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

Nadjib Lazaar
INRIA-Microsoft Research Joint Centre
rue d'Estienne d'Orves, Palaiseau, France
nadjib.lazaar@inria.fr

Youssef Hamadi
Microsoft Research
7 J J Thomson Avenue, Cambridge, UK
youssefh@microsoft.com

Said Jabbour
CRIL-CNRS, Université d'Artois
Rue Jean Souvraz, Lens, France
jabbour@cril.fr

Michèle Sebag
TAO, CNRS − INRIA − LRI
Université Paris-Sud, Orsay, France
sebag@lri.fr

Abstract

In recent years, parallel SAT solvers have leveraged with the so-called parallel portfolio architecture, where a set of independent conflict-directed clause learning algorithms compete and cooperate through clause sharing. This architecture however hardly scales up with the number of cores. In this paper, a dynamic multi-armed bandit approach is used to control the communication network (which cores are allowed to send clauses to a given core). The presented approach, referred to as Bandit Ensemble for parallel SAT Solving (BESS), is empirically validated on the recent 2012 SAT challenge benchmark.

1 Introduction

The widespread adoption of modern SAT solvers based on conflict-directed clause learning (CDCL) is the result of the efficiency gains made during the last decade. However, many new application domains with instances of increasing size and complexity are coming to challenge modern solvers. 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, particularly based on the divide-and-conquer methodology. However, the most successful ones exploit the parallel portfolio architecture where a set of independent CDCLs solver compete and cooperate through clause sharing [7, 2]. Systematic clause sharing among the CDCLs however hardly scales up with the number n of cores, as the $\mathcal{O}(n^2)$ 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 maximum length of the shared clauses [6].

The present paper is concerned with controlling the communication 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 network control. The challenge is twofold. The first issue is to design the appropriate reward, estimating the relevance of an emitter core w.r.t. the receiver core. The second issue is to adapt the MAB setting and the Upper Confidence Bound algorithm (UCB) [1] to a non-stationary framework. Indeed every core independently explores the search landscape; its production of clauses (and the worthiness thereof w.r.t. the receiver) are bound to vary along the search. The UCB criterion is accordingly extended to accommodate non-stationary distributions. The proposed approach, called

Bandit Ensemble for parallel SAT Solving (BESS), is validated on the 2012 SAT challenge benchmark, and comparatively assessed w.r.t. ManySAT.

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 [1].

Modern Parallel SAT Solvers. Modern SAT solvers extend the original Davis, Putnam, Logemann and Loveland procedure (DPLL) [3] with important components such as CDCL for conflict directed clause learning, restart policy, activity based heuristics, and pruning of the database of learnt clauses [9]. Unlike the divide and conquer approach, ManySAT [7] and Plingeling [2] solvers use a portfolio approach which lets several sequential CDCLs compete 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. In ManySAT, all cores share and exchange learned clauses up to some size limit [6]. 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.

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 [8].

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. [1] 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 $\hat{p}_{i,t}$. The UCB strategy selects in each time step the arm maximizing the quantity $\hat{p}_{j,t} + \sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{j,t}}}$ 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 decreasing 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, [5] 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, [4] address the *adaptive operator selection* (AOS) issue in stochastic optimization as a MAB problem, where the goal is to select the operator which maximizes the expectation of the objective improvement. Interestingly, the AOS problem also raises the non-stationary distribution issue: the worthiness of a stochastic perturbation operator varies along search. A related issue is to design the operator reward.

3 BESS: Bandit Ensemble for parallel SAT Solving

The goal of the presented work is to adaptively control the core communication in a large-scale parallel architecture. Specifically, the point is to select for each receiver core, the emitter cores

which are allowed to share the learnt clauses with the receiver, in a decentralized and adaptive manner. In the remainder of the paper, the maximum length L of the shared clauses is fixed. 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 any receiver core. How to adaptively adjust L and n along time is left for further work.

The BESS approach attaches a multi-armed bandit, referred to as individual MAB, to each receiver core. In the perspective of a given receiver, an emitter is *alive* in a time period iff it is allowed to send clauses to the receiver during this period; otherwise an emitter is *sleeping*. At the beginning of each time period (see below), the individual MAB decides whether an alive emitter will stay alive; otherwise, it becomes a sleeping emitter, and the oldest sleeping emitter is turned into an alive one.

In each time period, the individual MAB:
$*$ Computes the instant reward and updates the cumulative reward associated to each alive emitter;
$*$ Updates the *aliveness threshold*;
$*$ Turns every alive emitter into a sleeping one if its cumulative reward is lower than the threshold; in such a case, the sleeping emitter which has been sleeping for the longest time interval is awakened and set alive.

Time period. The length of a time period is measured in number K of conflicts (inconsistencies met by the SAT solver on the receiver core). A time-length of $K = 5,000$ conflicts was considered in [6]. In BESS, the time-length was set after a few trials, ensuring a sufficient correlation of the instant emitter rewards (see below) from one time period to the next time period ($K = 25$).

Instant Reward. The instant reward of an alive emitter with respect to a receiver is defined according to the continuous activity quality r_{CA}, inspired from the Variable State Independent Decaying Sum (VSIDS) heuristics [9]. Formally, for each literal x of a shared clause c, let $A_i(x)$ measure its VSIDS *with respect to* the i-th receiver, with $\mathcal{A}_i^{max}$ the maximum VSIDS. The instant reward of each clause c is defined as:

$$r_{CA}(c) = \frac{1}{|c|} \sum_{x \in c} f\left(\frac{A_i(x)}{\mathcal{A}_i^{max}}\right)$$

with f a non-linear bounded function, meant to discount the impact of clauses with low average VSIDS (the sigmoid is used in all experimental results). Eventually, the instant reward of each emitter core is computed by likewise averaging the instant reward of the emitted clauses: letting $\mathcal{R}_{i \leftarrow j}^t$ denote the set of clauses emitted by core i toward core j in time period t, we define:

$$r_{i \leftarrow j}^t = \sum_{c \in \mathcal{R}_{i \leftarrow j}^t} f\left(r_{CA}(c)\right)$$

Cumulative reward. The cumulative reward $r_{i \leftarrow j}$ is updated by relaxation from the instant reward; parameter λ is the relaxation rate accounting for the dynamic distribution of rewards.

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

Aliveness threshold. This threshold is used to reject the emitters which are no longer worth for the receiver: it estimates the average contribution of an emitter *with respect to the current* receiver. It is also updated by relaxation from the contribution of the youngest alive emitters (set A_{new}): $\tau_i^t = (1 - \lambda)\tau_i^{t-1} + (\lambda \times \frac{1}{|A_{new}|} \sum_{j \in A_{new}} r_{i \leftarrow j}^t)$

Every alive emitter is sent to sleep iff its cumulative reward is less than the aliveness threshold, and the emitter that was sleeping for the longest period of time is awakened.

In its present state, BESS only test the performance of each alive emitter comparatively to the aliveness threshold. The use of the overall Multi-Armed Bandit framework will be considered in further work (section 5).

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.

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 $L = 8$. As mentioned, the time period K was set to 25 after a few preliminary tests.

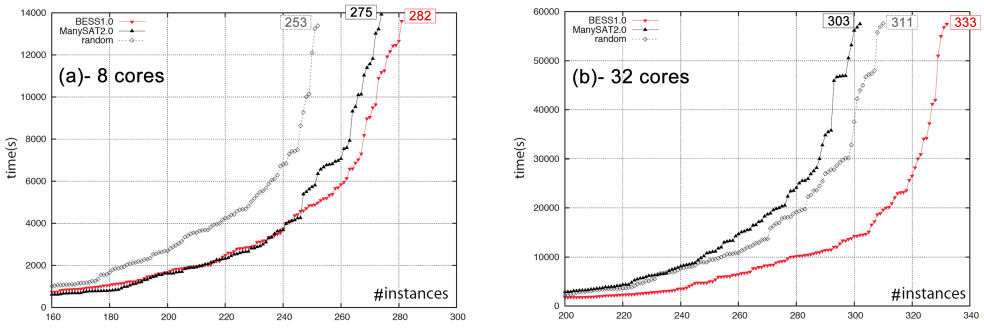

Figure 1: BESS: Number of instances solved *vs* computational cost on a 8 and 32-cores architecture, compared to ManySAT (full communication) and Random (random selection of n emitters for each receiver in each period time).

Fig. 1 displays the comparative performance of BESS and ManySAT, that is the time in seconds needed to solve a given number of instances, on the 8-core (Fig. 1.a) and 32-core (Fig. 1.b) settings. 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 (resp. 16) in the 8-core (resp. 32-core) setting, 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.

On the 8-core architecture, both ManySAT and BESS significantly outperform *Random*. 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 moderately but significantly outperforms ManySAT (solving 7 more problem instances).

A much clearer picture is obtained on the 32-core architecture. In this case, *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.

References

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

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

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

[4] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):25–64, October 2010.

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

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

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

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

[9] 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.