2E4-OS-09a-6

A Survey of Parallel Local Search for SAT

Alejandro Arbelaez*1 Philippe Codognet*2

*1JFLI / University of Tokyo *2JFLI-CNRS / UPMC / University of Tokyo

Local search algorithms are widely used to tackle combinatorial problems in several domains, one of the main features of these algorithms lies in the fact that they can tackle very large problems. Moreover, an interesting possibility is the potential speedup obtained by executing multiple copies of the sequential algorithm. In this paper, we review the current state-of-the art and future trends of parallel local search algorithms for the SAT problem, one the most important NP-complete problems. Generally speaking, these algorithms can be broadly classified into two categories: parallel portfolios, where several algorithms compete and cooperate to solve a given problem instance and multi-flip algorithms where several flips of the variables are performed at the same time.

1. Introduction

The Boolean satisfiability problem (SAT) consists in determining whether a given formula in *Conjunctive Normal Form* is satisfiable or not. The formula is a conjunction of clauses and each clause is a disjunction of variables (a literal or its negation). SAT solvers are widely used in several application domains, including: computational biology, software/hardware verification, planning, etc.

Today, parallel architectures provide an interesting opportunity to improve the performance of SAT solvers. The computational benefit of parallel SAT solving can be observed in both *capacity solving* and *speedup*. *Capacity solving* refers to the number of solved instances within a given time limit, and *speedup* refers to the ability of reducing the execution time (w.r.t. the sequential solver) to solve a given problem instance.

A classical manner to devise a parallel local search solver consists in executing multiple copies of different algorithms (or the same one with different random seeds) with or without cooperation. This approach is also known in the SAT literature as a parallel portfolio. Another way to parallelize the local search procedure consists in exploring the neighborhood in parallel at a cost of reconciling partial information in order to decide the best action. Not surprisingly, most researchers have focus their attention on the parallel portfolio approach as it provides two general advantages. First, it requires no extra work to implement, and second it has been theoretically and practically proven to be powerful in a wide range of domains; moreover the portfolio technique is not affected by the Amdahl's law. We recall that the Amdahl's law indicates that the parallel speedup of a given algorithm is bounded by the sequential portions of the code.

The goal of this paper is provide a literature review of the main approaches in the context of SAT local search algorithms and highlighting a set of future trends in this area. This paper is organized as follows: Section 2 presents a general description of local search algorithms; Section 3 describes the most remarkable parallel local search approaches for SAT; Section 4 presents a global perspective of the future trends of local search for SAT; and Section 5 presents general conclusions.

Contact: Alejandro Arbelaez, JFLI / University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo-Japan, and arbelaez@is.s.u-tokyo.ac.jp

2. Local Search for SAT

Algorithm 1 shows a generic scheme for a local search algorithm to solve a given SAT instance. The algorithm starts with an initial assignment for the variables (usually random), then iteratively identifies a variable and flips the truth value of the selected variable (see [8] for a complete presentation of variable selection heuristics). The selected variable usually minimizes the number of unsatisfied clauses in the problem, however, from time to time random selections are performed in order to avoid search stagnation. The stopping criteria for the algorithm is either a timeout or failure to find a solution after a given number of iterations.

Algorithm 1: Local Search

```
Start with initial assignment A;
repeat
    if A is a solution then
        return A;
    end
    x := select-variable(A);
    A := A with x flipped;
until stopping criteria is met;
return 'No solution found';
```

Historically, GSAT [18] and WalkSAT [16] are the most important local search algorithms for the SAT problem. GSAT selects the variable which achieves the greatest reduction in the number of unsatisfied clauses in the formula. WalkSAT selects, uniformly at random, an unsatisfiable clause c and from c selects (using a given heuristic) the most suitable variable to be flipped.

3. Parallel Local Search for SAT

In this section, we present a detailed presentation of the most remarkable parallel local search algorithms for the SAT problem.

3.1 Portfolio of local search algorithms

[12] belongs to the portfolio category, in this solver several copies of the gNovelty+ algorithm are executed in parallel without communication until an assignment which satisfies all the classes is obtained or a given timeout is obtained; gNovelty+ obtained a gold medal in the 2009 SAT competition (random category, parallel track).

In [5] the authors proposed seven strategies to exploit cooperation in parallel local search. In this framework, each process exchanges the best assignment for the variables found so far with other processes in order to properly craft a new starting point. The strategies range from a voting mechanism, where each algorithm suggests a value for the variables, to probabilistic constructions. Among these strategies *Prob-NormalizedW* exhibited an outstanding performance and obtained a silver medal in the SAT'11 competition (random category, parallel track). In *Prob-NormalizedW* the new assignment is carefully formulated to ensure that better values for the variables (w.r.t the number of unsatisfiable clauses) have greater probability of being used, but at the same time poor quality values for the variables still have a small probability of being used.

However, when moving to massively parallel systems, [3] identified two main limitations in *prob-NormalizedW*: (1) an important communication overhead and (2) an excessive diversification which leads to restarting from quasi-random assignments for the variables. These two limitations were overcome by defining groups of solvers of limited size (e.g., 16 processes) and limiting cooperation to members of the same group only, whereas computations between different groups run independently.

Recently, [4] conducted an empirical evaluation of the speedup of several local search algorithms for SAT, including: Sparrow [6], AdaptiveNovelty+ [9], PAWS [22], and VW [13] on four set of well-known benchmark families, i.e., Quasigroups [1], random, verification [7], and crafted. The empirical findings in this paper suggest that the speedup of a given algorithm varies from one instance to another, however, from a global perspective instances from the same benchmark family exhibit a similar shape in the speedup curve up to few hundreds of cores; the speedup is almost linear (or ideal) for crafted and verification instances, and sub-linear for random and quasigroup instances. It is also worth noticing that the speedup for a small subset of instances is super-linear. This phenomenon is formally explained in [19] for randomized algorithms; roughly speaking, the speedup factor is related to the runtime distribution of the sequential algorithm, for instance [19] presents a hypothetical situation in which the algorithm exhibits a perfect lognormal distribution (with μ=1 and σ=2), thus the speedup is super-linear up to an important number of cores, however, due to the properties of the lognormal distribution, the speedup become sub-linear after a given number of cores.

3.2 Portfolio of hybrid algorithms

Interestingly, the parallel portfolio approach also allows a straightforward combination of systematic and stochastic search methods, which is indeed one of the ten challenges in the area of propositional reasoning and satisfiability testing [17].

Challenge 7: *Demonstrate the successful combination of stochastic search and systematic search techniques, by the creation of a new algorithm that outperforms the previous examples of both approaches.*

MiniWalk [11] proposes a framework to combine a complete solver (MiniSAT) and an incomplete one (WalkSAT) to solve the MaxSAT problem. MiniWalk executes both solvers in parallel and uses MiniSAT to guide WalkSAT to promising areas of the search space. On one hand, MiniSAT stores the current state for each variable in the problem, i.e., *unassigned*, *true*, and *false*. On the other, at each iteration of the local search algorithm, MiniWalk flips the selected variable *iff* the state reported for MiniSAT is *unassigned*, otherwise the parallel solver forces WalkSAT to use the same truth value for the variable as the one reported for MiniSAT.

In [24] the authors proposed a hybrid algorithm which works in two stages. In the first stage, the algorithm executes a DPLL-like algorithm to divide the problem space into sub-spaces. Thus, during the second stage each subspace is allocated to different processors running a given local search algorithm (WalkSAT), the global search is stopped as soon as a solution is obtained or after given timeout is reached.

3.3 Multiple flips

PGSAT [14] is a parallel version of the GSAT algorithm. In this algorithm the entire set of variables is divided into τ subsets and allocated to different processors. At each iteration of the local search process, if no global solution has been obtained, the algorithm uses the GSAT heuristic to select and flip the best variable for each subset. An interesting observation of this work is that the parallel speedup increases as τ (number of processors) increases up to a given threshold τ_{opt}, after this point the performance drops considerably; moreover this threshold varies from instance to instance. Interestingly, there is a strong empirical evidence that the optimal value (τ_{opt}) for random and structured instances is correlated to the average connectivity of the corresponding variable-clause graph.

PGWSAT [15] extends PGSAT by adding random walks to the local search procedure. In this way, with a probability *wp* the algorithm selects a random variable from an unsatisfied clause, and with probability *wp* uses the PGSAT algorithm to flip multiple variables at the same time. The new algorithm outperforms PGSAT on a set of random 3-SAT instances around the phase transition region.

[21] shows a parallel version of genSAT, a generalization of the GSAT procedure. Unlike the GSAT procedure which selects the variable with the largest improvement in the objective function, breaking ties uniformly at random; gen-

SAT eliminates the tie-breaking mechanism and flips all the variables with the overall best improvement. In the parallel version of genSAT, the authors propose a method, based on Boltzmann networks, which selects an independent subset s of variables and flips the truth value, in parallel, of all the variables in s. The experimental validation showed that this framework usually requires fewer flips that the original GSAT method.

4. Future Trends

Up to now, most parallel local search algorithms for SAT have been designed for multi-core machines or small cluster with a few tens of cores. A key question is therefore to know whether this approaches scale up to massively parallel systems, i.e. with thousands of cores. To investigate this field it would be necessary to study the *capacity solving* and *speedup* of local search solvers taking into account the following three research directions: cooperation in local search for SAT; building statistical models to obtain the maximal speedup factor of new and existing solvers; and exploiting difference sources of parallelism (e.g. GPUs) to improve performance.

Moreover, motivated by the demonstrated importance of cooperation in local search for SAT, one possible research direction is to design a modeling language to extend the cooperative framework and eliminate the implementation details induced in the use of parallel libraries such as openMP and MPI. This modeling language shall focus the attention on designing cooperative strategies to tackle a wide range of benchmark families. Moreover, these new cooperative strategies should maintain a trade-off between *intensification* and *diversification*. *Intensification* refers to the ability of exploring promising regions of the search space, while *diversification* refers to the ability of searching unexplored areas of the search space.

Local search algorithms include several randomized components. Since their runtimes depend on these random choices, it can vary from one run to the other. Indeed, this feature allows the prediction of the parallel performance of a given local search algorithms by studying the solving time of the sequential algorithm as a probability distribution. Taking this into account, in [23] the authors propose a methodology, based in order statistics, to predict the parallel performance of executing multiple copies in parallel of a given local search algorithm without cooperation. Broadly speaking, the methodology consists in approximating the empirical sequential runtime distribution by a well-known statistical distribution (e.g. exponential or lognormal) and then derivate the runtime of the parallel version of the solver. This method is related to *order statistics*, a rather new domain of statistics, which is the statistics of sorted random draws. Interestingly, extensive experimental results indicates the the predicted performance accurately matches the performance of the empirical data.

Interestingly, the prediction of the parallel performance of a given local search algorithm might have important implications in other areas, such as automatic parameter tuning to device scalable local search algorithms. Currently, most parameter tuning tools (e.g., [10, 2]) are designed to improve the expected mean (or median) runtime, however, unless the algorithms exhibit a non-shifted exponential distribution, their parallel performance is far from linear and varies from algorithm to algorithm. Indeed, experimental results suggest that the best sequential algorithm is not the best one in massively parallel systems.

The current methodology is limited to satisfiable instances and requires to solve the problem in order to estimate the theoretical distribution. Taking this into account, further developments should consider the following two directions. First, analyzing the runtime distribution of unsatisfiable instances and estimating the maximum number of satisfiable clauses and the computational time required to reach that goal. Second, new algorithms shall focus in the prediction of the parallel performance of a given algorithm for unseen instances without full sequential execution. To this end, a promising area of research would be to combine the statistical model proposed in [23] with the extensive literature for predicting the runtime of a given sequential algorithm (see [20]).

The parallel nature of the Graphic Processing Units (GPUs) offers a potential reduction in the computational time of parallel local search solvers. The thread hierarchy in a GPU consists of threads, blocks, and grids. A block is a batch of threads (all executing the same code) and blocks are grouped in a grid (blocks are independent). This hierarchy matches the architecture of the local search process by allowing the combination of multi-walk and single-walk. Taking this into account, independent local search algorithms can be allocated in different blocks, each block exploiting parallelism by evaluating neighbors in parallel. However, the amount of memory available for each block in the GPU is very limited (i.e., maximum 48 KB per block); for this reason, it is important to investigate new strategies to divide the initial formula into independent subproblems that can be allocated in the device within the memory limitation. Moreover, the GPU also allows the exploration of a larger neighborhood (e.g., multiple flip exploration) at each iteration of the local search process with almost no computational overhead.

5. Conclusions

In this paper we have presented the most important parallel local search algorithms for the satisfiability problem. These algorithms can be divided in two main categories: parallel portfolios and multiple flips algorithms. Moreover, we also point out the most important future trends in the area.

References

[1] D. Achlioptas, C. P. Gomes, H. A. Kautz, and B. Selman. Generating satisfiable problem instances. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages

256–261, Austin, Texas, USA, July 2000. AAAI Press / The MIT Press.

[2] C. Ansótegui, M. Sellmann, and K. Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In I. P. Gent, editor, *15th International Conference on Principles and Practice of Constraint Programming*, volume 5732 of *LNCS*, pages 142–157, Lisbon, Portugal, Sept 2009. Springer.

[3] A. Arbelaez and P. Codognet. Massivelly Parallel Local Search for SAT. In *ICTAI'12*, pages 57–64, Athens, Greece, November 2012. IEEE Computer Society.

[4] A. Arbelaez and P. Codognet. From Sequential to Parallel Local Search for SAT. In *13th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP'13)*, 2013. To appear.

[5] A. Arbelaez and Y. Hamadi. Improving Parallel Local Search for SAT. In C. A. C. Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION'11*, volume 6683 of *LNCS*, pages 46–60. Springer, 2011.

[6] A. Balint and A. Fröhlich. Improving Stochastic Local Search for SAT with a New Probability Distribution. In O. Strichman and S. Szeider, editors, *SAT'10*, volume 6175 of *LNCS*, pages 10–15, Edinburgh, UK, July 2010. Springer.

[7] E. M. Clarke, D. Kroening, and F. Lerda. A Tool for Checking ANSI-C Programs. In *TACAS'04*, volume 2988 of *LNCS*, pages 168–176. Springer, March 2004.

[8] H. Hoos and T. Stütze. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[9] H. H. Hoos. An Adaptive Noise Mechanism for WalkSAT. In R. Dechter and R. S. Sutton, editors, *8th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence (AAAI'02/IAAI'02)*, pages 655–660, Edmonton, Alberta, Canada, July 2002. AAAI Press / The MIT Press.

[10] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[11] L. Kroc, A. Sabharwal, C. P. Gomes, and B. Selman. Integrating Systematic and Local Search Paradigms: A New Strategy for MaxSAT. In C. Boutilier, editor, *IJCAI'09*, pages 544–551, Pasadena, California, July 2009.

[12] D. N. Pham and C. Gretton. gNovelty+ (v.2). In *Solver description, SAT competition 2009*, 2009.

[13] S. D. Prestwich. Random walk with continuously smoothed variable weights. In F. Bacchus and T. Walsh, editors, *SAT'05*, volume 3569 of *LNCS*, pages 203–215, St. Andrews, UK, June 2005. Springer.

[14] A. Roli. Criticality and parallelism in structured sat instances. In P. V. Hentenryck, editor, *CP'02*, volume 2470 of *LNCS*, pages 714–719, Ithaca, NY, USA, September 2002. Springer.

[15] A. Roli, M. J. Blesa, and C. Blum. Random walk and parallelism in local search. In *Metaheuristic International Conference (MIC'05)*, Vienna, Austria, 2005.

[16] B. Selman, H. A. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In B. Hayes-Roth and R. E. Korf, editors, *12th National Conference on Artificial Intelligence (AAAI'94)*, volume 1, pages 337–343, Seattle, WA, USA, July 1994. AAAI Press / The MIT Press.

[17] B. Selman, H. A. Kautz, and D. A. McAllester. Ten Challenges in Propositional Reasoning and Search. In *IJCAI'97*, pages 50–54. Morgan Kaufmann, August 1997.

[18] B. Selman, H. J. Levesque, and D. G. Mitchell. A New Method for Solving Hard Satisfiability Problems. In W. R. Swartout, editor, *10th National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446, San Jose, CA, July 1992. AAAI Press / The MIT Press.

[19] O. V. Shylo, T. Middelkoop, and P. M. Pardalos. Restart strategies in Optimization: Parallel and Serial Cases. *Parallel Computing*, 37(1):60–68, 2011.

[20] K. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Comput. Surv.*, 41(1), 2008.

[21] A. Strohmaier. Multi-flip networks: Parallelizing gensat. In *KI-97: Advances in Artificial Intelligence, 21st Annual German Conference on Artificial Intelligence*, volume 1303 of *LNCS*, pages 349–360, September 1997.

[22] J. Thornton, D. N. Pham, S. Bain, and V. F. Jr. Additive versus Multiplicative Clause Weighting for SAT. In D. L. McGuinness and G. Ferguson, editors, *AAAI'04/IAAI'04*, pages 191–196, San Jose, California, USA, July 2004. AAAI Press / The MIT Press.

[23] C. Truchet, F. Richoux, and P. Codognet. Prediction of Parallel Speed-ups for Las Vegas Algorithms. Technical report. Draft, http://arxiv.org/abs/1212.4287.

[24] W. Zhang, Z. Huang, and J. Zhang. Parallel Execution of Stochastic Search Procedures on Reduced SAT Instances. In M. Ishizuka and A. Sattar, editors, *Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, volume 2417 of *LNCS*, pages 108–117, Tokyo, Japan, August 2002. Springer.