Cooperation control in Parallel SAT Solving: a Multi-armed Bandit Approach

Nadjib Lazaar, Youssef Hamadi, Said Jabbour, Michèle Sebag

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Cooperation control in Parallel SAT Solving: a Multi-armed Bandit Approach.

Nadjib Lazaar — Youssef Hamadi

Said Jabbour — Michèle Sebag

N° 8070

September 2012

Domaine 2

ISSN 0249-6399 ISRN INRIA/RR--8070--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Cooperation control in Parallel SAT Solving: a Multi-armed Bandit Approach.

Nadjib Lazaar [*] , Youssef Hamadi [†]
Said Jabbour [‡] , Michle Sebag [§]

Domaine : Algorithmique, programmation, logiciels et architectures
Équipes-Projets MSR-INRIA

Rapport de recherche n° 8070 — September 2012 — 15 pages

Abstract: In recent years, Parallel SAT solvers have leveraged with the so called Parallel Portfolio architecture. In this setting, a collection of independent Conflict-Directed Clause Learning (CDCL) algorithms compete and cooperate through Clause Sharing. However, when the number of cores increases, systematic clause sharing between CDCLs can slow down the search performance. Previous work has shown how the efficiency of the approach can be improved through controlling dynamically the amount of information shared by the cores[11], specifically the allowed length of the shared clauses. In the present paper, a new approach is used to control the cooperation topology (pairs of units able to exchange clauses). This approach, referred to as Bandit Ensemble for parallel SAT Solving (BESS), relies on a multi-armed bandit formalization of the cooperation choices. BESS is empirically validated on the recent 2012 SAT challenge benchmark.

Key-words: Parallel SAT solvers; portfolio approach; Multi-Armed Bandit problem.

[*] INRIA-Microsoft Research Joint Centre, 28 rue Jean Rostand 91893 Orsay Cedex, France, mail: nadjib.lazaar@inria.fr
[†] Microsoft Research,7 J J Thomson Avenue, Cambridge CB3 0FB, United Kingdom, mail: youssefh@microsoft.com
[‡] CRIL-CNRS, Université d'Artois Rue Jean Souvraz SP18, F-62307 Lens Cedex, France, mail: jabbour@cril.fr
[§] Université Paris-Sud, CNRS, LRI - Bat. 490, 91405 Orsay Cedex, France, mail: sebag@lri.fr

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Contrle de coopration dans une rsolution SAT parallle : Une approche Bandit Manchot

Résumé : Ces dernires annes, les solveurs SAT exploite de plus en plus les architectures parallles, dites de portfolio. Dans ce cadre, un ensemble de procdures CDCL (pour Conflict-Directed Clause Learning en anglais) sont en concurrence et en coopration travers un partage de clause. Cependant, lorsque le nombre de core augmente (unit de traitement), le partage systmatique des clauses apprises durant le procd CDCL peut ralentir les performances du solveur parallle. Les prcdents travaux ont montr que l'efficacit de l'approche de partage de clause peut tre amliore avec un contrle dynamique du nombre de clause partag par les cores cite HamadiJS09a, en particulier la taille maximale des clauses autorise. Dans cet article, une nouvelle approche est propose pour contrler la topologie de coopration (paires d'units autorises l'change de clauses). Cette approche, appele Ensemble de Bandit pour une rsolution SAT Parallle (BESS, Bandit Ensemble for parallel SAT Solving en anglais). L'approche de coopration BESS s'appuie sur une formalisation de Bandit manchot. BESS est valide empiriquement sur la rcente comptition SAT 2012.

Mots-clés : Solveurs SAT parallles; approche portfolio; problme du Bandit Manchot.

1 Introduction

The widespread adoption of modern SAT solvers based on the Conflict-Directed Clause Learning (CDCL) is the result of the efficiency gains made during the last decade. Nowadays, industrial problems with hundreds of thousands of variables and millions of clauses are routinely solved. However, many new application domains with instances of increasing size and complexity are coming to challenge modern solvers. For instance, in the last SAT competitions, many instances coming from Computational Biology remained unsolved. Fortunately for the domain, multi-core based parallel processing capabilities are now on every desktop. It then becomes legitimate to consider parallelization as a way to leverage existing CDCLs in order to efficiently meet the requirements of new application domains.

This technological shift has restarted research into parallel SAT solving, and many solvers have been presented since the early 2000. For historical reasons, many of them are based on the divide-and-conquer methodology [5, 6]. However, the most successful ones (regularly winning parallel tracks in SAT competitions) exploit the Parallel Portfolio architecture where a set of independent CDCLs solver compete and cooperate through Clause Sharing [12, 4]. Systematic clause sharing among the CDCLs however hardly scales up with the number n of cores, as n to $n-1$ communications slow down the search performance of the whole system. Previous work has shown how the efficiency of the approach can be improved through controlling dynamically the amount of information shared by the cores, specifically the allowed length of the shared clauses [11].

The present paper is concerned with controlling the cooperation network, that is, selecting the cores referred to as emitter cores that are allowed to send clauses to any given core called receiver core. The presented approach is based on a Multi-Armed Bandit (MAB) formalization of the emitter selection. The challenge is twofold. On the one hand, the relevance of an emitter core w.r.t. the receiver core, referred to as emitter reward, must be defined in order to select the best emitter cores relatively to a receiver core. The selection relies on the famed MAB Upper Confidence Bound algorithm (UCB) [3]. On the other hand, such rewards do not follow a static distribution; every core explores the search landscape and its production of clauses is bound to vary along its search. The UCB criterion is accordingly extended to accommodate non-stationary distributions. The proposed approach, a *Bandit Ensemble for parallel SAT Solving* (BESS), is empirically validated on the 2012 SAT challenge benchmark.

The paper is organized as follows. Section 2 briefly reviews related work and introduces formal background on modern parallel SAT solvers and Multi-Arm Bandits. Section 3 describes the BESS algorithm. Section 4 reports on the experimental setting and discusses the validation results. The paper concludes with some perspectives for further research.

2 Related work and formal background

This section briefly discusses the state of the art in parallel SAT solving, focussing on portfolio-based parallel approaches. For the sake of containedness, the multi-armed bandit framework is also presented together with the UCB algorithm [3].

2.1 Modern Parallel SAT Solvers

Modern SAT solvers extend the original Davis, Putnam, Logemann and Loveland procedure (DPLL) [8] with important components such as CDCL for conflict directed clause learning, restart policy, activity based heuristics, and pruning of the database of learnt clauses [15]. In these settings, when a conflict arises during the inference step, the implication graph converges on a conflict vertex capturing a variable assignment. This implication sequence is analyzed to determine the unsatisfying variable assignments that explain the conflict. This explanation is used in turn for intelligent backtracking, clause learning and the adjustment of the variable selection heuristics (i.e., variables activity).

Gradsat [5], the first parallel SAT solver based on a a CDCL, extends the zChaff solver using a master-slave model. It implements guiding-paths to divide the search space of a problem using a set of unit clauses. Moreover, learned conflict clauses can be exchanged between all slaves if they are smaller than a predefined size limit. Using a similar technology we can point to MiraXT [14], and pMiniSat [6].

ManySAT [12] and Plingeling [4] exploit a different solving strategy. Unlike the divide and conquer approach, these solvers use a portfolio approach which lets several sequential CDCLs complete and cooperate to solve the original instance. These CDCLs can differ from each other through complementary and/or different restart strategies, activity and polarity variables heuristics, clause-learning schemes, etc. The Plingeling solver presents a master-slave communication with only unit clauses exchange, whereas in ManySAT, all processing units share and exchange learned clauses up to some size limit. These limits can be static [12] or dynamic [11].

2.2 Portfolio-based approaches

In [10], the authors explore the diversification and intensification principles in a portfolio-based parallel SAT solver. The challenging question here is to find a good diversification/intensification tradeoff, enforcing an efficient division of labor between the different units and the search space they explore. A study of this tradeoff is proposed by defining two roles for the computational units. Some units are classified as masters and perform an original search strategy, ensuring diversification. The remaining units are classified as slaves that intensify their master's strategy.

The control-based clause sharing policy *aimdTQ* was first presented by [11]. This dynamic clause sharing policy uses an AIMD (i.e., Additive Increase/Multiplicative Decrease) feedback control-based algorithm used in TCP congestion avoidance. The *aimdTQ* policy adjusts dynamically the size of shared clauses between any pair of processing units. It maintains an overall number of exchanged clauses (i.e., throughput) and exploits the activity-based quality of shared clauses.

This algorithm has demonstrated that the efficiency of the parallel portfolio approach can be improved through dynamically controlling the amount of information shared by the units, specifically the allowed length of the shared clauses.

This approach however hardly scales up when the number of cores increases, due to the communication costs. In such cases, it becomes necessary to control which cores are allowed to send information to any other core.

2.3 Multi-armed bandit framework

How to control the communication and clause sharing among cores is formalized as an exploration vs exploitation (EvE) dilemma. A formal setting for EvE is the Multi-Armed Bandit problem (MAB), pertaining to the field of Game Theory [13].

The MAB problem involves K independent arms a.k.a. options. The i-th arm is characterized by its fixed reward probability p_i in $[0, 1]$. In each time step t, the arm selection strategy selects some arm j; with probability p_j it gets reward $r_t = 1$, otherwise $r_t = 0$.

The quality of an arm selection strategy is measured after its regret, defined as its loss compared to the optimal strategy, playing the arm with maximal reward p^* in each time step. Formally, the regret after N time steps is defined as:

$$\mathcal{L}(N) = Np^* - \sum_{t=1}^{N} r_t$$

The *Upper Confidence Bound* algorithm devised by Auer et al. [3] maintains two indicators for each i-th arm: the number of times it has been played up to time t, noted $n_{i,t}$ and the average corresponding reward noted $\hat{p}_{i,t}$. Subscript t is omitted when clear from the context. The UCB1 strategy selects in each time step the arm maximizing the quantity below:

$$\hat{p}_{j,t} + \sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{j,t}}}$$

where the left term $\hat{p}_{j,t}$ enforces the exploitation (favoring the arm with best empirical reward) and the right term $\sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{j,t}}}$ takes care of the exploration: each arm is selected infinitely many times as t goes to infinity; however the lapse of time between two selections of some under-optimal arm increases exponentially. The UCB algorithm provides optimal regret guarantees, logarithmically increasing with the number N of time steps (to be contrasted with the linear regret achieved by ϵ-greedy approaches).

Interestingly, the exploration vs exploitation dilemma is at the core of many portfolio-based approaches. For instance, [9] handle the algorithm selection problem as a MAB problem, where the goal is to select the algorithm most able to solve a given sequence of problem instances. Likewise, [7] address the *Adaptive Operator Selection* (AOS) issue in Evolutionary Algorithms as a MAB problem, where the goal is to select the operator which maximizes the expectation of the fitness improvement.

Note that in the latter context, the challenge is also to deal with a non-stationary setting: what is the best-suited evolutionary operator depends on the current phase of search. In the considered cooperation control problem, we likewise face a non-stationary setting: the reward associated to a core varies along search. whereas the standard MAB setting only considers stationary reward distributions. [7] thus extended the standard MAB criterion (which only considers stationary reward distributions) through estimating indicators $\hat{p}_{j,t}$ (respectively $n_{i,t}$) from respectively the rewards (resp. the number of selections) observed in the last W time steps, where the window length W is a user-supplied parameter.

3 Bandit Ensemble for Parallel SAT Solving: Overview

This section presents the Bandit Ensemble for Parallel SAT Solving (BESS) algorithm. As mentioned, parallel SAT solving needs to control more finely the information ex-

changed between the cores when considering large-scale parallel architectures: beyond the upper-bound on the length of the shared clauses [11], the communication architecture needs be dynamically controlled as well. The BESS algorithm achieves decentralized adaptive decision making: for each core, the selection of its emitter cores is dynamically achieved by a multi-armed bandit, referred to as individual MAB.

3.1 Individual MAB

An individual MAB is thus attached to a fixed receiver core. Let N denote the overall number of cores, and let n denote the fixed number of emitter cores allowed to share clauses (with a size up to L) with the receiver core[1]. The density $\rho = \frac{n}{N.(N-1)}$ of the communication topology is thus fixed (i.e., figure1).

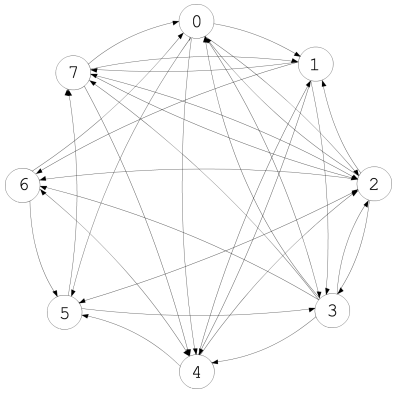

Figure 1: A directed graph density with $(N, n, \rho) = (8, 4, 0.14)$.

The question thus is to determine in each time period whether a given emitter core (naturally, distinct from the receiver core) is allowed to send clauses to the receiver; if this is the case, the emitter core is referred to as *alive*; otherwise it is *sleeping*.

Every K conflicts ($K = 25,\ 50,\ 100$ and 200 in the experiments), the decision of which emitters should be alive or sleeping is reconsidered. The lapse of time between two consecutive decision makings is referred to as *generation*.

Each individual bandit is a four-step process, which exploits the only information available to the receiver, that is, the one coming from the alive emitters. The individual bandit thus:

1. Updates the average cumulative reward associated to each alive emitter (since it has last been awakened) in the last generations;
2. Updates the reward threshold;
3. Determines the alive emitters with worse average cumulative reward, and compares these individual rewards with the threshold reward;
4. If the worst individual reward is confidently higher than the threshold, then basically all alive emitters are relevant in the sense that they are better than average;

[1]The upper-bound L on the size of the shared clauses is fixed in the remainder of the paper, as well as the number n of emitters. Further work is concerned with adaptively adjusting parameters L and n along the SAT search.

5. Otherwise, each worse alive emitter is sent to sleep, and the emitter that was sleeping for the longest period of time is awakened (set alive).

Let us describe the MAB approach and each step of the individual bandit on alg.1 and alg.2.

Algorithm 1: BESS

Input : A CNF formula $\mathcal{F}$, N cores, n alive emitters, L clause limit size
Output: A solution s of $\mathcal{F}$ or $unsat$ if $\mathcal{F}$ is not satisfiable

1 $s \leftarrow \emptyset$
2 **shared** $finish \leftarrow false$
foreach $i \in 0..N-1$ **do in parallel**
4 $\quad s \leftarrow MAB_i(\mathcal{F}, N, n, L)$

return s

Algorithm 2: $MAB_i(\mathcal{F}, N, n, L)$

1 $N_{set} \leftarrow \{0, \ldots, i-1, i+1, \ldots, N-1\}$, $A_b \leftarrow rand(N_{set}, n)$, $S_b \leftarrow N_{set} \backslash A_b$
2 $k, \tau_i \leftarrow 0 \qquad \Delta_i, R_i \leftarrow \emptyset$

while $true$ **do**
4 $\quad (\Delta_i^{k+1}, s) \leftarrow search(\mathcal{F}, \Delta_i, R_i)$
5 $\quad$ **if** $s \neq \emptyset$ **then** $finish \leftarrow true$ **return** s
6 $\quad$ **if** $finish$ **then break**
7 $\quad$ **foreach** $j \in N_{set}$ **do** $send(\Delta_i^{k+1}, L)$
8 $\quad$ **foreach** $j \in A_b$ **do** $R_i^{k+1} \leftarrow R_i^{k+1} \cup receive_j()$
9 $\quad (\Delta_i, R_i) \leftarrow (\Delta_i \cup \Delta_i^{k+1}, R_i \cup R_i^{k+1})$
10 $\quad r_i(A_b) \leftarrow reward(R_i^{k+1})$
11 $\quad \tau_i \leftarrow threshod\_update(\tau_i, r_i(A_b))$
12 $\quad (A_b, S_b) \leftarrow swap(r_i(A_b), \tau_i)$

14 **return** $\emptyset$

BESS algorithm (Algo.1) takes a CNF formula $\mathcal{F}$ for a parallel solving on N cores (from 0 to $N-1$) which generates N individual MAB. The cooperation-control approach is presented in Algo.2. This algorithm returns (sat/$unsat$) if it finishes, in which case all other units end by checking the *finish* shared variable (algo.2 line 6).
For a given core i, first we select randomly n alive emitters (i.e., A_b for *alive bandits*). The remaining and/or sleeping ones are in S_b.
The search starts with an initial state where the proper learnt clauses Δ_i and the foreign ones R_i are empty (i.e., $(\Delta_i, R_i) = (\emptyset, \emptyset)$). For each iteration, a CDCL procedure is

called (line 4) throughout a generation defined in term of number of conflicts (K conflicts). After each generation, the pair (Δ_i, R_i) evolves (line 9). We note Δ_i^{k+1} (resp. R_i^{k+1}) the learnt clauses (resp. received clauses) in the $k+1$ iteration.
On line 7 of alg.2, the unit i sends its proper learnt clauses Δ_i^{k+1} to each other unit j according to the clause size limit L. After that, the individual MAB receives all foreign clauses from the alive emitters (line 8). Based on the received clauses, the $reward$ and $threshold\_update$ functions update, respectively, the cumulative reward $r_{i\leftarrow j}$ and the reward threshold τ_i associated to the receiver core i (more details are in presented next sections 3.2, 3.3 and 3.4).
Based on reward and threshold values, the $swap$ function on line 12 swaps each worse alive emitter to sleep state and select the emitter with the longest sleeping time to be swapped to alive state.

3.2 Instant Rewards

In a given generation g, each core receives clauses of length up to the limit L from the alive emitters. In this section, we define the instant reward of each alive emitter according to different clause quality measurements. In the following, we note $R_{i\leftarrow j}^g$ (resp. $r_{i\leftarrow j}^g$) the received clauses (resp. the instant reward) from the alive emitter j to the receiver core i in generation g :

Size-based instant reward

Each clause c of size s removes 2^{V-s} possible instantiations from the search space, if V denotes the total number of variables of the current SAT formula $\mathcal{F}$. A first clause quality measurement is:

$$\mathcal{Q}_{size}(c) = -log(1-2^{-s})$$

Where the instant reward is set to

$$r_{i\leftarrow j}^g = \sum_{c\in R_{i\leftarrow j}^g} \mathcal{Q}_{size}(c)$$

LBD-based instant reward

As introduced in [2], the Literals Blocks Distance (LBD) is a way to measure the quality of learnt clauses. During search, each decision level produces a number of unit propagations. For a given clause, a "literal block" represents all literals of the same level. These literals have a chance that they are linked with direct dependencies from a semantic point of view. In practice, this measurement was introduced as a way to manage the learnt clause database of GLUCOSE solver [2].

Here, each receiver core computes the LBD of each received clause (noted $lbd(c)$ as LBD value of c clause). We propose an LBD-based clause quality as:

$$\mathcal{Q}_{LBD}(c) = \frac{\mathcal{Q}\_size(c)}{lbd(c)}$$

The instant reward is set to

$$r_{i\leftarrow j}^g = \sum_{c\in R_{i\leftarrow j}^g} \mathcal{Q}_{LBD}(c)$$

(Size/LBD)-based clause reward

It is established that clauses with the smallest size and/or LBD are more interesting. Here, we combine the two precedent quality measurements to produce a (Size/LBD)-based clause quality

$$\mathcal{Q}_{Size/LBD}(c) = -log(1 - 2^{-lbd(c)})$$

with an instant reward as

$$r^g_{i\leftarrow j} = \sum_{c \in R^g_{i\leftarrow j}} \mathcal{Q}_{SIZE/LBD}(c)$$

AH-based clause reward

In [11], a clause quality is defined using the activity of literals at the basis of VSIDS heuristic [15] (AH for Activity Heuristic). This heuristic allows the literals with greatest activity to be used in most of the recent conflicts. Indeed, when a conflict occurs, the activity of the literal that appears in the learnt clause is incremented. The most active literals are those related to the current part of the search space. Consequently, the clause quality measurement proposed in [11] exploits these VSIDS activities to quantify the relevance of a clause. Let us take A_i^{max} the current maximal activity of the receiver core i, and $A_i(x)$ the current activity of a given literal x. We note $\mathcal{L}(c)$ the set of active literals of a a given clause c:

$$\mathcal{L}_i(c) = \{x : \ x \in c \ s.t. \ A_i(x) \leq \frac{\mathcal{A}_i^{max}}{2}\}$$

We note also $\mathcal{P}_{i\leftarrow j}$ the set of clauses received by i from j with at least $\frac{|c|}{3}$ active literals:

$$\mathcal{P}_{i\leftarrow j} = \{c : \ c \in R^g_{i\leftarrow j} \ s.t. \ |\mathcal{L}_i(c)| \geq \frac{|c|}{3}\}$$

The AH-based quality of clauses sent by j to i is given as:

$$\mathcal{Q}_{AH} = \frac{|\mathcal{P}_{i\leftarrow j}|}{|R^g_{i\leftarrow j}| + 1}$$

Using this new quality measurement, the instant reward is set to:

$$r^g_{i\leftarrow j} = \mathcal{Q}_{AH} \times \sum_{c \in R^g_{i\leftarrow j}} -log(1 - 2^{|c|})$$

CAH-based clause reward

In this section, we propose a (Continuous Activity Heuristic)-based quality. Let us recall that $A_i(x)$ (resp. $\mathcal{A}_i^{max}$) the activity of the literal x (resp. activity max) according to the receiver core i:

$$\mathcal{Q}_{CAH}(c) = \frac{1}{|c|} \sum_{x \in c} f(\frac{A_i(x)}{\mathcal{A}_i^{max}})$$

With f a particular sigmoid function ($f : [0\ 1] \rightarrow [0\ 1]$) which starts with small beginnings and accelerates to reach a climax point (fig.2):

$$f(x) = (1/(1 + e^{\alpha(2x-1)}) - \beta) \times \gamma$$

with α a constant, $\beta = \frac{1}{(1+e^{\alpha})}$ and $\gamma = \frac{(2+e^{-\alpha}+e^{\alpha})}{(e^{\alpha}-e^{-\alpha})}$

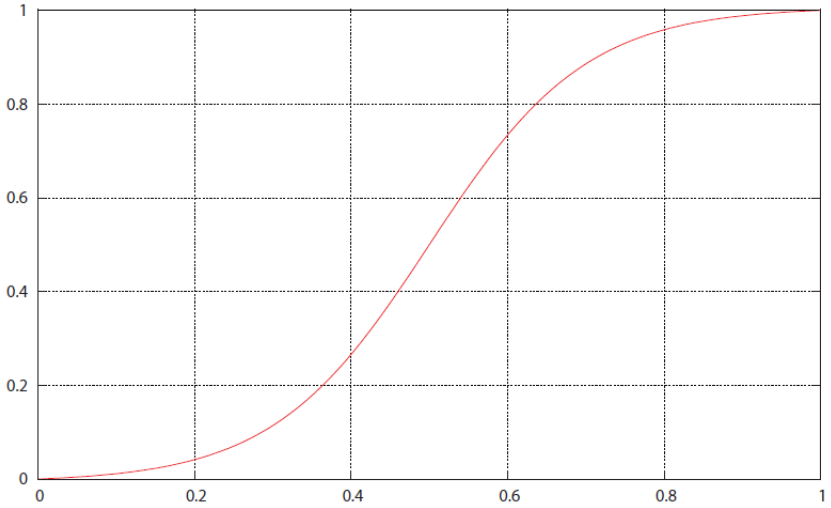

Figure 2: Sigmoid function $f(x)$ with $\alpha = 5$

Now, using this new quality measurement, we set the instant reward as:

$$r^g_{i \leftarrow j} = \sum_{c \in \mathcal{R}^g_{i \leftarrow j}} f(\mathcal{Q}_{CAH}(c))$$

3.3 The cumulative reward

The instant reward $r^g_{i \leftarrow j}$, defined in precedent section, is used to update by relaxation the cumulative reward estimate of the j-th alive emitter, where parameter λ is the relaxation rate:

$$r_{i \leftarrow j} = (1 - \lambda) r_{i \leftarrow j} + \lambda r^g_{i \leftarrow j}$$

3.4 Threshold update

In this section, we present different methods (M1 to M4) to update the reward threshold τ_i (associated to the receiver core i).

M1: *first generation reward average*. Here, the threshold is determined as a static value calculated at the beginning as the average reward of the first generation over all emitters,

$$\tau^g_i = \frac{1}{n} \sum_{j \in A_b} r^1_{i \leftarrow j}$$

M2: *Previous generation reward average*. At generation g, the threshold τ^g_i is determined as the average reward of the previous generation $(g-1)$ over all emitters,

$$\tau^g_i = \frac{1}{n} \sum_{j \in A_b} r^{g-1}_{i \leftarrow j}$$

M3: *Cumulative reward average of new alive emitters.* Since obviously only the rewards of alive emitters are known, the threshold is updated from the reward of the alive emitters *in the first generation after they have been awakened*. Let n_τ, initialized to 1, be the number of times a new emitter has been set alive since the beginning of the search. Let δ_τ be 1 if there is a new alive emitter j in generation g, with $r^g_{i\leftarrow j}$ its instant reward; then

$$\tau^g_i = \frac{n_\tau * \tau^g_i + r^g_{i\leftarrow j}}{n_\tau + \delta_\tau}$$

M4: *Weighted threshold.* The reward threshold τ_i is determined by relaxation on the cumulative average reward over all emitters,

$$\tau^g_i = (1-\lambda)\tau^g_i + (\lambda \times \frac{1}{n} \sum_{j \in A_b} r^g_{i\leftarrow j})$$

The comparison between the reward threshold and the worst average reward relies on the Hoeffding bound. Let us remind that the probability for an empirical average $\hat{p}_n$ built from n trials (all ranging in interval $[a, b]$) to be less than the true average p by a margin ϵ is bounded as:

$$Pr(\hat{p}_n < p - \epsilon) < exp\left(-\epsilon^2 n/(b-a)^2\right)$$

One accordingly computes whether the probability for the minimum r_i over all alive emitters, to be less than τ^g_i by ϵ is less than η, where η is a parameter of the algorithm:

$$\tau_g - min_i\{r_i\} = \epsilon \qquad\qquad \epsilon > (b-a)\sqrt{\frac{log(\eta)}{n_\tau}}$$

and if this is the case, the worst alive emitter is sent to sleep and a sleeping emitter is awakened.

While the emitter to be awakened could be uniformly selected, one selects the emitter that was sleeping for the longest period of time, emulating a no-replacement selection and thus avoiding the chance of forgetting any emitter for long.

The behavior of the individual MAB using the precedent update-threshold methods is illustrated on a simple artificial problem.

Let the instant reward associated to the j-th emitter w.r.t. the receiver i in generation g be defined as

$$r^g_{i\leftarrow j} = sin(a_j \times g + b_j)$$

where parameters a_i and b_i, uniformly drawn in respectively $[-1, 1]$ and $[0, 2\pi]$, are attached to the i-th emitter.

In the first generation, n emitters are uniformly selected among the N ones, and set to alive. The figure m-threshold compares the cumulative reward regret using the four methods (i.e., M1 to M4) w.r.t. the oracle. Here, the oracle selects, at each generation, the best n emitters to set as alive. Over 50 generation, the weighted threshold method (i.e., M4) is better than the cumulative and static ones (i.e., M1, M2 and M3).

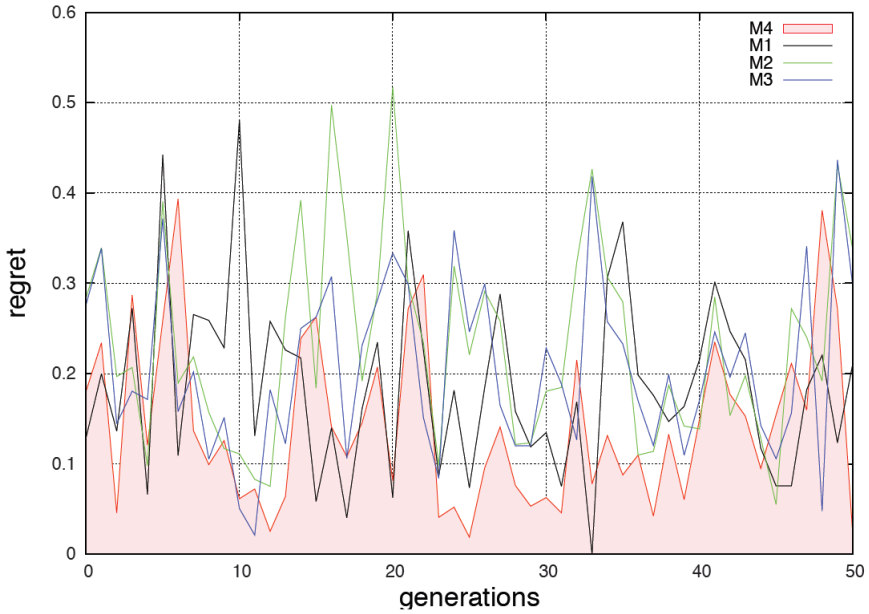

Figure 3: Threshold update methods comparison

4 Evaluation

This section reports on the empirical evaluation of the BESS algorithm, which is implemented on the top of the ManySAT parallel SAT solver. ManySAT, chosen as it won the parallel track on 2008 SAT race, is a portfolio approach involving several sequential CDCLs (which include all the classical features like two-watched-literal, unit propagation, activity-based decision heuristics, lemma deletion strategies, and clause learning). In addition to the classical first-UIP scheme, these CDCLs incorporate a new technique which extends the classical implication graph used during conflict-analysis to exploit the satisfied clauses of a formula [1]. They are differentiated in many ways (e.g., the use of different and complementary restarts, learning schemes...).

4.1 Experimental setting

Our tests were conducted on two platforms, 8-core Intel Xeon machines with 16GB of RAM running at 2.33GHz, and 32-core AMD Opteron Proc. 6136 machines with 64GB of RAM running at 2.4GHz. We used the 588 "Application" SAT+UNSAT instances of the latest SAT-Challenge 2012. The CPU time limit was set to $30mn$ CPU per core, hence a total of 4 hours on the first (resp. 16 hours on the second) platform. The number n of alive emitters is set to 4 (resp. 16) for the 8-cores (resp. 32-cores) architecture with a shared clause limit size $L = 8$.

Figure 4 shows the results on 8-cores, the time period K was set to 25 after a few preliminary tests.

4.2 Experimental validation

Fig. 4 displays the comparative performance of the different BESS settings and ManySAT, that is the time in seconds needed to solve a given number of instances on the 8-core architecture. Using a different time period g (i.e., 25, 50, 100 and 200), we compare the different reward functions:

R=1: Size-based reward

R=2: LBD-based reward

R=3: (Size/LBD)-based reward

R=4: AH-based reward

R=5: CAH-based reward

ManySAT uses a complete communication topology, every core sharing its clauses with every other core. The number n of alive emitter cores is set to 4, selected by BESS. As a sanity check, the performance obtained when the alive emitters are uniformly selected in each time period (legend *Random*) is also reported.

Here, both ManySAT and BESS, with a CAH-based reward (R=5), significantly outperform *Random*. Let us take part(a) of Fig.4, ManySAT and BESS coincide at the beginning of the curve, that is for the "easy" problems; for more difficult problems (resolution time $>$ 4,000 sec.), BESS with CAH-based reward moderately but significantly outperforms ManySAT (solving 7 more problem instances).

A much clearer picture is obtained on the 32-core architecture. Fig. 5 displays a comparative performance of BESS with the best setting get from 8-cores results (i.e., CAH-based reward, $|g| = 25$) w.r.t., ManySAT and the random mode. The number n of alive emitter cores is set to 16, selected by BESS. Here, *Random* moderately but significantly outperforms ManySAT, which confirms that the communication overload is detrimental to the resolution efficiency. The accuracy of the BESS process is demonstrated as BESS significantly outperforms ManySAT and *Random*, solving 30 more instances than ManySAT in the imparted time. In the meanwhile, BESS is also faster than ManySAT 2.0 (requiring less than 20,000 s. versus more than 50,000 s to solve 300 problems).

5 Conclusion

This paper, aimed at the scalability of parallel portfolio-based SAT, presents an ensemble Bandit-based approach which outperforms the state of the art ManySAT 2.0 parallel algorithm, by replacing a complete communication topology with an adaptively adjusted one.

More generally, this paper shows how a Multi-Armed Bandit approach can be used to adaptively control a communication topology in a decentralized way, yielding an efficient trade-off between i) the cooperation of a massive number of cores; ii) an affordable overall communication load among the cores; iii) an efficient communication topology adjustment. The main lesson learned concerns the interaction between the reward definition and the periodicity of the selection decisions.

As mentioned, a much simplified MAB framework has been considered in BESS insofar. The MAB framework however offers room to extend the BESS scope to adaptively determining i) the appropriate number of emitters for a given receiver; ii) the maximum length of the shared clauses, again for a given receiver.

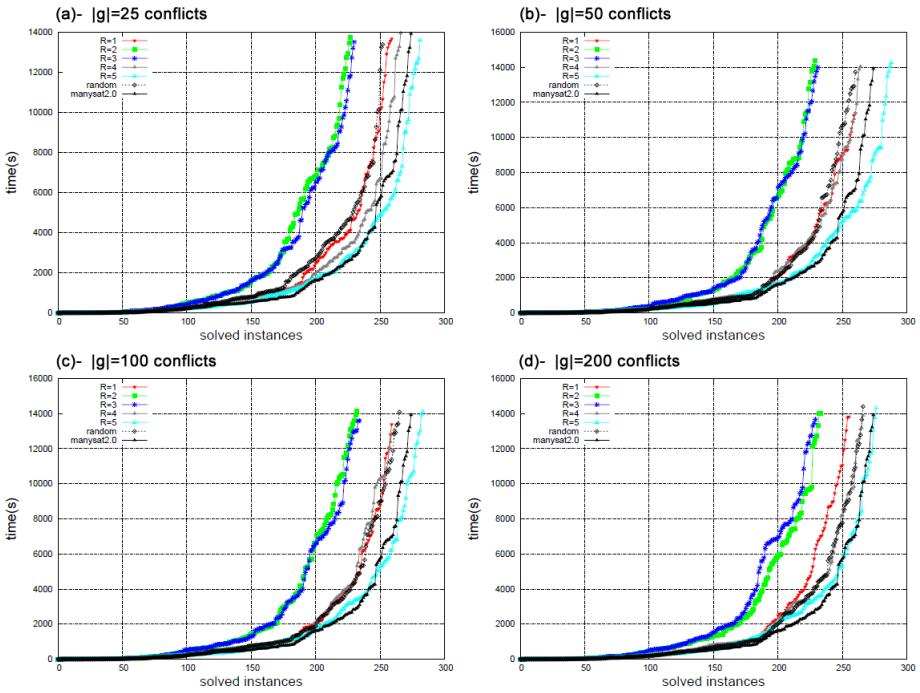

Figure 4: Results on 8 cores

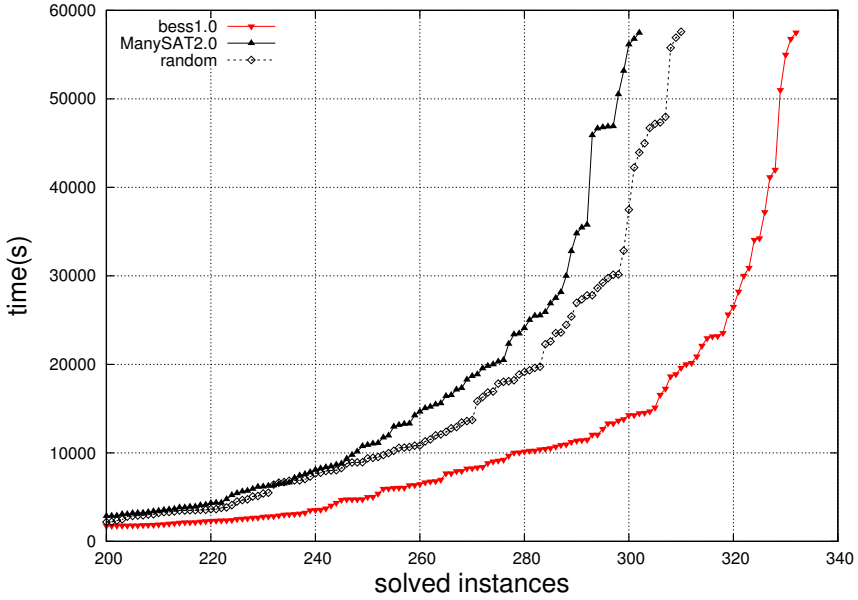

Figure 5: Results on 32 cores

References

[1] Gilles Audemard, Lucas Bordeaux, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. A generalized framework for conflict analysis. In *SAT*, pages 21–27, 2008.

[2] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *IJCAI*, pages 399–404, 2009.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[4] Armin Biere. Lingeling, plingeling, picosat and precosat. solver description, SAT-Race 2010. Technical report, 2010.

[5] Wahid Chrabakh and Richard Wolski. Gridsat: A chaff-based distributed sat solver for the grid. In *SC*, page 37, 2003.

[6] Geoffrey Chu and P. Peter J. Stuckey. PMiniSat: a parallelization of MiniSat 2.0. SATRace 2008, solver description. Technical report, 2008.

[7] Luís Da Costa, Álvaro Fialho, Marc Schoenauer, and Michèle Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *GECCO*, pages 913–920, 2008.

[8] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[9] Matteo Gagliolo and Jürgen Schmidhuber. Algorithm portfolio selection as a bandit problem with unbounded losses. *Ann. Math. Artif. Intell.*, 61(2):49–86, 2011.

[10] Long Guo, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Diversification and intensification in parallel sat solving. In *CP*, pages 252–265, 2010.

[11] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Control-based clause sharing in parallel sat solving. In *IJCAI*, pages 499–504, 2009.

[12] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Manysat: a parallel sat solver. *JSAT*, 6(4):245–262, 2009.

[13] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6:4–22, 1985.

[14] Matthew D. T. Lewis, Tobias Schubert, and Bernd Becker. Multithreaded sat solving. In *ASP-DAC*, pages 926–931, 2007.

[15] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *DAC*, pages 530–535, 2001.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399