Reinforcement Learning and Reactive Search: an adaptive MAX-SAT solver

Roberto Battiti and **Paolo Campigotto** [1]

1 Introduction

This paper investigates Reinforcement Learning (RL) applied to online parameter tuning in Stochastic Local Search (SLS) methods. In particular, a novel application of RL is proposed in the Reactive Tabu Search (RTS) scheme, where the appropriate amount of diversification in prohibition-based local search is adapted in a fast online manner to the characteristics of a task and of the local configuration. The experimental tests demonstrate promising results on Maximum Satisfiability (MAX-SAT) instances when compared with state-of-the-art SLS SAT solvers, such us AdaptNovelty$^+$, rSAPS and gNovelty$^+$.

2 Reinforcement Learning for Reactive Tabu Search

This paper investigates a novel application of Reinforcement Learning in the framework of Reactive Tabu Search (RTS) proposed in [1]. Tabu Search (TS) is a prohibition-based search technique based on local search. At a given iteration some local search moves (e.g., variable flips in the case of the SAT) are *prohibited*, only a non-empty subset of them is *allowed*: the local search move executed at iteration t will not be allowed for the next T iterations, where T is the prohibition parameter. In this work, T is assumed to take values over the interval $[T_{min}, T_{max}]$.

RTS is a proposal to determine a dynamic value of the prohibition parameter which is appropriate to a specific instance and to the local characteristics of the fitness surface around the current configuration.

Among all the RL methods developed, we consider the Least-Squares Policy Iteration (LSPI) algorithm [4], a form of model-free approximate policy iteration using a set of training samples collected in any arbitrary manner. In [6], we present an off-line application of LSPI to tune the prohibition parameter, in particular by considering an application to the MAX-SAT problem. The parameter-tuning policy is modeled as a Markov Decision Process (MDP) where the states summarize relevant information about the recent history of the search, and a near-optimal policy is determined by using the LSPI method.

In this work, we consider an online version of the method to determine a critical algorithm parameter while the algorithm is running on a selected instance. The impact of different choices for designing the Markov states and the definition of the basis function for the approximation architecture are discussed.

The effect of changing the prohibition parameter on the algorithm's behavior can only be evaluated after a reasonable number of local moves. We therefore divide the algorithm's trace into *epochs* $(E_1, E_2, \dots)$ composed of a suitable number of local moves, and allow changes of T only between epochs.

The state at the end of epoch E_i is a collection of features extracted from the algorithm's execution up to that moment.

Assume n and m the number of variables and clauses of the input SAT instance, respectively. Let $f(\boldsymbol{x})$ the score function counting the number of unsatisfied clauses in the truth assignment $\boldsymbol{x}$.

Each state of the MDP is created by observing the behavior of the Tabu search algorithm over an epoch of $2{*}T_{max}$ consecutive variable flips.

In particular, let us define the following:

- $\boldsymbol{x}_{\text{bsf}}$ is the "best-so-far" (BSF) configuration *before* the current epoch;
- T_{f} is the current fractional prohibition value (the actual prohibition period is $T = \lfloor nT_{\text{f}} \rfloor$);
- $\overline{f}_{\text{epoch}}$ is the average value of f during the epoch;
- $\overline{H}_{\text{epoch}}$ is the average Hamming distance during the current epoch from the configuration at the beginning of the current epoch itself.

These variables have been chosen because of the Reactive Search paradigm's concern on the trade-off between diversification (the ability to explore new configurations in the search space by moving away from local minima) and bias (the preference for configurations with low objective function values). The compact state representation chosen to describe an epoch is the following triplet:

$$\boldsymbol{s} \equiv \left(\Delta f, \frac{\overline{H}_{\text{epoch}}}{n}, T_{\text{f}} \right), \qquad \text{where} \qquad \Delta f = \frac{\overline{f}_{\text{epoch}} - f(\boldsymbol{x}_{\text{bsf}})}{m}.$$

The first component is the mean change of f in the current epoch with respect to the best value; all components of the state have been normalized.

The actions set is composed by two choices: $\mathcal{A} = \{\text{increase}, \text{decrease}\}$, with the following effects:

$$T_{\text{f}} = \begin{cases} \max\{T_{\text{f}} \cdot 1.1,\ T_{\text{f}} + 1/n\} & \text{if } a = \text{increase} \\ \min\{T_{\text{f}}/1.1,\ T_{\text{f}} - 1/n\} & \text{if } a = \text{decrease} \end{cases} \tag{1}$$

Changes in T_{f} are designed in order to ensure variation of at least 1 in the actual prohibition period T. In addition, T_{f} is bounded between a minimum and a maximum value (0 and .2 in our experiments).

An alternative definition for the actions set consists of setting T_{f} from scratch by one of the 20 uniformly distributed values in the range $[0.01, 0.2]$:

$$T_{\text{f}} = 0.01 * i, \text{ where } i \in [1, 20] \tag{2}$$

The reward signal is given by the normalized change of the best value achieved in the observed epoch with respect to the "*best-so-far*" value *before* the epoch: $(f(\boldsymbol{x}_{\text{bsf}}) - f(\boldsymbol{x}_{\text{localBest}}))/m$.

[1] Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy, email: {battiti, campigotto}@disi.unitn.it

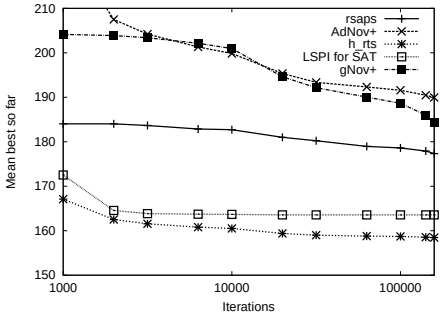

Figure 1. The comparison among our RL-based method and other SAT solvers.

For the case of the actions set defined via Eq. 1, we use the basis function set presented in [6].

If the actions set is defined by Eq. 2, assume action a being "set T_{f} to $0.01 * i$", $i \in [1, 20]$, and $\Phi_j(s, a)$ being the j-*th* entry for the considered basis function vector $\mathbf{\Phi}(s, a)$. We have:

$$\Phi_j(s,a) = \begin{cases} \Delta f & \text{if } j = 1 \\ \overline{H}_{\text{epoch}} & \text{if } j = 2 \\ \overline{H}_{\text{epoch}} \cdot \Delta f & \text{if } j = 3 \\ (\Delta f)^2 & \text{if } j = 4 \\ \overline{H}^2_{\text{epoch}} & \text{if } j = 5 \\ i/100 & \text{if } j = 5 + i \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The training phase is executed *online*, while solving a single SAT instance. This design choice implies that the best policy learnt by the SAT solver is not defined *a priori* by an off-line training phase over selected SAT instances, but it is determined by learning *while* the target optimization task is performed.

During an initial set up phase, 100 training examples for the input SAT instance are extracted to calculate the initial policy. Then, the solving phase is started. As soon as the search history provides a new example, it is added to the training set and the policy is updated.

3 Experimental results

For our tests, we use the benchmark described in [5], formed by MAX-3-SAT random instances with 500 variable and 5000 clauses. The T_{f} parameter has been bounded in $[0, .2]$.

To evaluate our novel MAX-SAT solver based on Reinforcement learning we report here a comparison with some of the best and famous SLS algorithms for MAX-SAT. In particular, the SLS techniques considered are the the AdaptNovelty$^+$ [7], the the RSAPS, a reactive version of SAPS [3], the H_RTS [1], and the gNovelty$^+$ [2].

For each algorithm, 10 runs with different random seeds are performed for each of the 50 instances taken from the benchmark set, for a total of 500 tests. Fig. 1 shows the average results as a function of the number of iterations (flips).

Fig. 1 indicates that our RL-based approach is competitive with the other existing SLS MAX-SAT solvers. In the experiment in Fig. 1, for our RL-based approach we consider the case where the update of the T_{f} value is performed by Eq. 1.

However, in Sec. 2 we presented two possible definitions for the action that updates the value of T_{f}:

1. T_{f} is increased/decreased by the value $1/n$ (see Eq. 1);

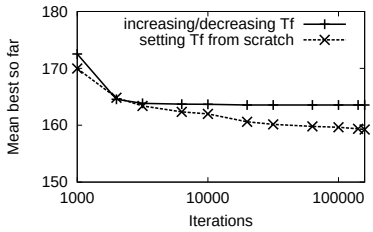

Figure 2. Performance of the two implemented actions for the update of the T_{f} value: increasing/decreasing vs setting T_{f} from scratch.

2. T_{f} is set from scratch via Eq. 2.

Fig. 2 compare the two hypotheses, showing an improvement in the second case. E.g., at iteration 100000 an improvement of 2.4% in the mean best-so-far value is registered. Setting the T_{f} value from scratch, our algorithm reaches the optimal performance of H_RTS. Furthermore, for the first hypothesis, a bigger increase/decrease of the T_{f} parameter has also been tested. In particular, we replaced the factor 1.1 in Eq. 1 by the value 1.3. However, in this case we obtain a little bit worse results.

4 Conclusions

This paper describes an application of Reinforcement Learning for the online tuning of the prohibition parameter in the Reactive Tabu Search algorithm. We discussed a couple of relevant architectural choices and presented preliminary experimental results. The results are promising: over the MAX-SAT benchmark considered our algorithm performs better than the gNovelty$^+$, which is a Gold Medal winner in the random category in the SAT 2007 competition and achieves results which are comparable with those obtained by the original RTS algorithm. These findings are confirmed by the additional experimental work not presented in this paper because of space limits.

REFERENCES

[1] R. Battiti and M. Protasi, 'Reactive search, a history-sensitive heuristic for MAX-SAT', *ACM Journal of Experimental Algorithmics*, **2**(ARTICLE 2), (1997). http://www.jea.acm.org/.

[2] C. Gretton D.N. Pahm, J.R. Thornton and A. Sattar, 'Advances in local search for satisfiability', in *20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007*, ed., John Thornton Mehmet Orgun, number 4830 in Lecture Notes in Computer Science, pp. 213–222. Springer, (2007).

[3] D.A.D. Tompkins F. Hutter and H.H. Hoos, 'Scaling and probabilistic smoothing: Efficient dynamic local search for sat', in *Proc. Principles and Practice of Constraint Programming - CP 2002 : 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13*, volume 2470 of *LNCS*, pp. 233–248. Springer Verlag, (2002).

[4] M.G. Lagoudakis and R. Parr, 'Least-Squares Policy Iteration', *Journal of Machine Learning Research*, **4**(6), 1107–1149, (2004).

[5] D. Mitchell, B. Selman, and H. Levesque, 'Hard and easy distributions of SAT problems', in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 459–465, San Jose, Ca, (July 1992).

[6] M. Brunato R. Battiti and P. Campigotto, 'Learning while optimizing an unknown fitness surface', in *Proceedings of the 2nd Learning and Intelligent OptimizatioN Conference (LION II), Trento, Italy, Dec 10-12, 2007*. Springer LNCS, in press, (2008).

[7] D.A.D. Tompkins and H.H. Hoos. Novelty$^+$ and adaptive novelty$^+$. SAT 2004 Competition Booklet. (solver description).