence its complexity. This suggests that the performance of stochastic local search algorithms for satisfiability problems can be improved by using backbone information. The Partial MAX-SAT Problem (PMSAT) is a variant of MAX-SAT which consists of two CNF formulas defined over the same variable set. Its solution must satisfy all clauses of the first formula and as many clauses in the second formula as possible. This study is concerned with the PMSAT solution in setting a co-evolutionary stochastic local search algorithm guided by an estimated backbone variables of the problem. The effectiveness of our algorithm is examined by computational experiments. Reported results for a number of PMSAT instances suggest that this approach can outperform state-of-the-art PMSAT techniques.

1 Introduction

Many problems in artificial intelligence (AI) and operations research (OR) are optimization problems, where the objective is to find a best assignment to a set of variables such that a set of constraints are satisfied. Real world problems found in application areas including scheduling [4] and pattern recognition [12] contain hard and soft constraints. Hard constraints must be satisfied by any solution, while soft constraints specify a function to be optimized. Various approaches have been proposed to represent over-constrained problems. Freuder and Wallace [12] presented the concept of partial constraint satisfaction, where the objective is to maximize the total number of satisfied constraints. Borning *et al.* [7] introduced the notion of constraint hierarchies, where the distinction between hard and soft constraints is extended to a multiple level constraint hierarchy.

Boolean satisfiability (SAT) is among the most interesting AI formalisms for reasoning, planning and learning [23]. The SAT problem asks to decide whether a given propositional formula, in conjunctive normal form (CNF), has a model. The maximum satisfiability (MAX-SAT) problem is the optimization version of SAT which consists to find an assignment maximizing the number of satisfied clauses. The weighted MAX-SAT is a more general case, where each clause is associated with a positive weight. The goal is to minimize the sum of weights of violated clauses. Problems involving hard and soft constraints can be naturally encoded as weighted MAX-SAT. Each hard constraint can be represented by a weighted cost which exceeds the sum of the weighted cost of all soft constraints. However, a solution for a MAX-SAT instance may violate some clauses whose satisfiability is a necessary condition for the feasibility of the real solution. For example, a MAX-SAT solution for a university time tabling may contain collisions of different courses in the same room at the same time, the same lecturer can be scheduled in different rooms at the same time, and so on. Cha *et al.* [8] introduced the Partial MAX-SAT (PMSAT) to formulate independently hard and soft constraints. Hard constraints are called *mandatory clauses*; their satisfiability is required for any PMSAT solution. Other related problems to PMSAT are the DISTANCE-SAT defined by Bailleux and Marquis [2], and the sub-SAT introduced by Xu *et al.* [24]. The DISTANCE-SAT problem asks to check if there is a model of a CNF formula, that conflicts with an expected configuration on at most a given number of variables. The sub-SAT is a formulation for relaxed Boolean satisfiability, that allows violation of a given number of clauses in a CNF formula.

The current research on algorithms used to solve PMSAT is limited. Cha *et al.* [8] used a weighting-type stochastic local search to solve PMSAT by repeating each mandatory clause n times. In this way, the search always prefers a solution which satisfies all mandatory clauses, regardless of the level of remaining clause violations. However, this can lead to an important increasing of the total number of clauses when their number is initially large. They applied various strategies to escape from local minima such as LWM, RESTART and RESET [8]. LWM strategy consists to add weights to all unsatisfied clauses, and to continue the search when a local minimum is reached. RESTART strategy allows the algorithm to restart from a random initial assignment, while RESET consists to reset the weights given by the algorithm and to continue the search from the current assignment. In the reported experimental study [8], RESET outperforms LWM and RESTART on random instances. In [14], a new approach for solving PMSAT is described. It is based mainly on recycling a model of the mandatory clauses to satisfy as many clauses in the second formula as possible. The reported results show the overall superiority of this method in comparison to a weighting-type local search algorithm. A problem of practical significance in the design of SAT and MAX-SAT solvers, is how to identify and exploit the problem structure properties to improve their performance. Some interesting properties which influence the hardness of a SAT have been identified such as the *easy-hard-easy* phase transition [9, 15], and the *backbone variables* [16], a set of literals which are true in every model. The backbone of a MAX-SAT instance is the set of assignments of values to variables which are the same in every possible optimal solution [20].

The aim of this paper is to integrate a backbone guide moves to a co-evolutionary stochastic local search algorithm for solving the PMSAT problem.

In a first phase, both formulas of a PMSAT instance are solved as a single MAX-SAT instance using a backbone guided co-evolutionary search. In a second phase, the best assignment found is recycled to satisfy all mandatory clauses using the estimated backbone. The effectiveness of this algorithm is demonstrated empirically on some PMSAT instances derived from standard SAT instances. In the reminder of this paper, we explain in more details the proposed method for PMSAT and report on results of computational tests in which our algorithm is compared to related approaches. In the next section, we describe a co-evolutionary method used for MAX-SAT (Bose-Einstein Extremal Optimization). In section 3, we present a brief review of backbone variables and related notions. In section 4, we formalize a new method for PMSAT. In section 5, we report on experimental results. We finally conclude and plan for future work in section 6.

2 Bose-Einstein Extremal Optimization Method for MAX-SAT

Bose-Einstein Extremal Optimization (BE-EO) [13] is an approximative algorithm for solving the MAX-SAT problem. It is based on an adaptation of Extremal Optimization (EO) [5] heuristic to MAX-SAT. The search space is explored according to EO, while starting solutions are sampled using the Bose-Einstein probability distribution.

Extremal Optimization method is introduced by Boettcher and Percus [5] for solving hard optimization problems such as the Graph Partitioning. It was motivated by the Bak-Sneppen [3] model of biological evolution which describes the co-evolutionary process of species. In this model, optimal adaptation emerges naturally from the dynamics of species by elimination of badly adapted ones. Species are sites of a lattice and each one has an associated fitness value ranging from 0 to 1. A fitness represents a time scale at which the species will mutate to a different species or become extinct. A selection process against the worst adapted species is applied. At each update, the smallest fitness value is replaced by a new random one which impacts the fitness values of its neighbors. After a certain number of steps, a state of optimal adaptation (*Self-Organized Criticality*) is reached in which all species are intimately connected. When the system is driven back to a SOC state, any perturbation of this equilibrium involves large fluctuations in the configuration of fitness values (*critical avalanches*). The duration t of these avalanches follows a power-law distribution $P(t) \propto t^{-\tau}$ (τ close to 1). Extremal Optimization method is a conversion of the extremal dynamics of the Bak-Sneppen model into an approximative algorithm for optimization problems. The search process is characterized by hill-climbing large fluctuations (i.e. avalanches in the Bak-Sneppen model) allowing search diversification. It evolves to a SOC state where sub-optimal solution can be found (almost all species have optimal fitnesses).

The Bose-Einstein distribution is a quantum distribution function. It describes the probability distribution of an amount of energy between identical but indistinguishable particles with integer spin, called *bosons* (e.g. photons).

Szedmak [21] proved that this distribution function can improve the performance of stochastic local search algorithms for satisfiability problems. He demonstrated that the mean Hamming distance between a sample of initial solutions and the optimal solution is reduced when initial solutions are generated using the Bose-Einstein distribution rather than the uniform one.

Given a MAX-SAT instance of n Boolean variables $x_1, \ldots, x_n$, and m weighted clauses $(c_i, w_i)_{i=1,m}$. Each clause c_i is a disjunction of literals (a variable x_i or its negation $\neg x_i$), and $w_i \in \mathbb{N}$ is its weight. A MAX-SAT instance is a conjunction of clauses (CNF formula). The fitness λ_i of a variable x_i is defined as the negation of the fraction of the sum of weights of unsatisfied clauses in which x_i appears, by the total weights of clauses connected to this variable :

$$\lambda_i = \frac{-\sum_{j=1}^{m} w_j | x_i \in c_j, \Im(c_j) = 0}{\sum_{k=1}^{m} w_k | x_i \in c_k} \tag{1}$$

$\Im(c_j) = 0$ means that the clause c_j is unsatisfied. The cost contribution of a variable x_i is defined by $-\lambda_i$. The best solution S found to a MAX-SAT instance is associated to the minimum of the cost function $C(S) = -\sum_{i=1}^{n} \lambda_i$.

The algorithm BE-EO for MAX-SAT is outlined as follows [13].

Algorithm BE-EO/MAX-SAT

```
1. Randomly generate a solution $S$ according to the Bose-Einstein distribution.
   Set $S_{max} \leftarrow S$.
2. If $S$ satisfies all the clauses of the MAX-SAT instance, return ($S$ : model).
3. Evaluate $\lambda_i$ for each variable $x_i$.
4. Rank $x_i, (i = 1, n)$ from the worst to the best according to $\lambda_i$. Select a rank
   $j$ such that $P(j) \propto j^{-\tau}$.
5. Flip the truth value of $x_j$ in $S$.
6. If $C(S) < C(S_{max})$ then set $S_{max} \leftarrow S$.
7. If the number of steps does not exceed the given bound, return to step 2.
8. If the number of generated Bose-Einstein initial solutions does not exceed
   the given sample size, then randomly generate a solution $S$ according to the
   Bose-Einstein distribution. Return to step 2.
9. Return ($S_{max}$).
```

Good performance is reported for BE-EO/MAX-SAT on some specific classes of weighted and unweighed MAX-SAT instances [13] outperforming WalkSAT [18] and a tabu search method.

3 Backbone Variables

The *backbone* of a problem instance is a set of variables having fixed values in all optimal solutions. These variables are critically constrained as the elimination of any one of them will exclude any optimal solution. Related notions to backbone in satisfiability are *backdoors* [23] and *spine* [6]. A backdoor is a variable subset such that if some particular truth values are assigned to these variables, the simplified instance is satisfiable and can be solved in polynomial time. Williams *et al.* [23] demonstrated that a concrete computational advantage can be obtained by exploiting backdoors. The spine of a set of clauses is a set of literals which are false in all models of a subset of satisfiable clauses [20].

Several researches dealing with competitive SAT and MAX-SAT solvers have made use of backbone variables. Monasson *et al.* [16] investigated the backbones of 3-SAT and $(2+p)$-SAT, and conjectured that a backbone is an order parameter for the decision problems. Other studies [17, 19, 1] have demonstrated that the size of the backbone is correlated with the hardness of SAT problems. Slaney and Walsh [20] have studied backbones in optimization and approximation problems including graph coloring, traveling salesperson problem, number partitioning and blocks word planning. They showed that backbones are often an important indicator of hardness in optimization and approximation. Subsequently, heuristic methods which identify backbone variables, may reduce problem difficulty and improve performance. Dubois and Dequen [11] proposed a systematic search method which incorporates estimated backbone variables. Telelis and Stamatopoulos [22] designed a method for generating initial assignments to an iterated algorithm by sampling heuristically the backbone variables, and reported good results on some random MAX-SAT instances. Climer and Zhang [10] developed a technique for identifying backbones and *fat variables* (variables which are absent from every optimal solution). They exploited it for discovering backbone and fat arcs for instances of the asymmetric traveling salesperson problem (ATSP) and achieved performance improvements. Zhang *et al.* [25] improved the performance of the well known WalkSAT procedure [18] on some instances of SAT and MAX-SAT from SATLIB [27] using structure information of reached local minima.

4 Backbone-Based Co-evolutionary Heuristic for PMSAT

Given two CNF formulas f_A and f_B over a set of variables $X = \{x_1, \ldots, x_n\}$. The PMSAT problem $P = f_A \wedge f_B$ asks to satisfy all the clauses of f_A and as many clauses in f_B as possible. The number of satisfied clauses in f_B determines the quality of a solution to P.

We propose a two-phase algorithm for solving P. In a first phase, P is considered as a MAX-SAT instance and approximated using a variant of the algorithm BE-EO/MAX-SAT. A backbone variables sampling is integrated to BE-EO/MAX-SAT to guide the search towards potentially good solutions. If the best solution found S_{AB} does not satisfy f_A, then a second phase is performed to recycle S_{AB} to a model of f_A using the backbone information captured in the first phase. The backbone sampling may help to improve the performance of the second phase process, as it encapsulates information about the likelihood of each variable. However, exact backbone cannot be computed unless all optimal solutions are known. Hence, only an estimated pseudo-backbone is performed using information extracted from reached local minima.

The pseudo-backbone sampling used in this work is inspired by the sampling scheme presented in [22]. Let Ω be a set of solutions on X. $S(x_i)$ denotes the truth value of x_i in the solution S. A variable frequency of positive occurrences of x_i in all solutions of Ω, is defined by :

$$p_i = \frac{\sum_{S \in \Omega} S(x_i)}{|\Omega|} \tag{2}$$

assuming that all local minima are of equal quality. Else a weight cost may be assigned to each local minimum. $Q(S)$ denotes the contribution of a solution S, defined as the total number of satisfied clauses in f_A and f_B. A multiplier coefficient, equals to $|f_A|$, is added to $Q(S)$ to underline the priority of satisfying clauses of f_A. Let $\#sat_{f_A}(S)$ and $\#sat_{f_B}(S)$ be the number of satisfied clauses by S in f_A and f_B, respectively. $Q(S)$ is defined by :

$$Q(S) = |f_A| \cdot \#sat_{f_A}(S) + \#sat_{f_B}(S) \tag{3}$$

A more reliable definition of $p_i (i = 1, n)$ is given by :

$$p_i = \frac{\sum_{S \in \Omega} Q(S) \cdot S(x_i)}{\sum_{S \in \Omega} Q(S)} \tag{4}$$

Let X_α denotes the set of variables which appear in the set of clauses α. The main steps of the algorithm, called BBC-PMSAT, are described as follows.

Algorithm BBC-PMSAT

Phase 1 : Solving $P = f_A \wedge f_B$ as a MAX-SAT instance

(a) Run BE-EO/MAX-SAT on P over X. Initialize Ω with reached local minima.

(b) Solve P using a variant of BE-EO/MAX-SAT (initial solutions are generated from Ω using variable frequencies p_i (Eqn. 4)).
At a new local minimum S, if Ω holds a solution S^* such that
$Q(S^*) < Q(S)$ (Eqn. 3), then replace S^* by S in Ω.

(c) Let S_{AB} be the best solution found after a preset number of steps.
If S_{AB} satisfies f_A then return S_{AB} as a solution to P.

Phase 2 : Recycling S_{AB} to satisfy f_A

(a) Let $f_A = f_{A_1} \wedge f_{A_2}$, where f_{A_1} is satisfied by S_{AB} and $X_{A_1} \cap X_{A_2} = \emptyset$ (simplification).

(b) Solve f_{A_2} over X_{A_2} as a SAT instance using a variant of the BE-EO/SAT (Phase 1, step (b)). Partial assignments to the variables of X_{A_2} are generated using variable frequencies p_i.

(c) After a preset number of steps, if a model S_{A_2} is found, then update S_{AB} and return it as a solution to P. Else return that no solution to P can be found.

5 Performance Evaluation

Since no public PMSAT instances are available, we generated them using SAT instances from DIMACS [26] and SATLIB [27] benchmark archives. We considered four sets of random and structured SAT instances of n variables and m clauses:

- uuf125-538* (100 random "phase-transition" hard 3-SAT instances of $n = 125$ and $m = 538$);
- f* (3 large random "phase-transition" hard 3-SAT instances: f600 ($n = 600$, $m = 2550$), f1000 ($n = 1000, m = 4250$), f2000($n = 2000, m = 8500$);
- par8-* (5 instances of SAT-encoded parity learning problem of $n = 350$ and $1149 < m < 1171$);
- flat* (10 instances of SAT-encoded graph coloring problem of $n = 300$ and $m = 1117$).

All SAT instances are chosen satisfiable in order to guarantee the generation of solvable PMSAT instances (f_A must be satisfiable). Random SAT instances are "phase-transition" hard ($\frac{m}{n} \simeq 4.3$ for random 3-SAT instances). Structured instances par8-* are also among the hardest DIMACS SAT ones. Random instances are generally used to control average problem difficulty by varying the ratio $\left(\frac{m}{n}\right)$ of clauses to variables, while structured instances are used to measure the effect of hidden structure on algorithm performance. PMSAT instances are generated using a partition of each SAT instance into two subsets F_A and F_B (representing f_A and f_B formulas, resp.) such that $|F_A| = [\alpha m] + 1$ and $|F_B| = m - |F_A|$, with $0 < \alpha < 1$. The program code is written in C and run on a computer (Pentium IV 2.9 GHz with 1 GBs of RAM) running Linux. BE-EO/MAX-SAT is run setting $\tau = 1.4$. All the results are averaged over 10 runs on each instance with a maximum of 300000 flips allowed per run. The total number of tries for each run of the algorithm BBC-PMSAT is shared between both phases of the algorithm. Let r be the first phase run length ratio of the total run length, $\#sat$ the number of solutions to PMSAT instances over 10 runs (it equals the average number of satisfied f_A instances) and v the relative error of a solution S given by:

$$v(\%) = \left(1 - \frac{\#sat_{f_B}(S)}{|F_B|}\right) \times 100 \tag{5}$$

A key question regarding the algorithm BBC-PMSAT is how to evaluate its performance. The first objective is to determine the effect of the first phase run length ratio of the total run length. The second objective is to determine the impact of the pseudo-backbone variables size on the performance. The third objective is to determine whether or not BBC-PMSAT is competitive with its variant, called C-PMSAT, which does not integrate pseudo-backbone sampling. Additionally, BBC-PMSAT is compared to a weighting-type local search algorithm, called WLS, used by Cha *et al.* [8] with RESET strategy to solve PMSAT

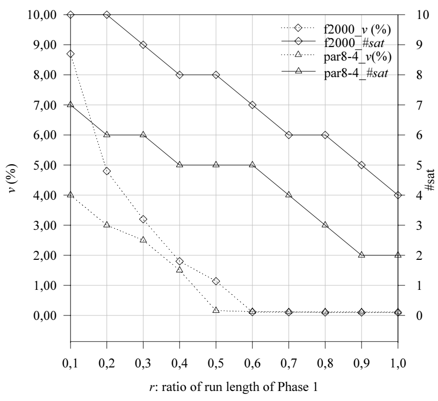

Fig. 1. Average error v (y-axis) and number of solutions $\#sat$ (additional y-axis) over 10 runs for instances f2000 and par8-4 ($\alpha = 0.3$) are plotted against the run length ratio of phase 1

instances. WLS/RESET solves PMSAT as MAX-SAT instance by repeating each clause in F_A, $|F_A|$ times.

Figure 1 presents average $\#sat$ and v over 10 runs obtained by BBC-PMSAT on the instances f2000 and par8-4, varying r from 10% to 100%. We observe clearly that the greater the value of r, the more the number of solutions $\#sat$ and the error v are reduced. An error v less than 1% is achieved after at least 50% of the total runtime length. However, allowing much more time to the first phase of the algorithm, means reducing the amount of time allowed to the second phase. Hence, the number of solutions $\#sat$ to PMSAT may decrease. For all

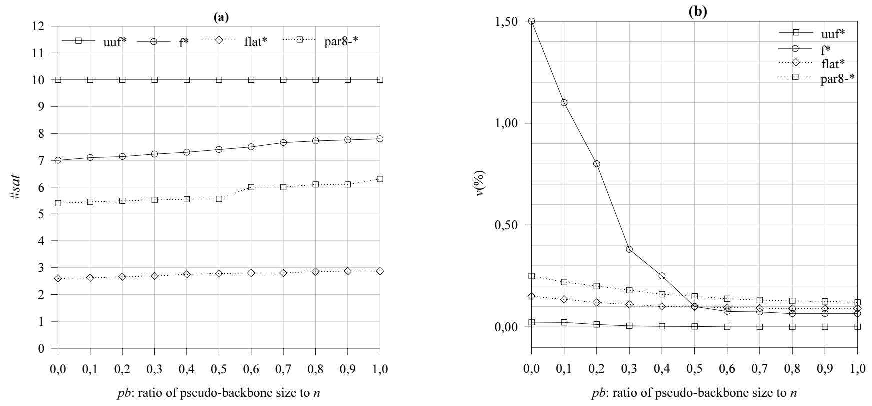

Fig. 2. ($\alpha = 0.3, r = 0.6$) **(a)** Average number of solutions $\#sat$ (y-axis) is plotted against the ratio of the pseudo-backbone size to the number n of variables for each instance class. **(b)** Average error v (y-axis) is plotted against the ratio of the pseudo-backbone sample size to the number n of variables for each instance class.

the instances, the average best performance in terms of average $\#sat$ and v is obtained with r ranging from 50% to 70%.

Figure 2 shows the number of solutions $\#sat$ and the average error v achieved by BBC-PMSAT on all the instances when varying pb, the ratio of backbone size to the number of variables n, from 0 to n ($\alpha = 0.3, r = 0.6$). As illustrated in figure 2.a, the number of solutions $\#sat$ is generally increasing with pb. It is constant in the case of uuf* which may have few backbone variables. For example, setting pb to $0.5n$, the gains achieved by BBC-PMSAT in terms of $\#sat$ on the instances f*, flat* and par8-* are 5.71%, 6.92% and 2.96%, respectively. For $pb = n$, the gains achieved on the same instances are 11.42%, 10.38% and 16.66%, respectively. Figure 2.b shows a fall in the average error v for all the instances: v decreases rapidly for the instances f*, until $pb = 0.5n$; v decreases relatively slowly for the remaining instances. For all the instances, the average best performance, in terms of quality of a solution (v), is obtained when $pb \geq 0.6n$. BBC-PMSAT performs

Table 1. Results of the algorithm BBC-PMSAT ($\alpha = 0.3, r = 0.6, pb = 0.7n$)

Instance	#sat	v(%)	CPU time		Flips	
			Avg	Std	Avg	Std
uuf*	10	0	0.374	0.050	3881.9	556.0
f600	9	0.0113	6.092	1.815	28958.2	4623.1
f1000	7	0.0951	19.812	3.152	79575.1	8768.4
f2000	7	0.1135	44.655	3.547	138120.9	9868.5
flat*	2.8	0.0923	3.238	0.185	26926.8	1601.4
par8-1	7	0.0713	5.117	1.350	34877.1	9215.0
par8-2	7	0.1501	5.723	0.702	39681.0	4886.0
par8-3	6	0.1185	4.931	1.050	31657.5	7325.1
par8-4	5	0.1290	4.581	0.900	31917.0	6513.0
par8-5	5	0.1870	7.960	0.841	61275.4	4935.0
Average	6.58	0.0968	10.248	1.359	47687.0	5829.1

Table 2. Results of the algorithm C-PMSAT ($\alpha = 0.3, r = 0.6, pb = 0.7n$)

Instance	#sat	v(%)	CPU time		Flips	
			Avg	Std	Avg	Std
uuf*	10	0.0231	0.514	0.200	6162.0	971.3
f600	9	0.0130	10.973	2.532	57457.3	7929.7
f1000	6	1.9150	22.650	1.650	115059.2	7021.6
f2000	6	2.5710	54.458	7.940	189310.3	35641.8
flat*	2.6	0.1504	4.154	1.201	32795.1	5831.0
par8-1	6	0.3511	7.516	0.324	60513.3	1472.1
par8-2	6	0.2510	7.380	0.508	58764.0	2410.5
par8-3	6	0.2315	5.763	1.810	44210.4	8355.0
par8-4	4	0.1712	4.716	0.152	36638.1	790.0
par8-5	5	0.2415	10.550	0.380	71652.1	1582.0
Average	6.06	0.5918	12.867	1.669	67256.1	7200.5

Table 3. Results of the algorithm WLS/RESET

Instance	#sat	v(%)	CPU time		Flips	
			Avg	Std	Avg	Std
uuf*	9.8	0.0123	0.845	0.205	8316.4	1485.0
f600	9	0.0130	10.620	2.010	47150.6	18420.0
f1000	6	0.1235	29.271	2.055	136171.0	19040.0
f2000	5	1.5764	60.028	5.601	265124.5	41190.0
flat*	2.1	0.2351	6.068	1.295	43166.3	9147.6
par8-1	4	0.3133	10.408	0.520	79525.1	3290.0
par8-2	4	0.4840	9.120	0.180	70720.0	1123.1
par8-3	3	0.3586	7.151	0.642	53540.2	4058.0
par8-4	1	0.6315	5.154	0.210	39628.0	1620.5
par8-5	4	0.4099	10.250	1.261	73040.2	7850.0
Average	4.79	0.4158	14.891	1.397	81638.2	10722.4

particularly well on the instances par8-* which may have a large backbone size, making all the variables critically constrained.

Computational results performed by BBC-PMSAT, C-PMSAT and WLS/RESET are presented in tables 1, 2 and 3, respectively. The first column lists the benchmarks. Columns 2, 3 show the average number of solutions $\#sat$ and the average error v over 10 runs. Columns 4, 5 show the average CPU time and its standard deviation. Columns 6, 7 show the average number of flips and its standard deviation. BBC-PMSAT is tested using $\alpha = 0.3$, $r = 0.6$ and $pb = 0.7n$. Overall, BBC-PMSAT outperforms C-PMSAT and WLS/RESET on all the instances. The average gains in number of solutions are 8.58% and 37.36% w.r.t. C-PMSAT and WLS/RESET, respectively. In term of runtime cost, the average falls are 20.35% and 31.17% w.r.t. C-PMSAT and WLS/RESET, respectively. In conclusion, our results demonstrate that BBC-PMSAT can find high quality solution and performs faster than C-PMSAT and WLS/RESET.

6 Conclusion and Future Work

In this work, we introduced a backbone-based co-evolutionary algorithm for PMSAT (BBC-PMSAT). This algorithm is based on a co-evolutionary stochastic local search method (BE-EO) which has been used successfully for solving a range of MAX-SAT instances. BBC-PMSAT approximates solutions to PMSAT in two main phases. In a first phase, PMSAT is solved as a MAX-SAT instance incorporating sampled pseudo-backbone variables to guide the search. In a second phase, the previously found solution is recycled to satisfy all the constrained clauses using estimated pseudo-backbone. BBC-PMSAT was compared to its variant without pseudo-backbone sampling (C-PMSAT) and to a weighting-type stochastic local search algorithm with RESET strategy (WLS/RESET) [8] for PMSAT. These algorithms were tested on four classes of PMSAT instances generated from standard SAT instances. The results indicate the effectiveness of using estimated pseudo-backbone variables. Indeed, BBC-PMSAT outperforms

C-PMSAT and WLS/RESET on all the instances in terms of average number of solutions and average runtime cost. The most significant gains are achieved on instances which may have large backbone size. The encouraging results obtained at this early stage, prove the high potential of this method. In future work, we plan to further investigate how the performance of BBC-PMSAT depends on the problem features and to continue computational tests on larger PMSAT instances.

References

[1] Achlioptas, D., Gomes, C., Kautz, H., Selman, B.: Generating satisfiable problem instances. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00), (2000) 256–261

[2] Bailleux, O., Marquis, P.: DISTANCE-SAT: complexity and algorithms. In Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99), (1999) 642–647

[3] Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters, 59, (1993) 381–384

[4] Beck, J.C., Fox, M.S.: A generic framework for constraint-directed search and scheduling. AI Magazine, 19(4), (1998) 101–130

[5] Boettcher, S., Percus, A.G.: Nature's way of optimizing. Artificial Intelligence, 119, (2000) 275–286

[6] Bollobas, B., Borgs, C., Chayes, J., Kim, J.H., Wilson, D.B.: The scaling window of the 2-SAT transition. Random Structures and Algorithms, (2001) 201–256

[7] Borning, A., Freeman-Benson, B., Wilson, M.: Constraint hierarchies. Lisp and Symbolic Computation, 5(3), (1992) 223–270

[8] Cha, B., Iwama, K., Kambayashi, Y., Miyasaki, S.: Local search for Partial MAX-SAT. In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), (1997) 263–265

[9] Cheesman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), 331–337

[10] Climer, S., Zhang, W.: Searching for backbones and fat: a limit-crossing approach with applications. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02), (2002) 707–712

[11] Dubois, O., Dequen, G.: A backbone-search heuristic for efficient solving of hard 3-SAT formulæ. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), (2001) 248–253

[12] Freuder, E., Wallace, R.: Partial constraint satisfaction. Artificial Intelligence, 58(1), (1992) 21–70

[13] Menaï, M.B., Batouche, M.: Efficient initial solution to extremal optimization algorithm for weighted MAXSAT problem. In Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2003), LNAI 2718, Springer, (2003) 592–603

[14] Menaï, M.B.: Solution reuse in Partial MAX-SAT Problem. In Proceedings of IEEE International Conference on Information Reuse and Integration (IRI-2004), (2004) 481–486

[15] Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. In Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92), (1992) 459–465

[16] Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L.: Determining computational complexity from characteristic 'Phase Transition'. Nature, 400 (1999) 133–137

[17] Parkes, A.J.: Clustering at the phase transition. In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), (1997) 240–245

[18] Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), (1994) 337–343

[19] Singer, J., Gent, I.P., Smaill, A.: Backbone fragility and the local search cost peak. Journal of Artificial Intelligence Research, 12, (2000), 235–270

[20] Slaney, J., Walsh, T.: Backbones in optimization and approximation. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), (2001) 254–259

[21] Szedmak, S.: How to find more efficient initial solutions for searching. RUTCOR Research Report 49-2001, Rutgers Center for Operations Research, Rutgers University, Piscataway, NJ, USA, (2001)

[22] Telelis, O., Stamatopoulos, P.: Heuristic backbone sampling for maximum satisfiability. In Proceedings of the 2nd Hellenic Conference on Artificial Intelligence, (2002) 129–139

[23] Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), (2003) 1173–1178

[24] Xu, H., Rutenbar, R.A., Sakallah, K.: sub-SAT: A formulation for relaxed boolean satisfiability with applications in routing. In Proceedings of the International Symposium on Physical Design (ISPD-02), (2002) 182–187

[25] Zhang, W., Rangan, A., Looks, M.: Backbone guided local search for maximum satisfiability. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), (2003) 1179–1186

[26] http://dimacs.rutgers.edu/Challenges/

[27] http://www.informatik.tudarmstadt.de/AI/SATLIB