Criticality and Parallelism in GSAT

Andrea Roli
DEIS - Università degli Studi di Bologna
Viale Risorgimento, 2 - Bologna (Italia)
aroli@deis.unibo.it

Abstract

In this work we show some empirical results on the parallelization of GSAT. We subdivided the set of variables in τ equal subsets and we applied GSAT in parallel on each subset. We observed the existence of an optimum degree of parallelism (τ_{opt}) for which the best performance, in terms of efficiency (time and number of iterations) and effectiveness (fraction of solved instances) is obtained. Moreover, we found that τ_{opt} is strictly correlated to the connectivity parameter (q).

1 Introduction

The Satisfiability Problem (SAT) is one of the most important NP-complete problems [2]: given a boolean formula, we ask whether an assignment to the variables exists which satisfies the formula. There are complete and incomplete SAT solvers. Among incomplete algorithms we mention GSAT [14] and WalkSAT [11] with their variants. Such algorithms belong to the class of *local search* algorithms; a local search algorithm starts with an initial assignment and moves from a state to another one by applying an operator, which modifies the current assignment. The goal is to find a solution with the optimum value of the *objective function*. Search generally stops when a (local) optimum is found, or the maximum number of iterations is reached.

An important extension of simple local search algorithms is the implicit *parallelization* of search. In [10, 1, 7, 8] optimization problems, like the Traveling Salesman Problem (TSP) and optimization on NK-landscapes, are considered; the local search algorithm performance is studied, as a function of the parallelism degree (number of subsystems or number of simultaneous moves). By getting some inspiration from these ideas, we developed a parallel version of GSAT. We observed that there exists an optimum degree of parallelism τ_{opt} which leads to the best performance in terms of efficiency (time and number of iterations) and effectiveness (fraction of solved instances). Moreover, experiments on random 3-SAT instances, suggest a correlation between the optimum degree of parallelism and the average connectivity among variables.

This paper is structured as follows: in Sec. 2 we discuss previous work about the parallelization of local search procedures; in Sec. 3 we briefly present the GSAT algorithm and its parallel version. Sec. 4 describes experimental results and in Sec. 5 we introduce the connectivity parameter and its relation with τ_{opt}. Finally, in Sec. 6 we outline some conclusions and further work.

2 Local Search and Parallelism

In this section we define the *parallelization* of local search algorithms and we present some related studies and results.

Local search is generally applied to Constraint Optimization Problems, defined as follows:

- a set of variables $X = \{x_1, \ldots, x_n\}$;
- variables domains $D_1, \ldots, D_n$;
- constraints among variables;
- an *objective function* to be optimized (minimized or maximized) $F : D_1 \times \ldots \times D_n \to \mathbb{R}^+$;

The Satisfiability Problem (SAT) can be stated as follows: given a set of clauses, each of which is the logical disjunction of $k > 2$ *literals* (a literal l_i is a variable x_i or its negation $\sim x_i$), we ask whether an assignment to the variables exists that satisfies all the clauses. It is possible to define a *representation* of SAT such that it can fit in the above-given definition of COP:

- a set of binary variables $X = \{x_1, \ldots, x_n\}$;
- variables domains $D_1 = \ldots = D_n = \{0, 1\}$;
- no constraints among variables;
- an *objective function* to be maximized: $F =$ number of satisfied clauses;

Local search algorithms (LS) need the definition of *states*, which are (feasible) configurations of the variables and of a *neighborhood function*, which assigns to each state s a set of feasible states $N(s)$. The simplest LS constructs a path along the state space by moving from a state s to another state in $N(s)$. The transition from a state to the successor is the result of the application of a *move operator* M (implicitly defined by the neighborhood function) on s. The search ends when it reaches the stop criterion, which is one of the following conditions:

- a local optimum is found. A local optimum is defined as the state such that it has the best value among its neighboring states (a local optimum can be a global optimum);
- the maximum number of iterations or the time limit are reached.

In [10, 1, 7, 8] some studies on the parallelization of local search algorithms are described. In [10] is analyzed the performance of τ simultaneous applications of the k-opt operator to the TSP, where k is the number of edges of the graph which are exchanged (τ is the degree of parallelism). The authors find a *critical value* τ_c, that produces a minimum in the cost curve; in other words, for $\tau < \tau_c$ and $\tau > \tau_c$ the best solution found has cost $c > c^*$, been c^* the cost of the solution found with τ_c. In the same work, the authors apply a parallel version of Simulated Annealing (SA) to optimization problem in NK-landscapes[1]. Suppose to have a minimization problem on boolean variables; the representation is an *energy landscape*: the goal is to find a minimum in this landscape. A move from state s_1 to state s_2 produces an energy difference $\Delta E = F(s_2) - F(s_1)$. The move operator is the simple *flip* of a variable (i.e., $x_i \leftarrow \sim x_i$). The algorithm behaves as follows: it randomly selects a variable and it flips it; it accepts this move with probability 1 if $\Delta E \leq 0$ and with probability $\exp(-\Delta E/T)$ if $\Delta E > 0$. T is the *temperature* parameter, which controls the annealing schedule: the higher T, the higher the probability to choose an *uphill* move. In the parallel version, every variable $x_i\,(i = 1, 2, \ldots, n)$ has probability p of being selected, that is, at each iteration pn variable flips are tested on average. Even in this case, the authors find the same behavior: there exists a p_{opt} for which the algorithm finds a better solution (with lowest E). Note that, in this case, the degree of parallelism is $\tau = pn$.

3 The GSAT procedure

In this section we introduce the GSAT algorithm in its original version and then we describe the way we parallelize it.

3.1 GSAT

GSAT was first introduced in [14] as a greedy local search algorithm to solve SAT problems (see Fig. 1). In its basic version, it starts from a random assignment and looks for a satisfying assignment by moving from one state to another in its neighborhood (defined as the set of states at Hamming distance equal to 1). Given a current state, the next state is chosen by *flipping* the variable that, if flipped, it increases the number of satisfied clauses the most. Ties are broken randomly.

GSAT has a hill-climbing component because it tries to increase the number of satisfied clauses by moving toward the best neighboring state. Moreover, it is able to escape from some local optima and *plateaus* by using *sideways moves*, i.e., moves from a state to another with the same difference of satisfied clauses.

3.2 Parallel GSAT

There is an easy way to parallelize GSAT: the set of variables is divided in equal subsets; if n is the number of variables, the number of subsets corresponds to the parallelism degree τ and the cardinality of each subset is n/τ. The procedure we obtain (thereinafter referred to as PGSAT, see Fig. 2) behaves as follows: at each iteration, the subset are considered in parallel and the "best" variable for each of them is flipped. Therefore, after an iteration, τ variables have been flipped. This has an effect similar to the introduction of noise, because the possible flips are evaluated supposing that variables belonging to other subsets are not modified.

[1]see [9] for references on SA and [6] for NK-landscapes.

```
procedure GSAT
Input: a set of clauses α, MAX-FLIPS and MAX-TRIES
Output: a satisfying truth assignment of α , if found
begin
    for i := 1 to MAX-TRIES
        T := a randomly generated truth assignment
        for j := 1 to MAX-FLIPS
            if T satisfies α then return T
            p := a propositional variable such that a change
                in its truth assignment gives the largest
                increase in the total number of clauses
                of α that are satisfied by T
        T := T with p reversed
        end for
    end for
    return "no satisfying assignment found"
end
```

Figure 1: GSAT algorithm

4 Experimental results

In this section we present experimental results obtained by the application of PGSAT on random 3-SAT instances; we first treat the case of Uniform Random (UF) 3-SAT instances (retrieved from the SATLIB benchmarks[2]). Then we report results on random *forced* instances[3]. The algorithm has been implemented in C and ran on a Pentium II 233MHz with 64 MB of RAM.

Each set of UF instances is composed of 100 satisfiable instances and is a problem in the threshold region (i.e., $m/n \approx 4.3$, where n and m are respectively the number of variables and clauses). Results are summarized in Table 1.

We define τ_{opt} the value of τ such that median time and median iterations have a minimum and the fraction of solved instances is the highest. We observe that (except for the 200-variables instances) there exists an optimum value for τ which allows to get the overall best performance (time, iterations and fraction of solved instances). The phenomenon is clear: in this case, as in [10], there exists an optimum degree of parallelism. There is also evidence of the fact that as the number of variables increases, also τ_{opt} increases.

Results on forced instances[4] are reported in Fig. 3 and Fig. 4. In these graphics we plotted median iterations vs. parallelism for each set of instances. Median time has the same behavior and the fraction of solved instances is always 100%. First of all, we notice that for every set of instances there is a minimum in the search cost (corresponding to τ_{opt}); moreover, we observe that τ_{opt} slightly decreases as m increases and there is not evidence for a sharp transition of τ_{opt} in the transition region. Therefore, we can conclude that exists a different parameter linked to τ_{opt}. This topic will be object of the next section.

[2] http://aida.intellektik.informatik.tu-darmstadt.de/SATLIB/.

[3] Random generated instances with at least one satisfying assignment.

[4] We generated two sets of 3-SAT random forced instances ($n = 120, 240$); each set is composed of 1000 instances for each value of m.

```
procedure PGSAT
Input: a set of clauses α, MAX-FLIPS, MAX-TRIES and τ
Output: a satisfying truth assignment of α , if found
begin
    for i := 1 to MAX-TRIES
        T := a randomly generated truth assignment
        for j := 1 to MAX-FLIPS
            if T satisfies α then return T
            parfor each subset X_k of variables (k = 1, 2, ..., τ)
                p_k :=    a propositional variable in X_k such that a change
                          in its truth assignment gives the largest
                          increase in the total number of clauses
                          of α that are satisfied by T
            end parfor each
        T := T with p_1, ..., p_τ reversed
        end for
    end for
    return "no satisfying assignment found"
end
```

Figure 2: PGSAT algorithm

5 The connectivity parameter

Typically, random CSPs are characterized by means of two parameters: the *average connectivity* q and the *constraints tightness* p (also named p_1 and p_2 elsewhere [3]). In order to study the behavior of PGSAT we used the *connectivity parameter* q, which quantifies the average number of variables connected with any variable. The parameter q ranges from 1 (variables are not linked) to n (every variable depends on all the others) and we consider the normalization of q, defined as $\overline{q} = q/n$. Note that in SAT instances with clauses composed exactly of three literals, the tightness p of each constraint is constant. In Table 2, values of $\overline{q}$ for uniform non forced and forced instances are reported: we observe that $\overline{q}$ decreases as n increases. Another observation derives from Fig. 5, where we plotted τ_{opt} against $\overline{q}$: the tendency depicted indicates that there is a strong correlation between τ_{opt} and $\overline{q}$. To reinforce this conjecture, we generated 3-SAT instances with the same $\overline{q}$ and we ran PGSAT over them. In Table 3 the results are reported: τ_{opt} is comprised between 4 and 5.

Beside the hypothesis of the correlation between τ_{opt} and $\overline{q}$, we should also formulate some considerations about the search made by GSAT. We can notice that, when the problem is underconstrained, a hill-climbing algorithm is likely to find wide flat and/or slightly gradually ascending regions in the search space and it has to cross them as quick as possible (i.e., with few moves); the parallelization of the algorithm helps doing this, because it allows "long jumps". Thus, when $\tau < \tau_{opt}$, the algorithm needs more moves to reach the optimum; dually, when $\tau > \tau_{opt}$, the algorithm cannot easily find the way to "climb" to the optimum peak, because its moves are wider than the size of its basin of attraction. And in these two cases, of course, the performance is worse than when $\tau = \tau_{opt}$.

Another important issue to address is the *correlation* of search landscapes. We evaluated the correlation structure of 3-SAT instances fitness landscape by using the *auto-correlation* function, as defined in [4, 5]. We defined the objective function F as the number of satisfied clauses and the move operator as a τ-flips move (i.e., the move flips τ variables). We observed

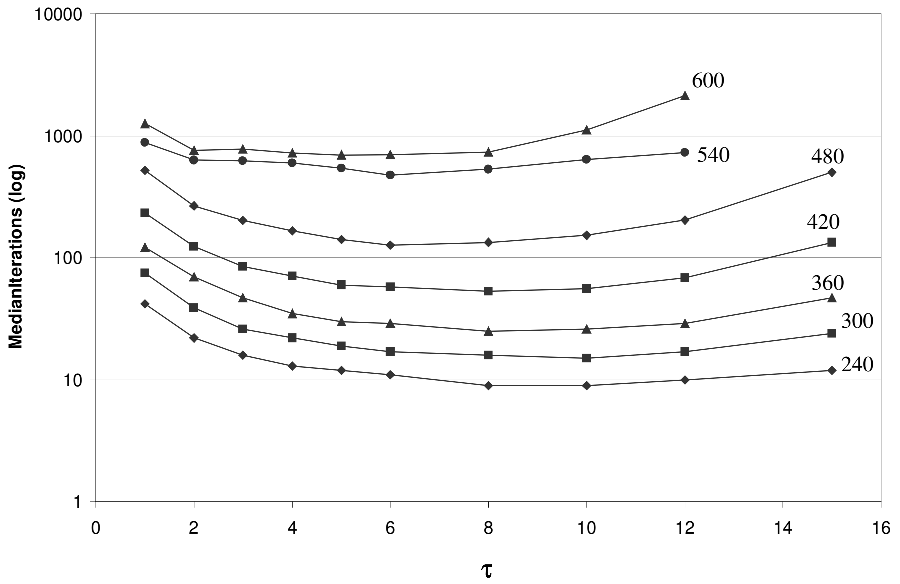

Figure 3: 120 variables. Median Iterations (in logarithmic scale) vs. τ (the number of clauses is reported at the end of the curves).

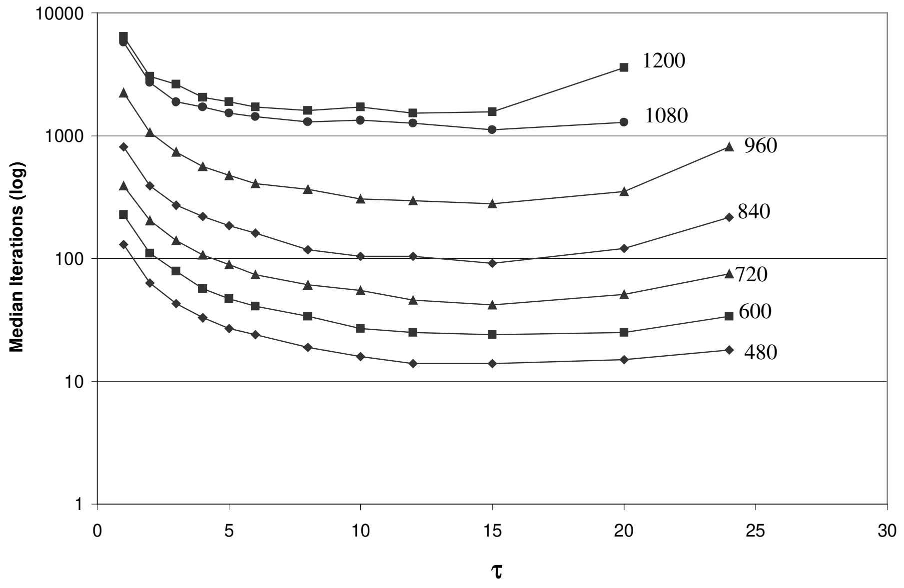

Figure 4: 240 variables. Median Iterations (in logarithmic scale) vs. τ (the number of clauses is reported at the end of the curves).

Table 1: Results on 3-SAT random instances from SATLIB. Median time (s), median iterations and fraction of solved instances out of 100 are reported. τ_{opt} is in bold. Parameters used: MAX-FLIPS $= 5n$, as in [14]; the algorithm was stopped after 10 minutes.

Instances set	τ	**Median Time (ms)**	**Median Iter.**	**Solved**
uf20, 20-91	**1**	< 0.001	38	1
	2	< 0.001	123	1
	4	0.020	1002	1
	5	0.050	2045	1
uf50, 50-218	1	0.020	536	1
	2	0.020	516	1
	5	0.070	2241	1
	10	4.847	140547	0.8
uf100, 100-430	1	0.261	5114	1
	2	0.019	3553	1
	4	0.29	5434	1
	5	0.22	4160	1
	10	2.674	48094	0.88
uf150, 150-645	1	1.963	26635	1
	2	1.022	13438	1
	3	0.871	11405	1
	5	0.971	12291	1
	6	0.661	7809	1
	10	1.752	21878	0.92
uf200, 200-860	1	26.848	276885	0.97
	2	6.179	62594	1
	4	7.001	68512	0.97
	5	4.977	49289	0.99
	10	4.586	43877	0.91

that there is not evidence for a link between auto-correlation of fitness landscape and τ_{opt}.

6 Conclusion

In this work we investigated the intrinsic parallelization of GSAT and we implemented a "parallel flip" version called PGSAT. We showed that there exists an optimum degree of parallelism (τ_{opt}) which allows to get the best performance, both in efficiency (time and number of iterations) and effectiveness (number of solved instances). We observed that τ_{opt} is correlated to the connectivity of the problem and instances with the same connectivity have nearly the same value for τ_{opt}. In previous results [10, 8], it is conjectured that problem connectivity has a key role for the best parallelization. This hypothesis finds another empirical proof in our results, extending its validity to SAT problems.

We would like, also, to briefly recall our previous work on Boolean Networks and SAT, where we found a similar behavior in SAT solvers based on Probabilistic Boolean Networks (the interested reader can refer to [13, 12]). In this work, we map a SAT problem into a Boolean Network in such a way that solutions of SAT correspond to *fixed points* of the net. SAT is solved by a dynamic system whose evolution in time corresponds to the algorithmic computation phase and stationary states represents solutions. In Probabilistic Boolean Networks, the

Table 2: Connectivity of uniform and forced formulas of 120 variables.

m	**Forced 3-SAT**	**Uniform 3-SAT**
240	0.103422	0.103506
300	0.125912	0.125690
360	0.147563	0.147486
420	0.168840	0.168890
480	0.189729	0.189842
540	0.210014	0.209496
600	0.229437	0.229211
660	0.248685	0.248553
720	0.267356	0.267342

Table 3: Results on Random 3-SAT formulas (100 instances for each set) with $\overline{q} = 0.2$. τ_{opt} is in bold.

	τ	**Median Time**	**Median Iter.**	**Solved**
n=50, m=82, q=0.2	1	< 0.001	9	1
	2	< 0.001	6	1
	5	< 0.001	5	1
	10	< 0.001	8	1
n=80, m=220, q=0.2	1	< 0.001	63	1
	2	< 0.001	39	1
	4	< 0.001	25	1
	5	< 0.001	23	1
	10	< 0.001	29	1
n=100, m=352, q=0.2	1	0.003	448	1
	2	0.002	287	1
	4	.001	138	1
	5	0.001	156	1
	10	0.002	245	1

system evolution is performed by the parallel update of τ variables, randomly chosen among the variable set. We found the existence of an optimum τ which leads to the best performance (both in efficiency and effectiveness). Moreover, we noted that τ_{opt} decreases as $\overline{q}$ increases.

Important questions are still open, both specifically bound to SAT solvers and general: is there a similar behavior for other kinds of SAT instances? How does it depend upon other parameter, such constraints tightness? Is this a general phenomenon arising in other CSPs and optimization problems and for other local search algorithms? What are the problem inner structure and parameters we should study in order to understand the behavior of (parallel) local search algorithms?

Beside these theoretical questions, the parallelization of LS has important impact in the algorithms performance for COPs. We have applied PGSAT to MAX-SAT problems and we have observed an analogous behavior [15].

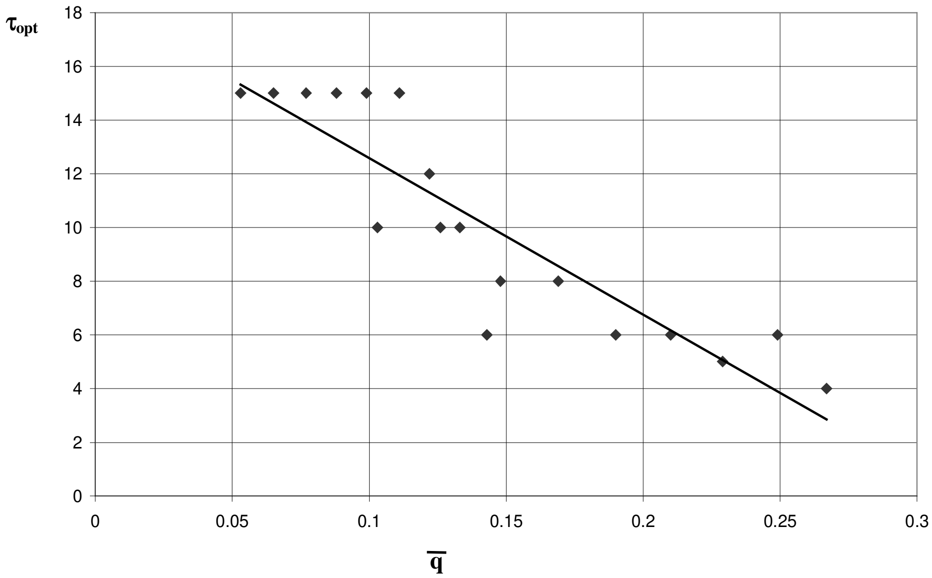

Figure 5: τ_{opt} vs. $\overline{q}$

Acknowledgments

I am grateful to Michela Milano for her help, fruitful discussions and suggestions and I would like to thank Thomas Stützle for his comments on a previous version of this work.

References

[1] Frank-Michael Dittes. Optimization on rugged landscapes: A new general purpose Monte Carlo approach. *Physical Review Letters*, 76(25):4651–4655, June 1996.

[2] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness.* W.H. Freeman, 1979.

[3] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The constrainedness of search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 246–252, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.

[4] Wim Hordijk. A measure of landscapes. *Evolutionary Computation*, 4(4), 1996.

[5] Wim Hordijk and Peter F. Stadler. Amplitude spectra of fitness landscapes. *Adv. Complex Systems*, 1(1), 1998. Also available as SFI working paper 98-02-021.

[6] Stuart A. Kauffman. *The origins of order.* Oxford University Press, New York, 1993.

[7] Stuart A. Kauffman. *At home in the universe.* Oxford Press, 1995.

[8] Stuart A. Kauffman and William Macready. Technological evolution and adaptive organizations. *Complexity*, 26(2):26–43, March 1995.

[9] S. Kirkpartick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, 13 May 1983*, 220(4598):671–680, 1983.

[10] William G. Macready and Stuart A. Kauffman Athanassios G. Siapas. Criticality and parallelism in combinatorial optimization. *Science*, 271:56–59, January 1996.

[11] David McAllester, Bart Selman, and Henry Kautz. Evidence for invariants in local search. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 321–326, Menlo Park, July 27–31 1997. AAAI Press.

[12] Michela Milano and Andrea Roli. Solving the satisfiability problem through boolean networks. In Evelina Lamma and Paola Mello, editors, *Lecture Notes in Artificial Intelligence - AI*IA99: Advances in Artificial Intelligence*, volume 1792, pages 72–83. Springer, 2000.

[13] Andrea Roli. Boolean networks: Applications to the satisfiability problem. Master's thesis, Faculty of Engineering - Dept. of Electronics, Computer Science and Systems - University of Bologna - Italy, July 1998. In Italian, available at http://www.lia.deis.unibo.it/Staff/AndreaRoli/pubs/tesi.htm.

[14] Bart Selman, Hector J. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California, 1992. American Association for Artificial Intelligence, AAAI Press.

[15] Andrea Roli and Christian Blum. Critical Parallelization of Local Search for MAX-SAT. Submitted to AI*IA2001 - 7th Conference of the Italian Association for Artificial Intelligence. Preprint available at: http://iridia0.ulb.ac.be/~aroli/