Problem structure and multi-move local search: Results achieved and lines for further research

Andrea Roli

Technical Report No.

TR/IRIDIA/2004/24.01

December 2004

Problem structure and multi-move local search: Results achieved and lines for further research*

Andrea Roli
Dipartimento di Scienze
Università degli Studi "G. D'Annunzio"
Pescara – Italia
a.roli@unich.it

December 8, 2004

Abstract

The impact of problem structure on search is a relevant issue in artificial intelligence research and related areas. Among the possible approaches to analyze problem structure, the one referring to constraint graph enables to relate graph parameters and characteristics with search algorithm behavior. In this work, we present and discuss an empirical study which connects this topic with a phenomenon called *criticality and parallelism*. This phenomenon has been observed in the context of parallel-move local search algorithms applied to combinatorial optimization problems. The main result is that increasing parallelism leads to better solutions, but up to a degree at which the solution quality degrades. Moreover, the optimal parallelism is negatively correlated with the system connectivity. We show empirical evidence for the presence of a similar phenomenon when local search for satisfiability problems is concerned. We study the behavior of a parallel local search as a function of the number of simultaneous local moves (parallelism). For different random instance typologies, we show that an optimal value of parallelism exists such that the algorithm achieves, on average, the best solution quality. Experimental results also show that the higher the system connectivity, the lower the optimal parallelism. We also extend the study toward structured instances and we conjecture that, in this case, the optimal parallelism depends on the peaks of node degree distribution of the graph associated to the instances.

1 Introduction

The impact of problem structure on search is a relevant issue in AI research and related areas. This topic has been recently received more attention, due to the following reasons: *(i)* real-world problems are often more difficult to solve than random generated problems of the same size and *(ii)* results obtained by applying statistical mechanics techniques (such as phase transition analysis [18]) have shown a strong correlation between search effectiveness and some critical parameters of the instances at hand.

In this work we investigate the relation between some SAT/MAXSAT instance features and the behavior of local search. We will define structural features on the basis of a constraint graph associated to the instances and in particular we will deeply investigate the impact of the node degree distribution on the behavior of multi-move local search —also known as parallel local search, since more than one local move is applied synchronously at each iteration. This work is inspired by a phenomenon called *criticality and parallelism*, first discovered in combinatorial optimization problems such as the TSP and NK-models [26]. The main result is that increasing

*This work have been partially written during a visiting period at IRIDIA.

parallelism leads to better solutions, but up to a degree at which the solution quality degrades. Moreover, the optimal parallelism is negatively correlated with the system connectivity.

In this work, we present an empirical analysis to investigate whether and under which hypotheses a similar phenomenon can be observed also in satisfiability problems (both the satisfiability problem —SAT— and the maximal satisfiability problem —MAXSAT). We first address the issue of relating SAT instances with graphs with the aim of defining general structural parameters of the instances. Then we present results of the 'parallelization' of local search on random 3-SAT instances. With 'parallel local search' we mean local search in which more than one variable flip is performed. (Conceptually, moves are applied synchronously.) We show that there exists an optimal value of parallelism which drives local search to achieve an optimal performance in terms of average solution quality. We also observe that the optimal parallelism is negatively correlated with a structural parameter of the graph, namely the average node degree. These results hold both for SAT and MAXSAT instances. Furthermore, we extend our analysis in two opposite directions: *(i)* on artificial generated SAT instances characterized by constant node degree and *(ii)* on structured SAT instances, characterized by an irregular node degree distribution.

This work aims at providing a complete picture of the studies concerning criticality and parallelism w.r.t. local search for SAT and MAXSAT [36, 41, 40, 37, 38, 39]. Furthermore, most of the previous experiments have been performed again in order to have homogeneous data. We will show and discuss results and we will present research lines for further work. Indeed, this work should be considered as a first investigation into the subject.

The paper is structured as follows. In Sec.2, we briefly summarize the literature concerning criticality and parallelism in combinatorial optimization. In Sec.3 we give the definition of the graph structure associated to SAT and MAXSAT instances. The definition of this graph enables us to characterize the relations among problem variables and to provide a definition of *system connectivity*, as defined in [26, 24, 23]. Then, Sec.4 describes the parallel local search we used and reports experimental results on random SAT and MAXSAT instances. For both the problems, we will show that an optimal value of parallelism exists which enables the algorithm to achieve an optimal performance in solution quality (on average). Moreover, the empirical analysis shows that this optimal value is negatively correlated with the connectivity of the instance. Sec.6 reports results concerning SAT instances characterized by a constant connectivity graph (i.e., all nodes have the same number of edges). The results show that the optimal parallelism is independent of the ratio between clauses and variables and it is dependent on the connectivity of the graph. Sec.7 discusses experiments performed on structured SAT instances. These instances have a very irregular node degree distribution, which is neither constant, nor Gaussian (as in the case of random SAT instances). The results obtained suggest a generalization of the previous results, since an optimal parallelism is found for structured instances, as in the case of constant degree and random ones and we conjecture that the value of the optimal parallelism is affected by the peaks of the node degree distribution. We then discuss the results in Sec.8. Finally, we succinctly report related work in Sec.9 and we conclude with future work in Sec.10.

2 Criticality and Parallelism in Combinatorial Optimization

The phenomenon called *criticality and parallelism* has been observed in the context of local search algorithms applied to combinatorial optimization problems (COPs), where local search is modified by applying more than one local move in parallel [26, 24, 23]. It has been shown that the effectiveness of these algorithms depends on the parallelism degree τ (number of simultaneous moves): if τ increases, the solution quality[1] also increases up to a point (corresponding to τ_{opt}) at which it starts decreasing. It has also been shown that τ_{opt} is negatively correlated with the *connectivity* among variables of the problem: the higher the connectivity, the lower τ_{opt}.

[1]In the following we will use the expression *solution quality* referring to the value of the objective function; in case of a minimization problem, the lower the objective function value, the higher the solution quality.

In [26, 9, 23, 24] some studies on the parallelization of local search algorithms are described. In [26], the authors apply a parallel version of simulated annealing to optimization on NK-models [22, 23]. In brief, this approach can be described as follows. Suppose to have a minimization problem on n boolean variables $x_1, \ldots, x_n$. The search space can be represented as an *energy landscape*: the goal is to find a minimum in this landscape. Every variable x_i is associated to an energy value e_i, which is a function of x_i and other K variables. The objective function of the system (total energy) is $E = \frac{1}{n}\sum_{i=1}^{n} e_i$. A move from state[2] s_1 to state s_2 results in an energy difference $\Delta E = E(s_2) - E(s_1)$. The application of the move operator is, in this case, simply a *flip* of a variable (i.e., $x_i \leftarrow \neg x_i$). The basic algorithm behaves as follows: it randomly selects a variable and 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 a *temperature* parameter, which controls the annealing schedule: the higher T, the higher the probability to choose a non-improving 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 parallel variable flips are tested on average. Hence, the degree of parallelism of search is $\tau = pn$. The authors discover that there is a p_{opt} for which the algorithm finds the solution with the lowest E: higher or lower values of p on average produce higher total energy values.

Since the effect of a variable flip on the objective function value is evaluated as if it was the only one to change, parallel (i.e., simultaneous) flips introduce a kind of noise in the energy evaluation. As observed in [27], the introduction of noise increases the effectiveness of local search, since it helps to escape from local optima. It is worth to note that [27] shows that the quality of solutions found increases as noise increases, up to a critical value above which the performance decreases again. However, differences and similarities between parallel local search and local search with noise have still to be completely discovered and explained.

Analogous results are reached in [23, 24], where yet a different approach is chosen. The COP is, in this case, the optimization of a NK-model with variables arranged in a bi-dimensional lattice; every variable corresponds to a cell in the lattice and K indicates the number of neighboring cells linked to it. The lattice is divided in P non-overlapping patches and a simple local search is applied in parallel to each patch. A variable is flipped if it decreases the energy of the patch it belongs to. The authors find an optimal number of patches which allows the search to reach the lowest total energy value.

The underlying principle of the last approach is that, in order to optimize systems composed of conflicting elements, it is generally useful dividing the system in subsystems and optimize each of them independently. One of the effects of simultaneous changes is to help the search to avoid local optima, as they introduce a kind of *noise* due to the fact that each subset performs a local move supposing the other subsets do not change. Moreover, the authors claim that the optimal subdivision drives the system in a state such that subsystems coordinate themselves for a global optimization goal.

In [26] and [24], the authors also find that the objective function rapidly decreases in correspondence of the optimal parallelism. This abrupt behavior change corresponds to a phase transition that can be observed in an order parameter that is a function of the entropy of the system. Hence, the introduction of the term *criticality*, referring to the critical value of the order parameter. For further details, we forward the interested reader to the original papers.

These works on parallelization of search propose very useful ideas for the improvement of local search for COPs and suggest new directions to understand local search behavior.

In summary, the literature concerning criticality and parallelism encompasses the following ways to parallelize local search:

- At each iteration, apply a local move on each variable (or a solution component) with probability p. This results in an average parallelism of pn, where n is the number of variables.
- Divide the problem in τ subsystems (which are, in general, not independent) and apply local search to optimize each of them independently.

[2]a state is a complete variable assignment

- At each iteration, temporarily exclude some constraints between variables and apply a local move on the relaxed problem.

The aim of this work is to investigate the behavior of parallel local search applied to satisfiability problems as a function of the number of parallel moves. In order to move the focus from NK-models to satisfiability problems, we need first to define the connectivity of a satisfiability problem instance, so as to have an equivalent parameter to K in NK-models.

3 Structure of Satisfiability Problems

SAT belongs to the class of NP-complete problems [10] and can be stated as follows: given a set of clauses, each of which is the logical disjunction of $k > 2$ *literals* (a literal is a variable or its negation), we ask whether an assignment to the variables exists that satisfies all the clauses. MAXSAT is a NP-hard problem and can be stated as follows: given a set of clauses, the objective is to find an assignment to the variables such that it maximizes the number of satisfied clauses[3].

The definition of structure emerging from the literature on Constraint Satisfaction Problems (CSPs) and COPs is usually based on the informal notion of a property enjoyed by non-random problems. Thus, *structured* is used to indicate that the instance is derived from a real-world problem or it is an instance generated with some similarity with a real-world problem. Commonly, we say structured for a problem which shows, under some abstraction, regularities such as well defined subproblems, patterns or correlations among problem variables.

There are also some quantitative measures of structure, such as entropy (see for example [17]), small-world proximity [47] and compression ratio [42].

The impact of problem structure on search performance has been studied from different perspectives. Studies on the impact of problem structure on heuristic search can be found, for example, in [4, 52, 49, 25]. Important results and observations on structure and problem hardness are reported in [15, 13, 14]. The effects of problem encoding are discussed in [20, 5]. Finally, the search algorithms behavior w.r.t. graph properties has been discussed in [48, 47, 33, 25, 16].

Some problems suggest a natural structural description, since they have a representation suitable for structure analysis. A classical example are problems defined on graphs, such as the Graph Coloring Problem and the k-Cardinality Tree Problem. In general, we have to choose a level of abstraction and a suitable data structure to associate to the problem. Then we characterize the structure on the basis of relevant properties of the model we have obtained.

In this section we give the definition of a graph associated to SAT/MAXSAT instances. This graph has strong similarities with the *constraint graph* defined for constraint satisfaction problems and it has been elsewhere introduced as the *interaction graph* [35]. Given a constrained problem — with binary constraints— the *constraint graph* is the graph in which nodes correspond to problem variables and edges link variable involved in a constraint.

The graph we associate to a SAT instance is an undirected graph $G = (V, A)$, where each node $v_i \in V$ corresponds to a variable and edge $(v_i, v_j) \in A$ $(i \neq j)$ if and only if variables v_i and v_j appear in a same clause. For instance, in Fig.1 the graph corresponding to the formula $F_1 = (a \vee \neg b) \wedge (b \vee d) \wedge (c \vee \neg d \vee \neg e)$ is depicted.

Observe that the same graph corresponds to more than one formula, since nodes are connected by only one arc even if the corresponding variables belong to more than one clause. For example, the graph of Fig.1 corresponds also to the following instances: $F_2 = (\neg a \vee \neg b) \wedge (\neg b \vee \neg d) \wedge (\neg c \vee \neg d \vee \neg e)$, $F_3 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (b \vee d) \wedge (c \vee \neg d \vee \neg e) \wedge (c \vee d \vee e)$. F_2 has the same number of clauses as F_1, but some clauses in F_2 are different. Also F_3 has the same associated graph, but it has a different number of clauses than the previous formulas. For brevity, in the following we will refer to the simple graph defined at the beginning of this section as *SATgraph*.

Having a set of clauses associated to the same graph, makes this representation quite rough. Nevertheless, in the following, it will be shown that some properties of the *SATgraph* strongly

[3]If weights are associated to clauses, the objective is to maximize the weighted sum of satisfied clauses. In this case, the problem is called Weighted MAXSAT.

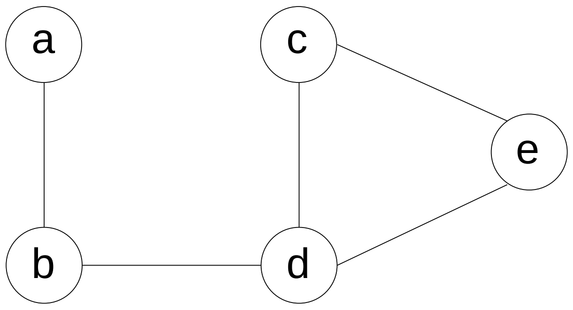


Figure 1: Constraint graph associated to a SAT instance.

affect the behavior of local search applied to SAT and MAXSAT instances.

The relevant parameter of the *SATgraph* we have to consider for the purposes of this paper, is the *node degree*.

In an instance with n variables, each node v_i, $i = 1, \ldots, n$, has a degree $q_i \in \{0, 1, \ldots, n-1\}$. For k-SAT problems, defined as conjunction of clauses with exactly k literal each, it holds $k - 1 \leq q_i \leq n - 1$. We define the *average connectivity* of the instance as the average node degree of the corresponding graph, i.e., $q = \frac{1}{n}\sum_{i=1}^{n} q_i$. Moreover, to have a uniform measure of connectivity among instances with different number of variables, we also introduce the normalization of q: $\overline{q} = \frac{q}{n-1}$. In the following, we will use interchangeably the expressions connectivity and node degree of an instance $\mathcal{I}$, where the latter is defined on the graph associated with $\mathcal{I}$. In order to compare the node degree distribution between instances, we will also consider the frequency of node degree $Freq(j)$ = 'frequency of a node connected to exactly j nodes' and the cumulative frequency $CumFreq(j)$ = 'frequency of a node connected to not more than j nodes'.

The connectivity gives a rough evaluation of how a modification occurring on a node affects the other nodes and spreads across the graph. The higher the connectivity, the stronger the "information spreading"[4]. The impact of connectivity will be discussed in detail in the following.

In this work, we use the *SATgraph* as a model of the structure of SAT/MAXSAT instances. However, graphs of different kind can be introduced to study the structure of SAT problems. For instance, a graph can be defined with weights on edges to account for the number of clauses involving the connected variables. Moreover, it is possible to construct a graph where nodes corresponds to literals instead of variables and adding one node for each clause.

4 Parallel Local Search for SAT and MAXSAT

In this section we present one possible way of parallelizing local search to tackle SAT and MAXSAT. Then, we show and discuss experimental results concerning random SAT and MAXSAT instances.

4.1 Parallel GSAT

Since the previous results on criticality and parallelism in combinatorial optimization were obtained by applying a (quite simple) local search characterized by a strong hill climbing tendency, for our experiments on SAT we chose GSAT [44].

GSAT was first introduced in [44] as a greedy local search algorithm to solve SAT problems (see Alg. 1, as it was described in the original paper). In its basic version, it starts from a random assignment and looks for a satisfying assignment by moving from one state to another one 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, leads to the greatest gradient in the number of satisfied clauses. Therefore, a variable flip is performed even if the total

[4]These considerations have been initially motivated by Kauffman's work [22, 23].

number of unsatisfied clauses increases. Indeed, if the incumbent solution is a local optimum, all the neighboring solutions correspond a lower number of satisfied clauses and a flip is performed toward the solution that keeps the number of unsatisfied clauses the lowest.

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. Despite the effectiveness of sideways moves, GSAT can be stuck in small areas of the search space without escaping (i.e., it *stagnates*[5]) and other more recent local search algorithms [19] perform better on SAT instances.

There is a very easy way to parallelize GSAT: the set of variables is divided in equal subsets. The subsets are randomly constructed at the beginning of each iteration. 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[6]. The procedure we obtain (thereinafter referred to as PGSAT, see Alg. 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.

We would like to stress that the algorithms at hand are implemented sequentially. With "parallel moves" we mean "synchronous moves". Anyhow, these results could be beneficial also for implementations on parallel architectures.

4.2 Random 3-SAT Instances

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 [21] benchmarks[7]). Then we report results on random *forced* instances[8].

The instances taken from SATLIB are located in the threshold region [1, 30, 18] (i.e., $m/n \approx 4.3$, where n and m are respectively the number of variables and clauses) and are satisfiable.

We analyzed the behavior of parallel PGSAT on random instances by using as a termination condition a maximum number of moves without improvement[9]. Therefore, the algorithm stops as soon as it stagnates. This termination condition is of the same nature as the one used in the original experiments on criticality and parallelism, since it stops when the system reaches a steady state. We also performed experiments using a time limit as a termination condition and we found that results are qualitatively the same.

We run PGSAT on ten instances for every value of n (n = 20, 50, 100, 150, 200, 250). Results, averaged over 100 trials per instance, show the number of unsatisfied clauses returned at the end of the run (Fig.2). We can observe that for every value of n the average number of unsatisfied clauses returned by the algorithm has a minimum corresponding to a $\tau_{opt}(n)$. The success rate (not reported in the figures) follows the same qualitative behavior, even though the maximal success rate is achieved for a τ slightly less than τ_{opt}. In Table 1 we report average (with standard deviation) and median of τ_{opt} values related to the considered instances.

From the table we observe that τ_{opt} tends to increase as n increases. Nevertheless, this fact is not just a direct consequence of size scaling, as we can see by studying the behavior of PGSAT on instances of the same size, but varying ratio m/n. We generated forced random 3-SAT formulas with various ratio between clauses and variables. Results are reported in Fig.3, Fig.4 and Fig.5. In these graphics we plotted the average error (in logarithmic scale) vs. parallelism for each set of instances. First of all, we notice that for every set of instances there is a minimum in the average error. Note that, since the instances with $m/n = 2$ are extremely easy, they are solved by

[5]With the terminology of dynamical systems, we could say that this condition corresponds to the reach of an attractor [3, 2, 8].

[6]More precisely, all subsets have equal cardinality, except for one which contains $\tau + n \ mod \ \tau$ variables.

[7]www.satlib.org

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

[9]This cutoff value has been set to n.

Algorithm 1 GSAT

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

Algorithm 2 Parallel GSAT

```
Input: a set of clauses $\alpha$, $\tau$, $MAXMOVES$
Output: a truth assignment of $\alpha$
$T \leftarrow$ initial truth assignment {Randomly generated}
$MOVES \leftarrow 0$
$besterror \leftarrow Eval(\alpha, T)$ {$Eval(\alpha, T)$ returns the number of unsatisfied clauses}
$bestsolution \leftarrow T$
while $MOVES < MAXMOVES$ do
  if $T$ satisfies $\alpha$ then
    return $T$
  end if
  Divide the set of variables in $\tau$ subsets (randomly)
  for all subset $X_k$ of variables ($k = 1, 2, \ldots, \tau$) do
    $p_k \leftarrow$ a propositional variable in $X_k$ such that a change
    in its truth assignment gives the largest decrease in the number of clauses of $\alpha$ that are not
    satisfied by $T$
  end for
  $T \leftarrow$ $T$ with the truth value of $p_1, p_2, \ldots, p_\tau$ reversed
  $MOVES \leftarrow MOVES + 1$
  $error \leftarrow Eval(\alpha, T)$
  if $error < besterror$ then
    $MOVES \leftarrow 0$
    $bestsolution \leftarrow T$
    $besterror \leftarrow error$
  end if
end while
return $bestsolution$
```

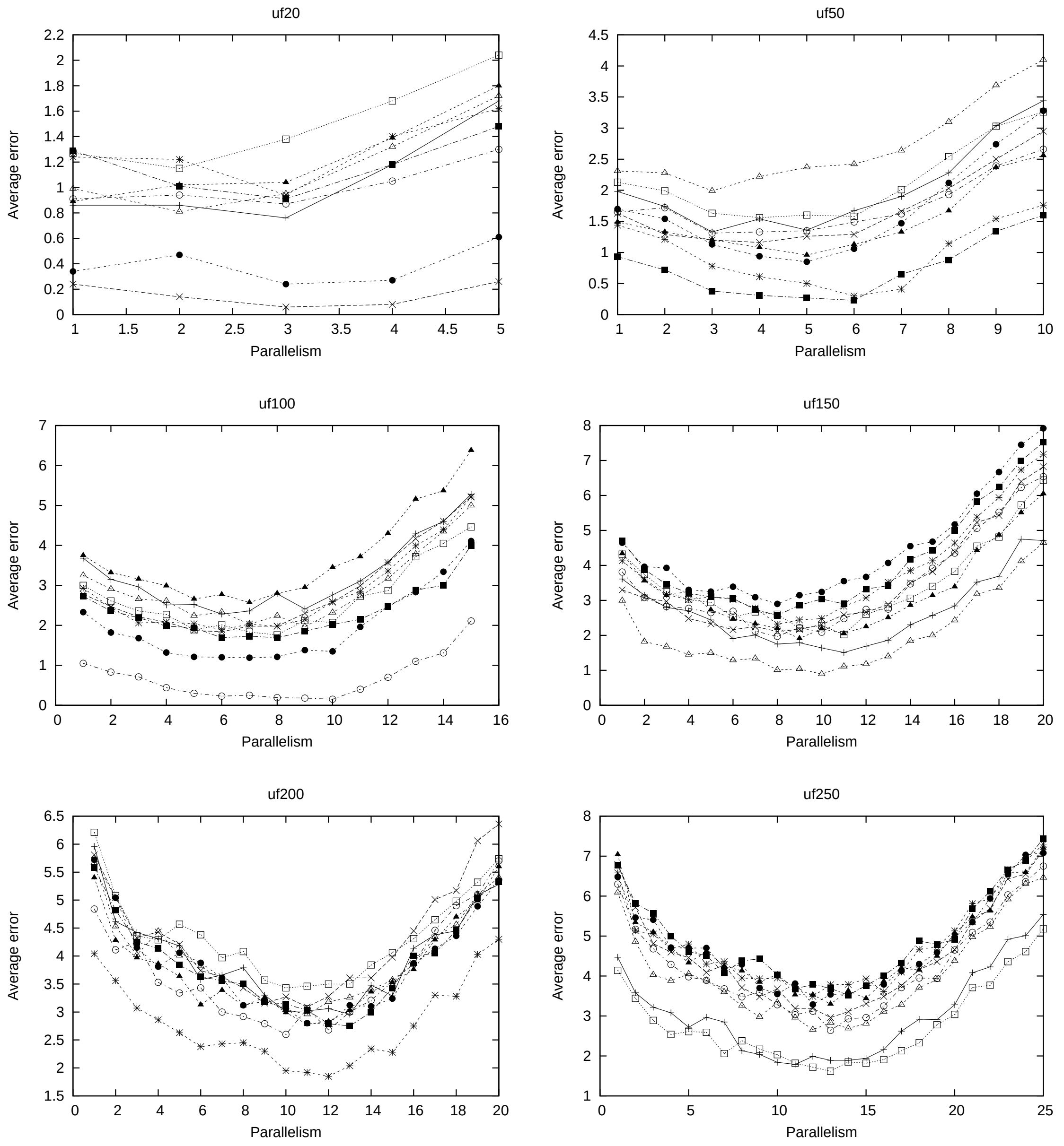


Figure 2: Average (number of unsatisfied clauses) vs. parallelism (τ) for random 3-SAT instances with $20, 50, 100, 150, 200$ and 250 variables ($m/n \approx 4.3$). Results are averaged over 100 trials.

Table 1: Average, standard deviation and median values of τ_{opt}.

n	average	std. dev.	median
20	2.4	0.84	3
50	4.5	1.27	4.5
100	7.6	1.43	7.5
150	9.4	1.35	9.5
200	11.4	1.07	11
250	12.9	1.37	13

Table 2: τ_{opt} for forced instances. In case of different values of τ enabling PGSAT to achieve a success rate of 100%, we reported the maximum value.

n	m	τ_{opt}
50	100	11
	150	6
	200	7
	250	5
100	200	16
	300	9
	400	9
	500	7
200	400	28
	600	18
	800	15
	1000	13

Table 3: Average connectivity and its normalized value for 3-SAT random generated instances. The values refer to one instance per class, but these values are practically the same for all the instances belonging to the same class.

number of variables	q	$\overline{q}$
20	14.70	0.7736
50	20.56	0.4195
100	22.70	0.2292
150	24.04	0.1613
200	24.00	0.1206
250	24.24	0.0973

PGSAT with all the considered values of τ; however, for extreme values of τ, the performance of the algorithm decreases.

In Table 2 we reported the values of τ_{opt} for the considered forced instances. We clearly see that τ_{opt} considerably varies, despite the fact that n is constant. This observation will be reinforced by the results of the following sections.

Another important point to observe is that τ_{opt} decreases as m increases and there is not evidence for a sharp transition of τ_{opt} in the transition region. The intuition behind this is that the ratio m/n relates only to satisfiability and to the *hardness* of the instance [7, 30, 12, 45], therefore it does not directly influence the qualitative behavior of local search with parallel flips.

To explain the behavior of τ_{opt} across the instances we considered, we conjecture that τ_{opt} is negatively correlated with q and it is not just an effect of size scaling. Indeed, at fixed ratio between clauses and variables, q decreases as n increases, as shown in Table 3.

In Fig.6, we plotted τ_{opt} against $\overline{q}$: the tendency depicted indicates that there is an apparent negative (nonlinear) correlation between τ_{opt} and $\overline{q}$. Therefore, we can conjecture that τ_{opt} depends on the average normalized connectivity (at least, for random 3-SAT and under the approximation introduced by the *SATgraph*).

4.2.1 Summary of first experiments set

The experiments performed on random 3-SAT instances suggest two observations:

1. There is a value of $\tau_{opt} \in \{1, \ldots, n\}$ that leads PGSAT to achieve the best performance. The average error returned by PGSAT(τ_{opt}) is not higher than that of PGSAT(τ), for $\tau \in$

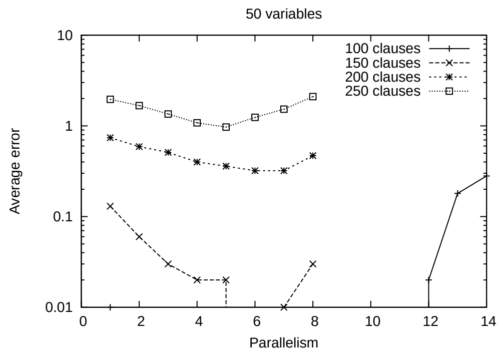


Figure 3: Forced instances with 50 variables. Average error (in logarithmic scale) vs. τ.

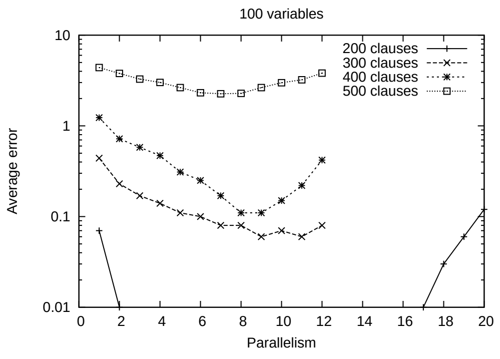


Figure 4: Forced instances with 100 variables. Average error (in logarithmic scale) vs. τ.

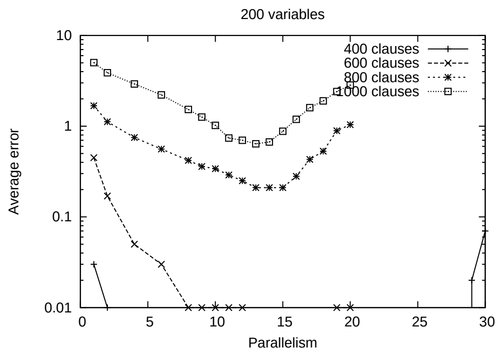


Figure 5: Forced instances with 200 variables. Average error (in logarithmic scale) vs. τ.

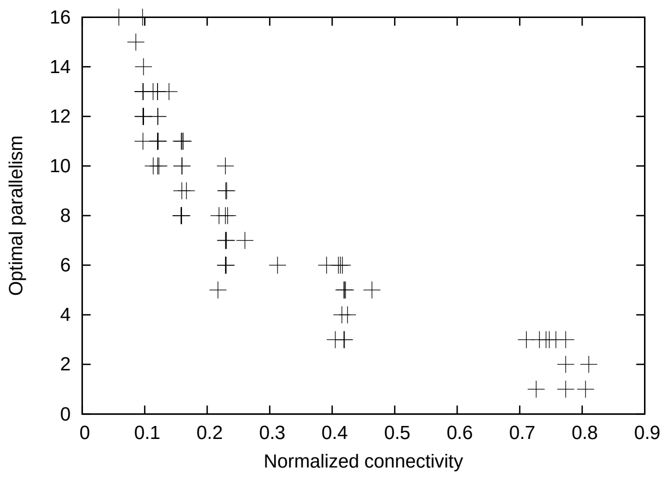


Figure 6: τ_{opt} vs. $\overline{q}$

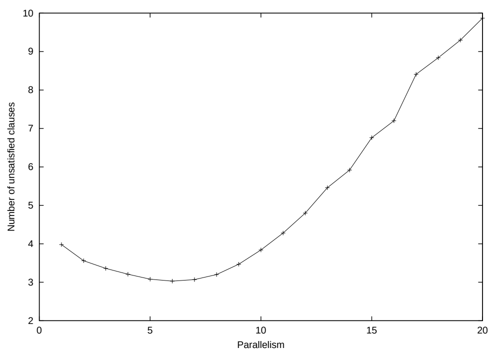


Figure 7: Average error vs. τ. Instance with 100 variables and 430 clauses. Results are averaged over 10000 runs.

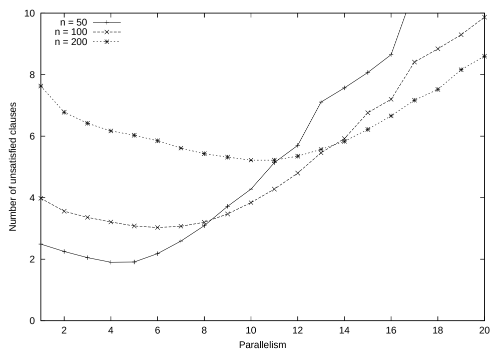


Figure 8: Average error vs. τ. Instances with 50,100,200 variables and 218,430,860 clauses respectively. Results are averaged over 10000 runs.

$\{1, \ldots, n\}$.

2. τ_{opt} seems negatively correlated with the average connectivity of the *SATgraph*: the higher $\overline{q}$, the smaller τ_{opt}.

The first observation is not surprising. Indeed, since GSAT could stagnate in a small region of the search space, applying more than one flip in parallel can help the search to escape from that region. At the other extreme, if $\tau = n$ the search does not proceed, but it just oscillates. Therefore, it is likely that a value τ_{opt} such that $1 < \tau_{opt} < n$ enables the algorithm to perform better. Nevertheless, what makes the results very interesting is the second observation: the value of τ_{opt} seems to be linked with a structural property of the instance (the average connectivity $\overline{q}$), independently of other instance parameters, such as the ratio m/n. Moreover, the supposed correlation between τ_{opt} and $\overline{q}$ perfectly fits in the previous results on criticality and parallelism in combinatorial optimization. In this case, the intuition behind the phenomenon is that when the relations between variables are loose (i.e., $\overline{q}$ is low), a variable flip affects a relatively small number of other variables, thus the system can be subdivided into several nearly independent subsets. On the contrary, tight relations produce a large network of dependencies among the variables and thus they are more sensitive to single flips.

We should also observe that there is no sharp transition between low to high average error (or viceversa) like the one observed in the experiments performed on NK-models. This fact can be explained observing that the greedy local search characterizing GSAT is more effective than the simple hill climbing and less sensitive to local minima.

4.3 Results on MAXSAT

We applied PGSAT also to MAXSAT instances, to test whether the connection between connectivity and τ_{opt} is influenced by the property of formulas to be satisfiable. We discovered that the phenomenon appears with the same characteristics also in unsatisfiable formulas and considering an optimization problem. We emphasize that we are not concerned with *instance hardness*, but rather with the comparison of algorithm effectiveness across varying τ values.

We tested PGSAT on random generated unsatisfiable instances with three literals per clause (3-SAT), retrieved from SATLIB. The graph in Fig.7 reports the average error (number of unsatisfied clauses) as a function of τ for an instance with 100 variables and 430 clauses. This graph shows the typical behavior of the algorithm. Observe that the original algorithm ($\tau = 1$) reaches an average error of 4 and, as τ increases, the error decreases until a minimum at $\tau = \tau_{opt} = 6$; above that value the average error starts to increase.

Table 4: Median and mean of optimal values of τ for each set of instances.

number of variables	Median τ_{opt}	Mean τ_{opt}
50	4	4.18
75	6	5.9
100	6	6.3
125	7	7.5
150	8	9.4
175	10	10.33
200	11	10.7
225	12	12.36
250	12	12.6

Table 5: Average connectivity and its normalized value for typical 3-SAT random generated instances.

number of variables	q	$\overline{q}$
50	19.88	0.4057
75	21.76	0.2941
100	22.88	0.2311
125	23.33	0.1881
150	23.99	0.1610
175	24.00	0.1379
200	24.11	0.1212
225	24.21	0.1881
250	24.34	0.0976

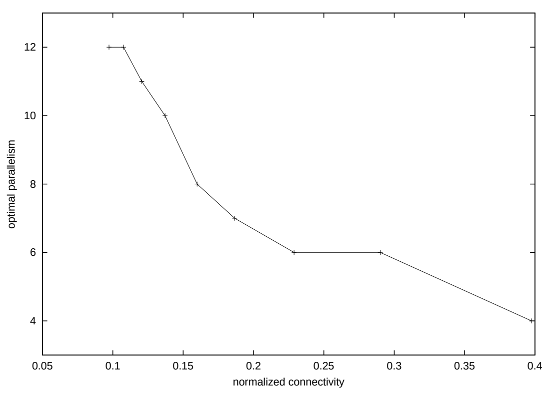


Figure 9: Optimal parallelism τ_{opt} vs. normalized average connectivity $\overline{q}$.

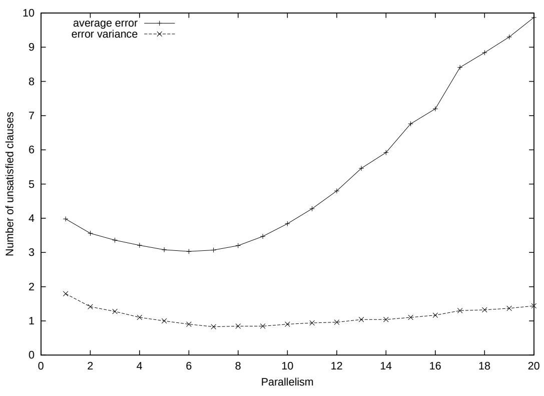


Figure 10: Average error and variance vs. τ. Instance with 100 variables. Results are averaged over 10000 runs.

For all sets of instances, a similar behavior has been noticed. Moreover, as can be observed in Fig.8, the higher the number of variables, the higher τ_{opt}. This result is summarized in Table 4, where for each set median and mean of τ_{opt} are reported[10].

Table 5 shows the typical values of $\overline{q}$ for the instances considered in this analysis. Fig.9 shows the graphic combination of Table 4 and Table 5. We notice that the curve representing τ_{opt} vs. $\overline{q}$ is monotonically non increasing, in accordance with the results of the previous section.

A further analysis of the statistics presented earlier in this section, permits to discover an interesting phenomenon. For example, the graph of Fig.10 shows the average error and its variance as a function of τ for an instance with 100 variables. We observe that the minimum error is obtained for a value of τ slightly smaller than the value for which the variance has a minimum. We conjecture that near the critical value τ_{opt} the algorithm achieves the highest effectiveness and thus it converges toward a smaller region around the best value it can find.

Concluding, we have shown that an optimal value of parallelism exists also for MAXSAT and that it is negatively correlated with the average connectivity of the instance. These results are in accordance with the results discussed in the previous section and they can be seen also as a generalization of them, since they are related to an optimization version of SAT.

[10]The average error has been evaluated over 100 runs and then median and mean of τ_{opt} over the 10 instances of each set have been considered.

5 τ_{opt} vs. $\overline{q}$: Experimental analysis

So far, we have presented and discussed experiments on PGSAT tackling random SAT/MAXSAT instances. Observing the results, we can conjecture that τ_{opt} is strongly (negatively) correlated with the average node degree of the *SATgraph*. In fact, for random 3-SAT instances —both satisfiable and unsatisfiable— we noted that the higher $\overline{q}$, the lower τ_{opt}. From the graphs plotted in Fig.6 and Fig.9, we also note that the relation between τ_{opt} and $\overline{q}$ is nonlinear.

In this section we present and discuss results concerning random 3-SAT instances. The parameter of interest for the generation of the instances are the number of variables n and the average normalized connectivity $\overline{q}$. We generated instances for almost all the possible combinations of the parameters in the following ranges: $n \in \{50, 100, 150, 200, 250\}, \overline{q} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. These instances are generated by iteratively considering a random generated clause (with variables negated with probability 0.5) and adding it to the formula if its normalized connectivity $\overline{q}$ is less than the desired value $\overline{q}_0$. This procedure terminates when $\overline{q} \in [\overline{q}_0 - \epsilon, \overline{q}_0 + \epsilon]$, with $\epsilon = 0.001$.

For some values of n, it was not possible to generate the instances corresponding to some extreme values of $\overline{q}$. In fact, the average connectivity of 3-SAT instances tends to saturate around the value 24, when $n \geq 150$, so it is very unlikely that a random generated instance can reach a very high value of normalized average connectivity.

In Table 6 we reported the average value of τ_{opt} (along with standard deviation) and its median value. In case of a range of values corresponding to the maximal average solution quality, we considered the highest extreme of the range. Statistics are computed by running PGSAT(τ) 1000 times on each instance and averaging over 10 different instances.

We first observe that for fixed instance size, the value of τ_{opt} considerably vary and τ_{opt} is non-increasing with $\overline{q}$. Moreover, we can observe that the values of τ_{opt} related to a particular $\overline{q}$ value, are approximately the same independently of the instance size.

From these results we conjecture that, for random 3-SAT instances, the structure parameter which affects τ_{opt} is the normalized average connectivity. The behavior of PGSAT on k-SAT instances, with k arbitrary is subject of ongoing work.

The testbed used, uniform randomly generated formulas, have the disadvantage of imposing an average evaluation of the connectivity, supposing negligible the variance of the individual node degree. Moreover, the SAT/MAXSAT instances considered in practice are not random but rather structured and often characterized by non random distributions of node degree. In order to overcome the drawbacks of experimenting on random instances, we investigated in two opposing directions. In the following sections we will first show the results obtained on SAT instance with *SATgraph* characterized by a fixed node degree (i.e., $q_i = q, i = 1, \ldots, n$). Then, in Sec.7, we will address the question of whether and under which conditions the phenomenon discovered for random instances is still present in structured ones.

6 Constant-degree 3-SAT instances

In the previous sections, we have characterized the connectivity of random 3-SAT instances with the average connectivity $\overline{q}$, which seems to be enough informative for uniform generated instances. In random 3-SAT instances, nodes have in general different degree, even though the values are mainly concentrated around the mean.

In order to compare the node degree distribution between instances, we consider the frequency of node degree $Freq(j)$ = 'frequency of a node connected to exactly j nodes' and the cumulative frequency $CumFreq(j)$ = 'frequency of a node connected to not more than j nodes'. Fig.11 shows a typical case of cumulative frequency vs. the normalized node degree for random 3-SAT instances retrieved from SATLIB. The instances belong to the threshold region (clauses/variables ≈ 4.3) and are satisfiable. The graph corresponding to a random 3-SAT instance is a random graph $G_{n,p}$ [32], where n is the number of nodes and p is the probability that any pair of nodes are connected. In fact, uniform random 3-SAT instances of SATLIB are generated by randomly selecting, for each

Table 6: Average τ_{opt} for instances with fixed $\overline{q}$. In case of different values of τ enabling PGSAT to achieve the maximal average solution quality, we reported the maximum value.

n	$\overline{q}$	$\langle\tau_{opt}\rangle$	std.dev.	median
20	0.2	10.0	0.0000	10
	0.3	8.9	1.3703	9
	0.4	6.0	0.9428	6
	0.5	4.7	0.9486	5
	0.6	3.3	0.8232	4
	0.7	2.9	0.5676	3
	0.8	2.4	0.6992	3
	0.9	2.0	1.0541	3
	1.0	2.3	0.8233	3
50	0.2	11.10	0.7378	11
	0.25	9.10	1.1005	10
	0.3	6.80	0.9189	7
	0.4	4.60	0.6992	5
	0.5	4.50	0.8498	5
	0.6	4.00	0.4714	4
	0.7	4.20	0.4216	4
	0.8	3.90	0.5676	4
	0.9	4.00	0.0000	4
	1.0	3.90	0.5676	4
100	0.08	22.00	1.4142	22
	0.1	19.40	0.6992	20
	0.15	12.10	1.1005	13
	0.2	8.20	1.0327	8
	0.3	6.20	0.4216	6
	0.4	5.50	0.5270	6
	0.5	5.50	0.5270	6

clause, three literals among the complete set of $2n$ literals[11]. Thus, every pair of variables has the same probability to belong to a same clause. For random graphs such as $G_{n,p}$, the distribution probability of connectivity follows a Poisson distribution, i.e.,

$$\text{prob}\{\text{a node is connected exactly to other } j \text{ nodes}\} = e^{-\lambda}\lambda^j/j!$$

where the parameter λ is the expected node degree, therefore, in our case, $\lambda = (n-1)\overline{q} = q$. For instance, in Fig.12 the frequency of a 3-SAT instance with 100 variables is plotted.

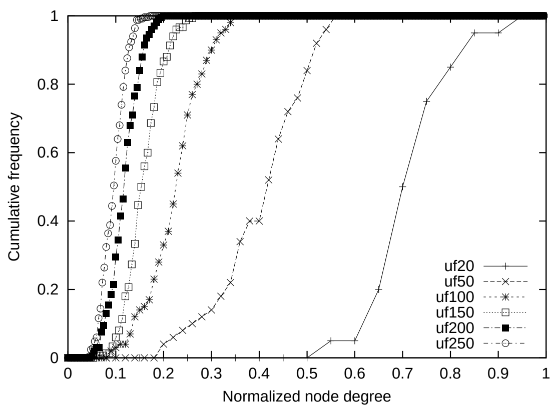
In order to perform experiments with the minimum amount of parameters which can vary, we devised a method to generate random k-SAT instances (for our purposes, $k = 3$) where variables have the same node degree.

Instead of starting from a SAT formula, expressed as a conjunction of clauses with fixed number of literals, we start from a graph with the desired connectivity properties and we use it as a skeleton for generating a SAT formula. The procedure used to generate constant-degree k-SAT instances is described in Appendix A.

6.1 Experimental results

We have tested PGSAT on a benchmark composed of constant degree 3-SAT instances. The benchmark is composed of six sets of thirty instances. Each set contains instances with the

[11]The description of the generation procedure is available at `www.satlib.org`.



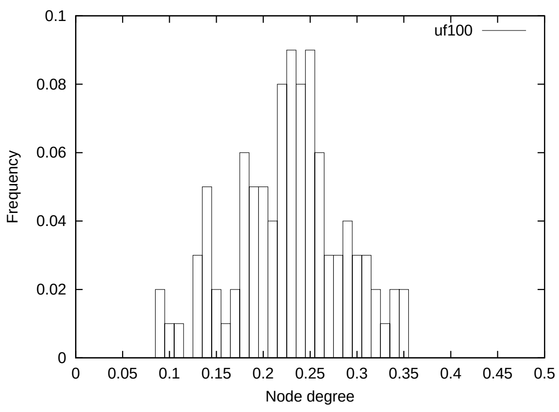


Figure 11: Cumulative frequency vs. normalized node degree for Uniform Random 3-SAT instances in the threshold region.

Figure 12: Frequency against normalized connectivity in random 3-SAT instance uf100-01. The instance has an average normalized connectivity of 0.229.

same number of variables (n =20, 50, 100, 200, 400, 600) with varying ratio between clauses and variables ($m/n = 3, 4, 5$). The connectivity q is equal to 4, so we have: $\overline{q}_{20} \approx 0.211$, $\overline{q}_{50} \approx 0.082$, $\overline{q}_{100} \approx 0.040$, $\overline{q}_{200} \approx 0.020$, $\overline{q}_{400} \approx 0.010$, $\overline{q}_{600} \approx 0.007$. For each pair ($m$,$n$) we randomly generated ten instances.

We run PGSAT 1000 times for every instance. When the ratio m/n is in the proximity of the critical value 4.3 or above it, the instances are likely to be unsatisfiable.

Results are plotted in Figs.13, 14, 15 and 16. On the left we reported the average error against the parallelism, while on the right the rank of parallelism is plotted against the parallelism itself, i.e., we ranked the values of τ depending on the average error. The lower the rank, the better.

We observe two main points. First, even for instances with the same number of variables and clauses, the optimal parallelism is not exactly the same. Indeed, we can observe that τ_{opt} is confined in a range of values. This fact can be attributed to the variance among random generated instances and the simplifications introduced by the *SATgraph*[12]. Furthermore, in some preliminary measurements, we observed that those differences may be explained by differences in the ruggedness of the search landscape.

A second important observation is that the range of optimality is practically the same independently of the ratio m/n. This second observation supports the conjecture that the optimal parallelism mainly depends on the instance connectivity and it is independent of other parameters, such as the ratio m/n.

7 Structured Instances

In this section, we show experiments performed on structured SAT instances. First, we experimentally show that for structured instances there exists an optimal value of parallelism that enables the algorithm to reach the optimal performance. Second, by analyzing the frequency of node degree of the graphs associated with the SAT instance, we observe that an asymmetric and not regular distribution strongly affects the algorithm performance when τ is concerned.

7.0.1 Experimental Results: structured vs. random instances

In this section we first analyze the node degree distribution of graphs associated to structured SAT instances. Then we show and discuss results of PGSAT, run with different values of τ, on

[12]We remind that one *SATgraph* corresponds to a set of SAT instances.

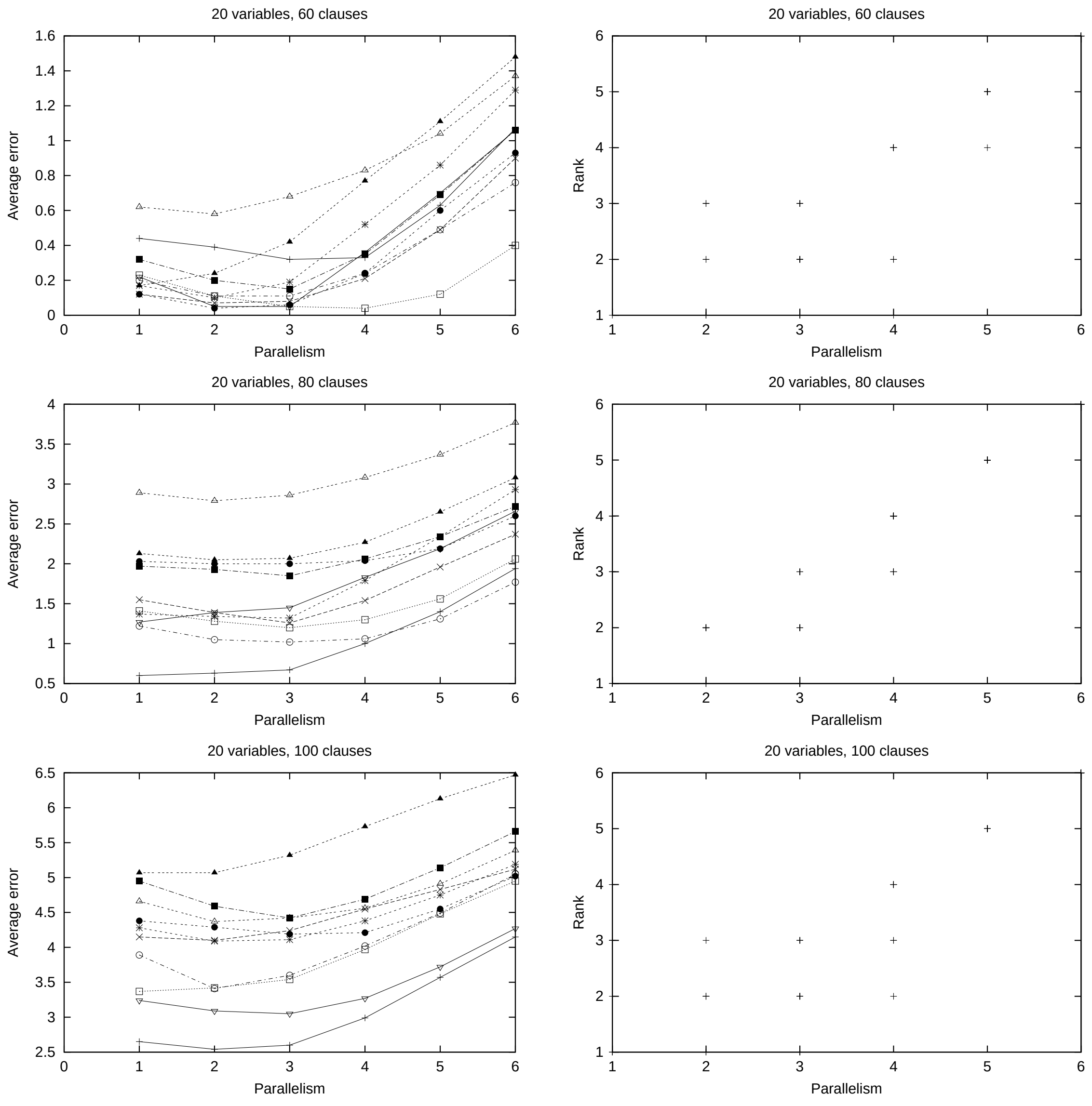


Figure 13: Left: Parallelism vs. Average error (number of unsatisfied clauses). Right: Parallelism vs. rank (the lower, the better).

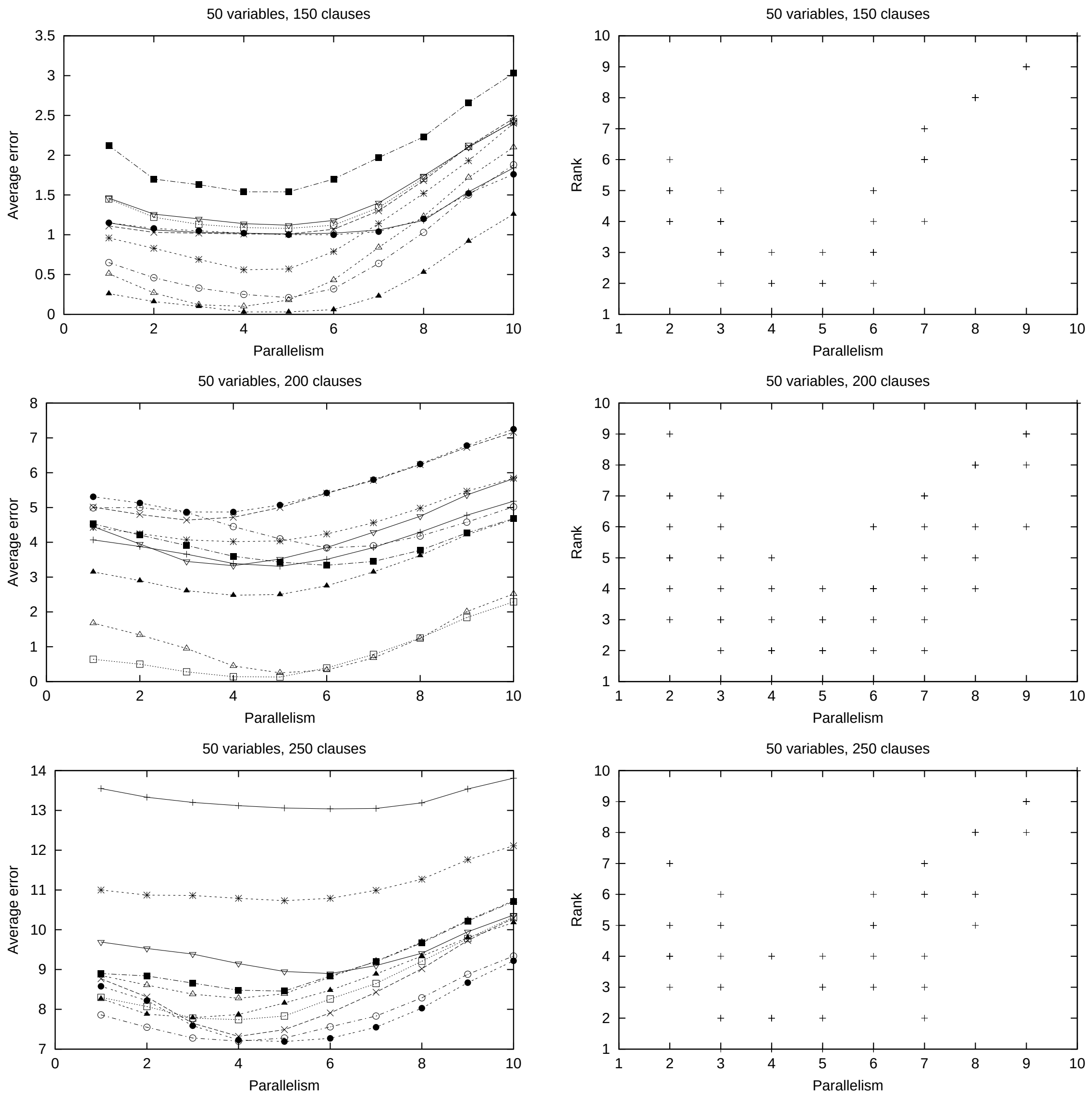


Figure 14: Left: Parallelism vs. Average error (number of unsatisfied clauses). Right: Parallelism vs. rank (the lower, the better).

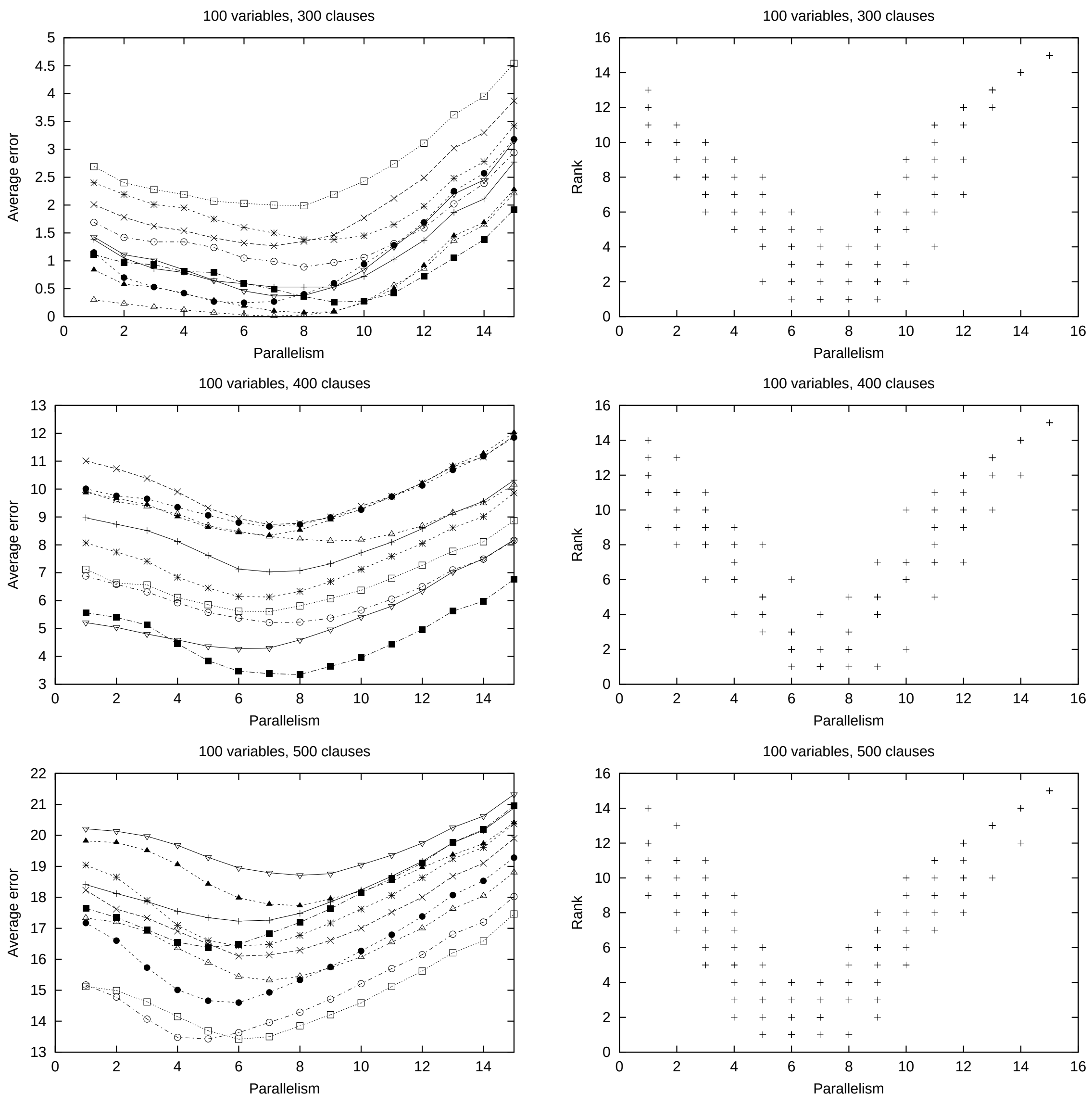


Figure 15: Left: Parallelism vs. Average error (number of unsatisfied clauses). Right: Parallelism vs. rank (the lower, the better).

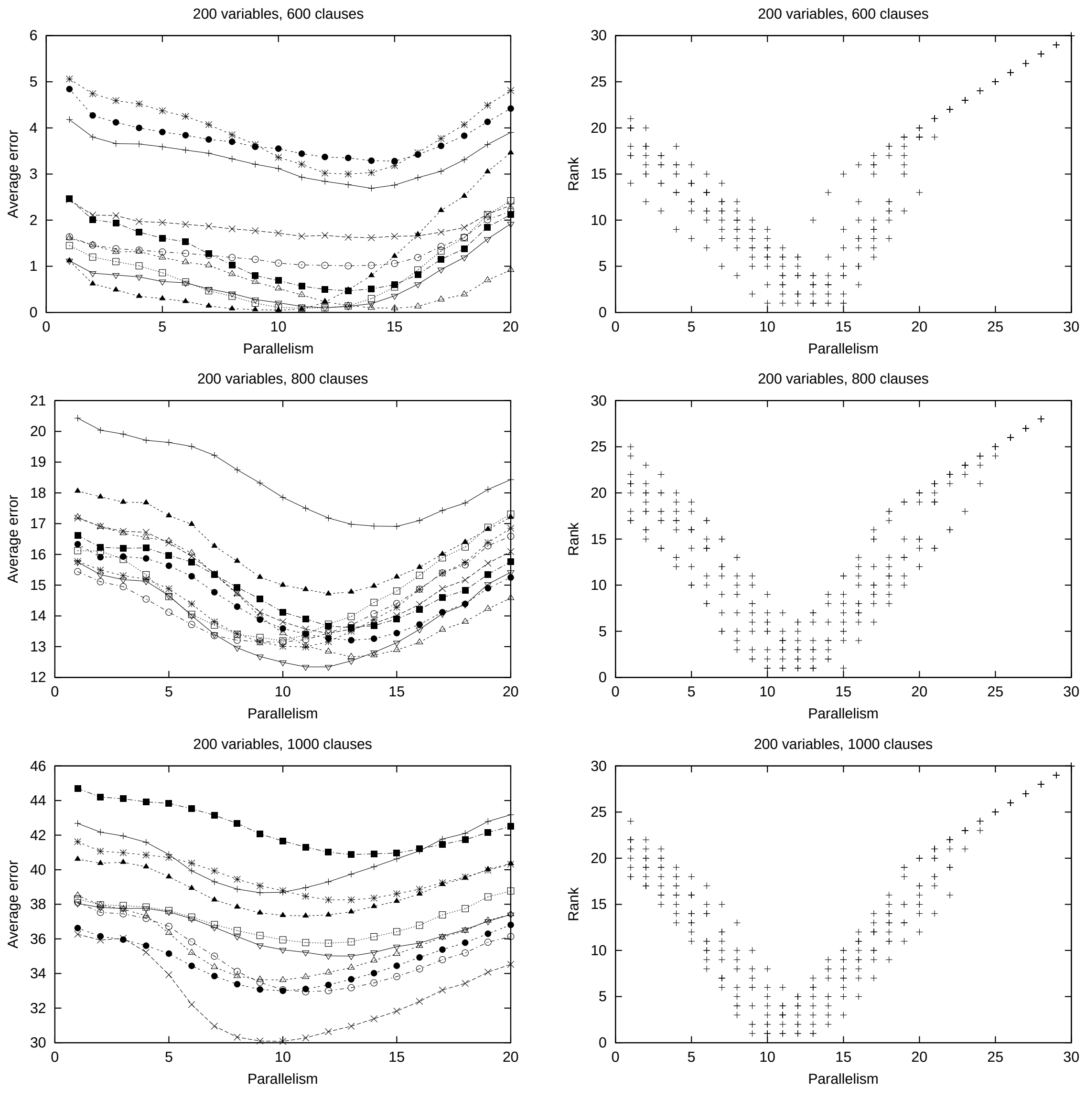


Figure 16: Left: Parallelism vs. Average error (number of unsatisfied clauses). Right: Parallelism vs. rank (the lower, the better).

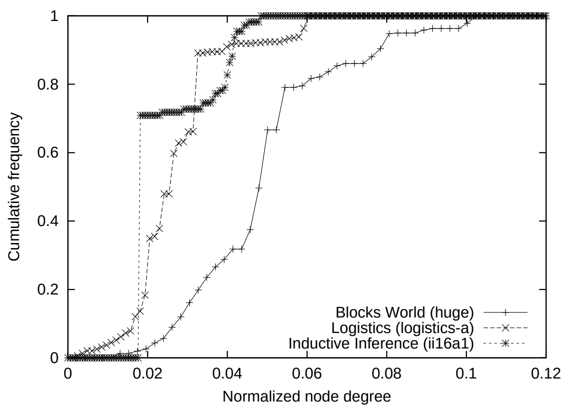


Figure 17: Cumulative frequency vs. normalized node degree in structured instances.

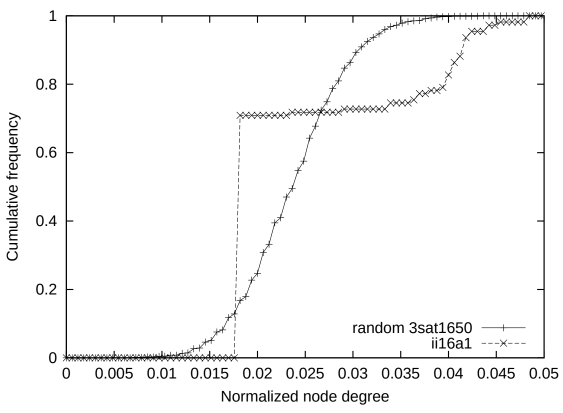


Figure 18: Cumulative frequency vs. normalized node degree in a structured instance (*ii16a1*) and in a random 3-SAT instance *3sat1650* of same size ($n = 1650$) and connectivity ($\overline{q} \approx 0.0239$).

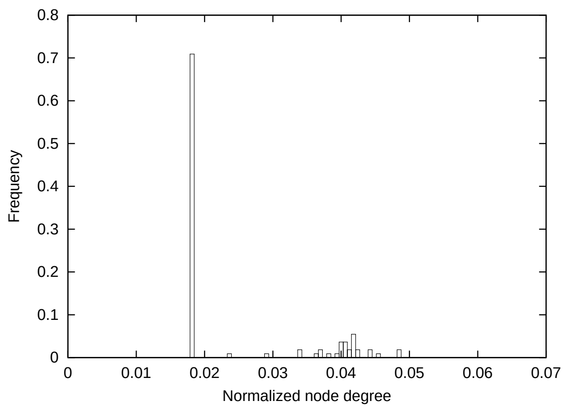


Figure 19: Frequency vs. normalized node degree in the instance *ii16a1*.

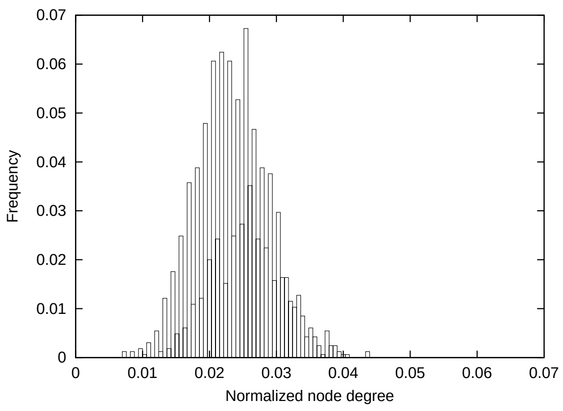


Figure 20: Frequency vs. normalized node degree in the instance *3sat1650*.

two representative structured SAT instances.

Structured instances are characterized by the presence of some regularity in their components, for example graph problems based on ring or lattice topology, SAT problems obtained by encoding logic problems (circuit testing, inductive inference, planning, etc.). In SAT problems generated by an encoding procedure from other problems there are two sources of structure: the inherent structural properties of the original problem and the relations among variables introduced by the encoding procedure. As noted in [5], the inherent structure of the problem could be partially lost in the encoded formulation. However, independently of the origins of structure, the SAT instances we consider clearly show node degree distributions very different with respect to random instances. Fig.17 shows typical cases of the curve of cumulative frequency for structured SAT instances taken from SATLIB, produced by encoding a Blocks World Planning Problem (*huge*), a Logistics Planning Problem (*logistics-a*) and Inductive Inference (*ii16a1*). The plotted curves show apparent differences with those of random SAT problems. They are not as regular as random ones and they have gaps and plateaus, especially in the uppermost part of the curve. Structured instances have in general a more spread and non-uniform connectivity distribution.

The instance *ii16a1* is the most peculiar and differs the most from that of random instances.

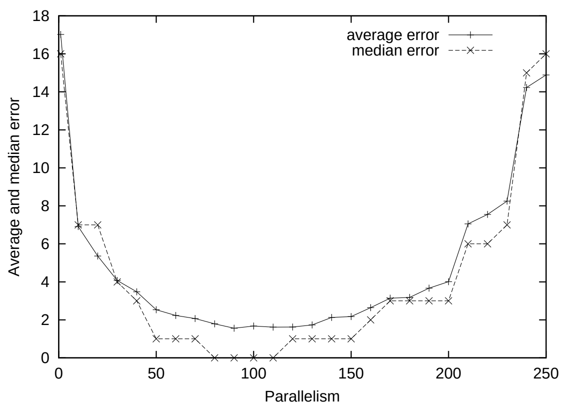


Figure 21: Average and median error against τ for PGSAT on the instance *ii16a1*.

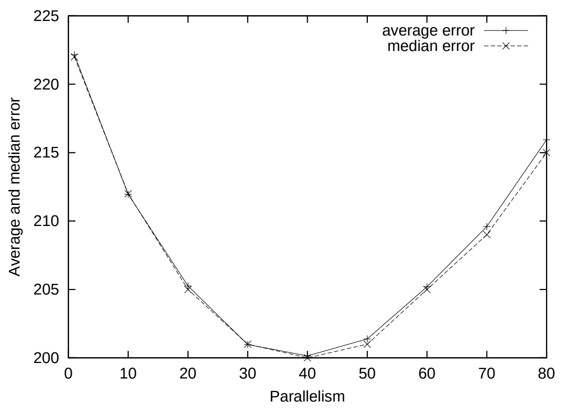


Figure 22: Average and median error against τ for PGSAT on the instance *3sat1650*.

It has 1650 variables and a normalized average connectivity $\overline{q}_{ii16a1} = 0.0239$. Its cumulative frequency is shown in Fig.18, along with the cumulative frequency of a random 3-SAT instance of the same size and normalized connectivity (instance *3sat1650_q-const*[13]). Fig.19 and Fig.20 plot the respective frequency of node degree. We can note that the node degree frequency of the structured instance is highly asymmetric and has a peak close to 0.018, corresponding to the large gap in the cumulative frequency. Therefore, *ii16a1* has a very large number of nodes with lower connectivity than the average. Conversely, the node degree frequency of the random instance is regular (it approximately fits the Poisson distribution with high mean) and the highest peak in frequency is very close to the mean.

Fig.21 and Fig.22 show the average and median error (number of unsatisfied clauses) respectively on *ii16a1* and *3sat1650_q-const* for PGSAT with different values of τ. Results are averaged over 500 trials. We first observe that also for the structured instance there exists an optimal value of τ. Nevertheless, despite the fact that the two instances have the same average connectivity, the optimal parallelism is higher for *ii16a1* than for *3sat1650_q-const*. We conjecture that the high number of nodes with low degree present in the instance *ii16a1* is the cause of higher optimal parallelism.

We performed the same kind of experiments on the *logistics-a* instance, by comparing it with a random 3-SAT instance with the same size (828 variables) and normalized average connectivity $\overline{q} = 0.0275$ (instance *3sat828*). Fig.23 and Fig.24 show the respective node degree frequency. Note that, while the distribution of the instance *3sat828* follows a Poisson distribution, the distribution of *logistics-a* is not regular and has a high peak at normalized node degree 0.0326. The results of PGSAT performance with respect to τ are plotted in Fig.25 and Fig.26. We can observe that also for this structured instance there is a value of τ leading to a minimum average error, but in this case the optimal parallelism for *logistics-a* is lower than that of the random instance. This difference may be explained by observing that the highest peak in *logistics-a* node degree frequency corresponds to a node degree higher than the average value, which characterizes the random instance.

7.1 Morphing from random to structure

In Sect.4.2 we have seen that the average connectivity affects the optimal parallelism for random instances. The results of the previous section show that for structured instances, characterized by asymmetric frequency distribution, the optimal parallelism seems to be affected by the highest peaks of the node degree distribution. To investigate more deeply this difference, we generated instances with a controlled amount of *structure*, by means of a technique called *morphing* [11].

[13]This instance is generated with the procedure described in Sec.5.

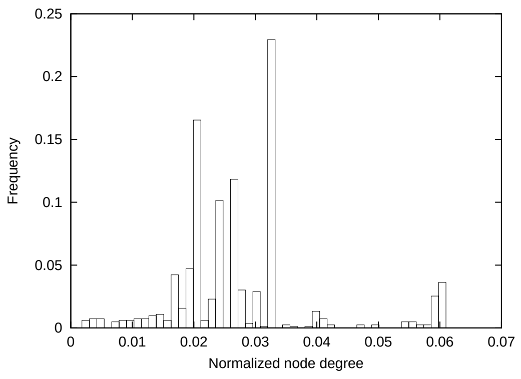


Figure 23: Frequency vs. normalized node degree in the instance *logistics-a*.

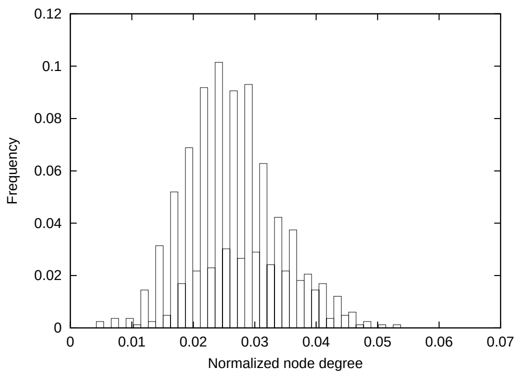


Figure 24: Frequency vs. normalized node degree in the instance *3sat828*.

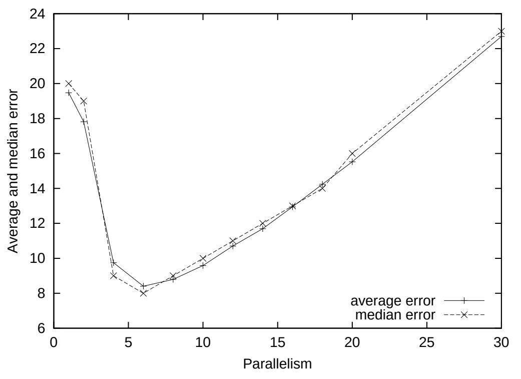


Figure 25: Average and median error against τ for PGSAT on the instance *logistics-a*.

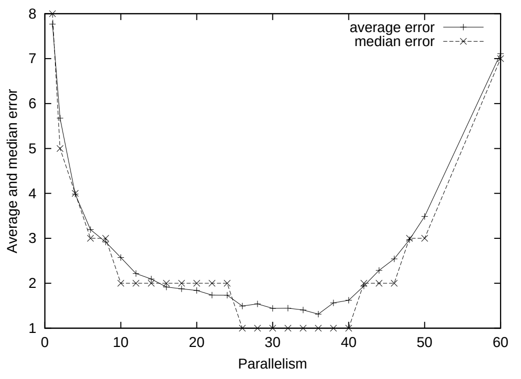


Figure 26: Average and median error against τ for PGSAT on the instance *3sat828*.

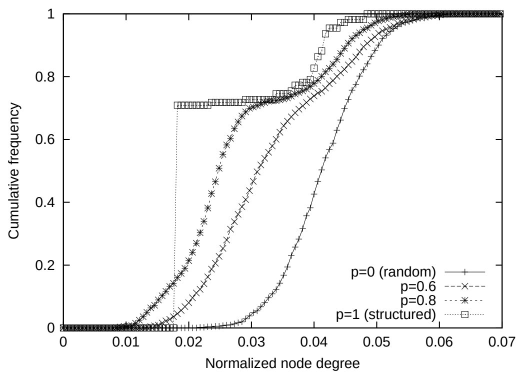


Figure 27: Cumulative frequency vs. normalized node degree for instances generated by morphing. $p \in [0,1]$ controls the randomness/structure ratio: the higher p, the higher the amount of structure.

This method enables to generate instances gradually morphing from source to destination instance by varying a parameter $p \in [0,1]$. The lower p used to generate an instance, the more similar to the source. To be applied on SAT problems, the method needs instances with the same number of variables (n) and clauses (m). A new SAT instance is generated by selecting each of the m clauses either from the source or the destination. The clause is chosen from the destination instance with probability p. We generated a satisfiable random 3-SAT instance with 1650 variables and 19368 clauses (*3sat1650_large*[14]), with the same size and number of clauses of the instance *ii16a1*. With $p = 1$ we obtain *ii16a1* and *3sat1650_large* with $p = 0$. Since p controls the number of clauses belonging to the structured instance *ii16a1*, it also measures the amount of structure in the generated instance. Fig.27 shows the node degree cumulative frequency of source (random), destination (structure) and instances generated by the morphing method. The node degree frequency of the considered instances is plotted in Fig.28. The results of 500 trials of PGSAT for different values of parallelism are shown in Fig.29. Observe that the optimal parallelism increases with p, therefore we can conjecture that the more similar the node degree frequency distributions of two instances, the closer should be their optimal parallelism degrees.

In summary, we have found evidence of the fact that an optimal parallelism value exists also for structured instances. We conjecture that this value is strongly affected by the asymmetric frequency of node degree. In particular, we observe that the highest peaks location seem the most relevant characteristic influencing the optimal parallelism. Nevertheless, a more extensive experimental analysis is required to prove our conjecture.

[14]This instance is different from the instance *3sat1650_q-const* previously used for a comparison with a structured instance

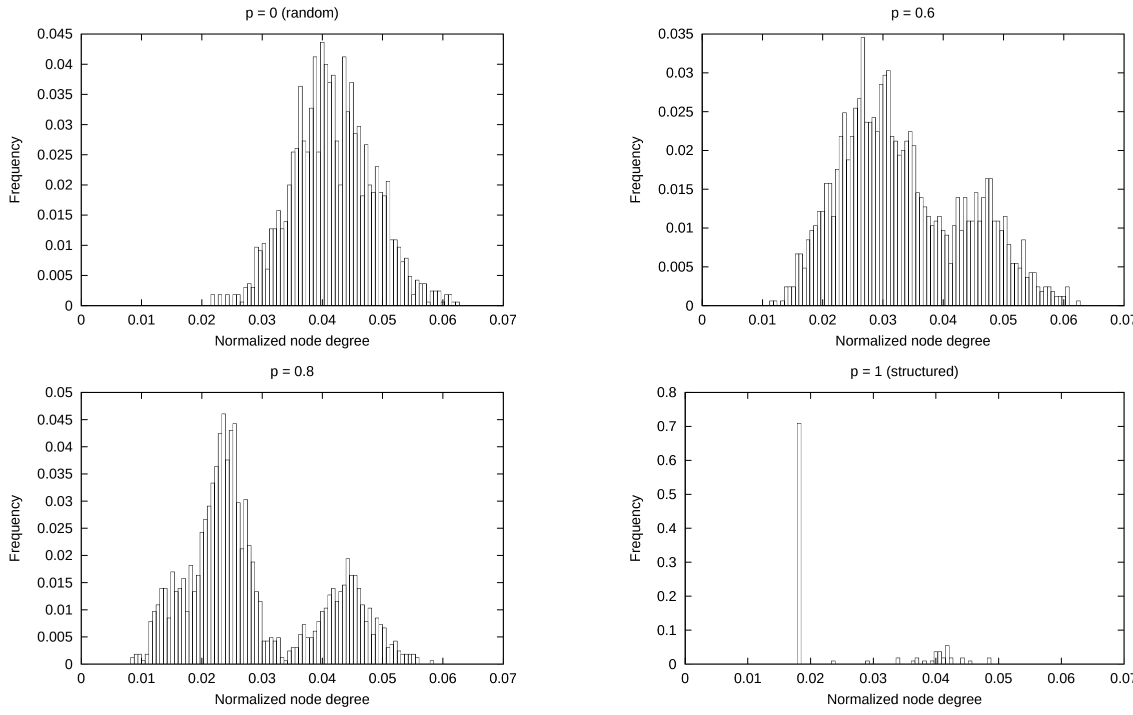


Figure 28: Frequency against normalized node degree for instances generated by morphing from a random instance (*3sat1650_large*) to a structured one (*ii16a1*).

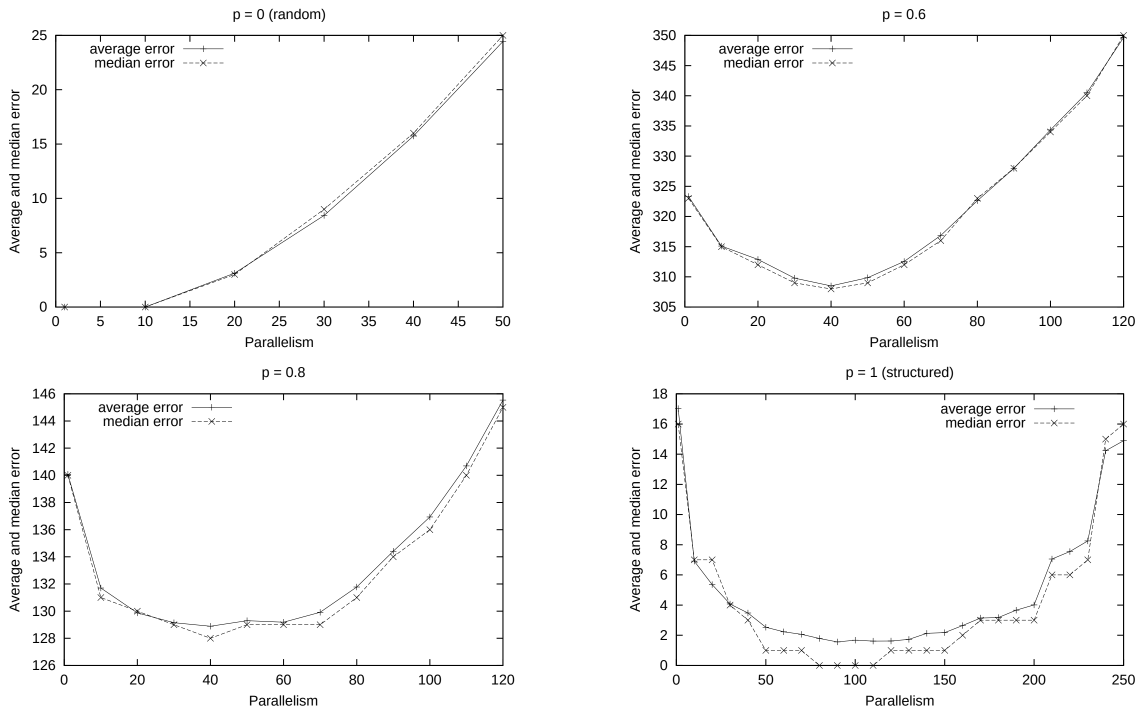


Figure 29: Average and median error of PGSAT run with different values of τ on instances generated by morphing from a random instance (*3sat1650_large*) to a structured one (*ii16a1*).

8 Discussion

The results presented in the previous sections clearly show that PGSAT exhibits a behavior with a *bad-good-bad* pattern as a function of τ. Moreover, this behavior is shown across different typologies of instances. It can also be asserted that τ_{opt} is affected by the node degree distribution and we could observe that the higher the connectivity, the lower τ_{opt}. A general analytical or empirical relation is still lacking, though. In this section, we discuss the limits of the range of applicability of our results. Furthermore, we will consider similarities and differences between our experiments and the criticality and parallelism phenomenon.

8.1 Limits of the *SATgraph*

As observed in Sec.3, the *SATgraph* represents a very simplified view of the structure of a SAT instance. First of all, it does not capture the tightness of constraints, since it only checks *if* two variables are involved in the same clause and it does not take into account the number of these clauses, nor the number of literals per clause. Moreover, the *SATgraph* focuses only on the network topology of a SAT instance and it is blind to structural instance properties related to the solution space, such as *backbones* [31] and *backdoors* [53]. This approximation introduced by the *SATgraph* poses a limit on the range of applicability of our results. For instance, in preliminary experiments on random k-SAT instances, we observed that τ_{opt} is not independent of the clause length, since we have found different values for τ_{opt} for different clause lengths.

Furthermore, we found some important exceptions to the hypothesis that τ_{opt} is negatively correlated with $\overline{q}$. We artificially generated random SAT instances composed of a *core* instance to which we added a number of clauses composed of n literals[15]. The core instance is a random 3-SAT instance of the SATLIB benchmark. Adding one or more clauses with n literals means to change the average normalized connectivity of the *SATgraph* from a fractional value to 1, since the *SATgraph* becomes completely connected. We would expect that for such instances τ_{opt} was very close to 1, but this is not what we observed. Indeed, τ_{opt} still maintains the same value it was found for the core instance, even if we add a large number of n-literal clauses. These results clearly show that the core instance plays a role that is not captured by the *SATgraph*. A possible explanation is that the added clauses do not perturb the solution space and the fundamental characteristics of the search landscape. This conjecture drives us directly to the analysis of results obtained on structured instances in Sec.7. The question arising is the following: since the structured instances considered have clauses of different length, why does they seem to be in agreement with the original hypotheses? Are the clauses of such instances all (or almost all) affecting the search space? To answer these questions a deeper analysis has to be conducted.

8.2 Criticality in local search for SAT?

Concerning the question of whether the criticality and parallelism phenomenon can be observed also in multi-move local search for SAT, we can not definitely conclude that the phenomenon is exactly the same found in [27].

The main reason is that, in the original work, the authors found an order parameter for which a critical value exists such that a phase transition is observable. This order parameter is the average entropy of the system and it is defined as follows. When the algorithm has reached a steady state (i.e., it is stagnating) the probability that variable x_i flips is given by p_i. Then, the entropy of variable x_i is evaluated: $S_i = -p_i \log_2 p_i - (1-p_i)\log_2(1-p_i)$. The order parameter is the average entropy $S = \frac{1}{n}\sum_{i=1}^{n} S_i$. When the algorithm is run with the optimal degree of parallelism, S shows an abrupt transition from 0 to 1 and it can be formally shown that a phase transition takes place.

In our experiments, the definition of an analogous parameter is not possible. In fact, in PGSAT, τ variables are always flipped at each iteration, since a move is performed even if it reduces the number of satisfied clauses. Therefore, the entropy of the system would be just proportional to

[15]where n is, as usual, the number of variables

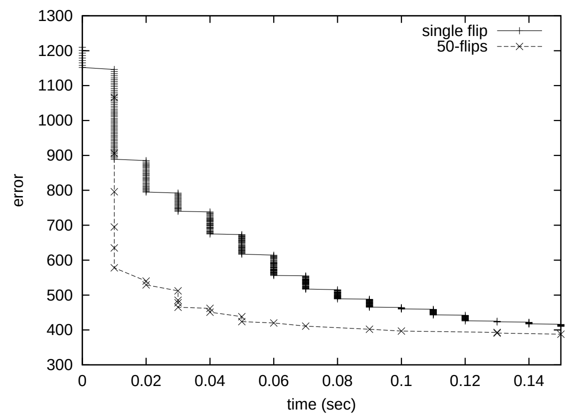


Figure 30: Single flip vs. multi-flip GSAT on an unsatisfiable 3-SAT instance with 1000 variables and 10000 clauses.

τ and it would not give indications on the energy of the system. These considerations have been also experimentally confirmed.

Alternative order parameters should be defined, in order to check if the optimal parallelism is also linked to a phase transition. Nevertheless, we believe that such a phase transition does not exist, since we did not observe abrupt changes in the behavior of PGSAT. Indeed, it is important to observe that the algorithm we applied is more effective than simple hill climbing and simulated annealing on SAT/MAXSAT instances, therefore the abrupt change in the behavior might be smoothed (as, indeed, is apparent from the plots in the previous sections).

8.3 Improving local search performance

Besides the interest about the phenomenon itself, this kind of parallelization can be used to improve local search, in which it is possible to perform more than one local move at a time. The intuition behind the effectiveness of parallel local moves is that multi-flip moves help to escape from local minima and reach faster search space regions with low objective value. This behavior can be presumably explained by the fact that, at τ_{opt}, the distance between a state and its successor, reached after a multi-flip move, enables local search to achieve the optimal trade-off between intensification and diversification.[16] This conjecture is supported by experimental results in [24, 23] and by experiments we performed on large (unweighted) MAXSAT instances. For example, consider Fig.30 where the initial iterations of PGSAT with single and multi-flips are compared. The algorithm is run on an unsatisfiable 3-SAT instance with 1000 variables and 10000 clauses, with $\tau = 1$ and $\tau = 50$. As we can note, the multi-flip version achieves larger solution improvements than the single flip one in the same amount of time.

Since up to now theoretical results are still missing to compute the optimal parallelism value for a given instance, we need an empirical method to tune τ as close as possible to τ_{opt}. The easiest way would be to run PGSAT for $\tau = 1, 2, \ldots$, using a small cutoff value and stop as soon as a minimum average error has been found. Nevertheless, this method is, in general, computationally expensive, as it might require several runs before finding τ_{opt}. Indeed, since the algorithm is stochastic, several runs have to be performed before achieving significant statistics.

We developed another method which requires just one short run for every candidate for τ_{opt}. This method is based on the observation of solution improvements PGSAT achieves during one execution. From a mathematical standpoint, our objective is to find the minimum of a curve $\epsilon(\tau)$ representing the average error ϵ with respect to τ. From the observation of experimental data,

[16]We refer to the informal definition of *intensification* as the greedy exploitation of search history and *diversification* as the exploration of the search space. For a discussion on this topic, we forward the interested reader to [6].

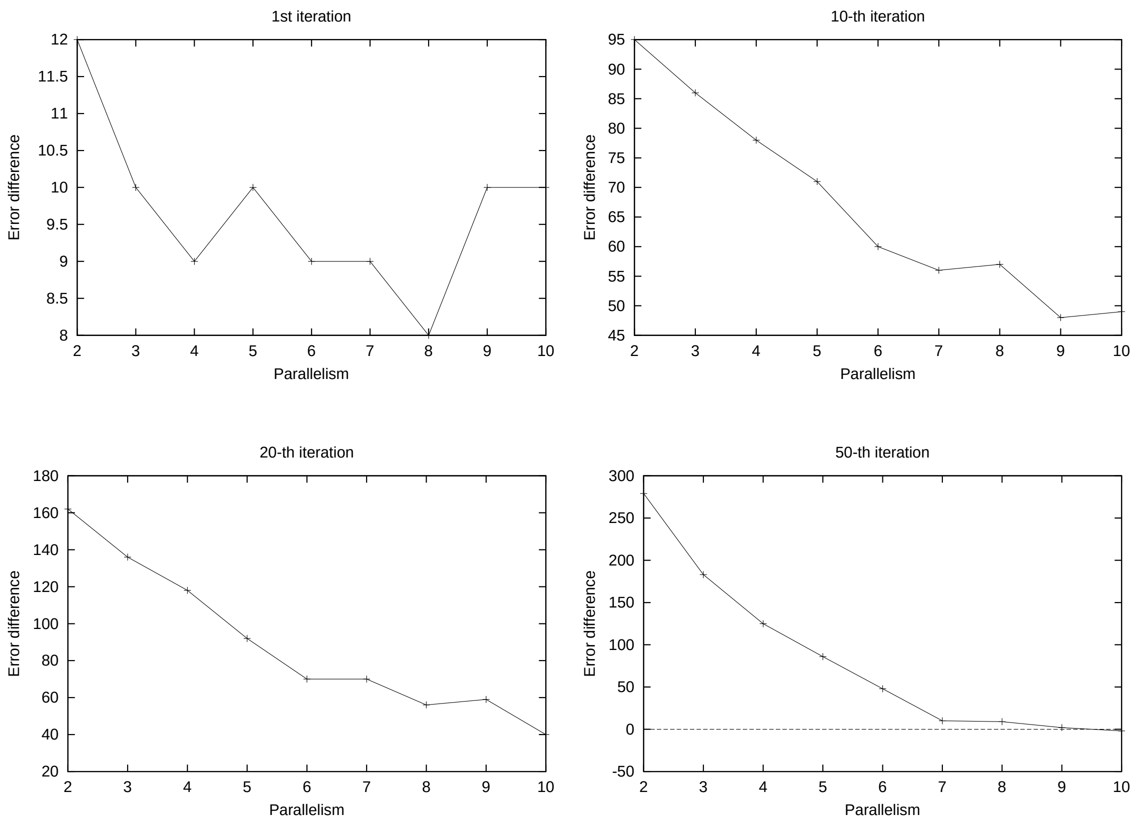


Figure 31: Difference of the error (number of unsatisfied clauses) reached after $1, 10, 20, 50$ iterations between successive values of parallelism. For each point (τ, y) in the plot, the coordinate y is given by $\epsilon(\tau - 1) - \epsilon(\tau)$, where $\epsilon(\tau)$ is the number of unsatisfied clauses returned by PGSAT with parallelism τ.

we can assume that the curve is convex and with one minimum. Therefore, τ_{opt} is given by the abscissa τ such that the derivative of the curve is zero, i.e., $\frac{d\epsilon(\tau)}{d\tau}|_{\tau=\tau_{opt}} = 0$. Since the values of τ are integer, an approximation of the derivative is $\Delta\epsilon(\tau) = \epsilon(\tau) - \epsilon(\tau - 1)$. Thus, the objective is to find $\overline{\tau}$ such that $\epsilon(\overline{\tau}) - \epsilon(\overline{\tau} - 1) \approx 0$. Fig.31 shows typical results obtained with short runs of PGSAT on the *logistics-a*[17], for which $\tau_{opt} \approx 6$. For every run length $(1, 10, 20, 50)$ we stored the objective function value (number of unsatisfied clauses) reached by PGSAT with different values of τ. For each point (x, y) in the plot, the coordinate y is given by $\epsilon(x - 1) - \epsilon(x)$. For example, let us consider the *1st iteration* curve. After one iteration, the difference of the errors returned by PGSAT with $\tau = 1$ and $\tau = 2$ is 12, i.e., $\epsilon(1) - \epsilon(2) = 12$. All the runs have the same initial state and the algorithm uses the same random number generator seed. We can observe that at the *50th iteration* $\epsilon(\tau - 1) - \epsilon(\tau) \approx 0$ for $\tau = 9$ and the error difference approaches zero at $\tau = 7$. Moreover, we can observe that, except for the plot of the *1st iteration*, the curve decreases approximately linearly up to a point where the slope decreases quite abruptly. This point corresponds, with good approximation, to τ_{opt}. The fact that at the knee of the curve $\Delta\epsilon$ is not zero can be explained by observing that the solution improvements achieved with high parallelism (even $\tau > \tau_{opt}$) at the beginning of the search are higher than those of the remaining part of the search. The results obtained yield to the definition of a simple procedure to tune τ: run PGSAT for a very small number of iterations for different values of τ and set it at the value corresponding to $\epsilon(\tau - 1) - \epsilon(\tau) \approx 0$. Alternatively, we can set τ to the value corresponding to the knee in the

[17]We have chosen *logistics-a* just as a representative example of the experiments we made on the considered instances.

curve, thus further reducing the required number of iterations. Up to now, we have determined the near-optimal value of τ by directly observing the plotted data, but the development of an automatic procedure is subject of current work.

9 Related Work

This work is inspired by work of Kauffman et al. on criticality and parallelism [26, 24, 23] and Monte Carlo algorithms for NK-models [9]. Even though not with the same objective, other works deal with multiple flips in local search for SAT and MAXSAT problems. Yagiura and Ibaraki [54] present a computational study of r-flips neighborhoods ($r = 2, 3$). Parkes [34] considers the parallelization of WalkSAT [27, 43] for 3-SAT instances in the underconstrained region and observes that parallel flips does not degrade the local search, indirectly confirming our results. Indeed, in the underconstrained region the connectivity is low, therefore the optimal parallelism is high. Finally, Strohmaier [46] discusses the implementation of GSAT on a multi-flip neural network and Milano and Roli [29, 28] present a general method for tackling SAT with boolean networks, explicitly considering the possibility of multi-flips local search algorithms.

10 Conclusion and Future Work

In this work, we first showed that for different classes of problems (random and structured SAT, MAXSAT, lattice SAT) there is a value of τ that optimizes the algorithm performance in terms of average solution quality. Furthermore, it has been discovered that, given a *SATgraph*, its connectivity strongly affects the optimal value of τ_{opt}. In case of constant node degree and random instances, the higher the normalized connectivity $\overline{q}$, the lower τ_{opt}. In case of small differences among instances with the same connectivity, we have conjectured that the fitness landscape correlation provides a justification for differences in τ_{opt}. These results are in accordance with the previous results found in the literature.

Besides these experiments, the impact of *SATgraph* structure has been investigated by applying PGSAT to structured SAT instances. In our experiments, we have observed that for structures instances there exists an optimal value of τ, as in the case of random and constant degree SAT instances. Moreover, we have found clues for the dependence of τ_{opt} on the node degree frequency, and in particular on the frequency peaks.

The relation between graph connectivity and multi-move local search is still to be deeply understood. In fact, we have also outlined some limitations of the use of the *SATgraph* and more accurate analyses are required. (For example, the use of different kinds of graphs, such as weighted graphs, to capture the structure of the instances and empirical analysis on k-SAT instances, with k $\neq$ 3.)

There are very interesting open issues to explore, such as the exploitation of graph properties to define the subsets of strongly connected variables (instead of dividing variables at random) and the analytical study of connectivity and optimal parallelism. Moreover, another interesting issue is the investigation of the relation between criticality and parallelism and the introduction of noise in local search. Results of a preliminary study on this subject are reported in Appendix B.

Other characteristics of the graph associated with a SAT instance have to be studied, e.g., variance of node degree, maximum and minimum degree, clustering factor and small-world properties [51, 50, 47]. The relation between these properties and multi-flip local search has still to be investigated. Moreover, the analysis of graph parameters is very important to understand algorithms strengths and weaknesses with respect to the instance they are tackling and to optimize parameters and heuristics.

The application of the results of this paper goes beyond the simple empirical investigation of hill climbing-like procedures. Algorithm other than GSAT might benefit from multi-flips application, for example GWSAT [43] and WalkSAT, more effective on SAT than GSAT itself. We are currently exploring different ways of parallelizing local search, for example by probabilistic flips, such that each variable is flipped with a probability that depends on the increase of number of satisfied clauses. Moreover, we are studying the relations between SAT/MAXSAT instance connectivity and properties of the search landscape. There are other very interesting open issues to explore, such as the analytical study of connectivity and optimal parallelism. Finally, we may suppose that there could be an analogous phenomenon also when complete algorithms are applied. The critical parameter can be, for instance, the randomization degree, or the number of assignments performed at each construction step.

Acknowledgments

I would like to thank Marco Dorigo, Michela Milano and Thomas Stützle for useful comments on this work.

References

[1] D. Achlioptas and C. Moore. The asymptotic order of the random k -SAT threshold. In *Proceedings of FOCS 2002*, pages 779–788, 2002.

[2] Y. Bar–Yam. *Dynamics of Complex Systems.* Studies in nonlinearity. Addison–Wesley, 1997.

[3] R. Battiti and G. Tecchiolli. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[4] J.C. Beck and M.S. Fox. Dynamic problem structure analysis as a basis for constrained -directed scheduling heuristics. *Artificial Intelligence*, 117:31–81, 2000.

[5] R. Béjar, A. Cabiscol, and C.P. Gomes C. Fernàndez, F. Manyà. Capturing structure with satisfiability. In T. Walsh, editor, *CP 2001 – 7th International Conference of Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 137–152. Springer, 2001.

[6] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, September 2003.

[7] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI91*, pages 331–337. Morgan Kaufmann, 1991.

[8] R. L. Devaney. *An introduction to chaotic dynamical systems.* Addison–Wesley, second edition, 1989.

[9] F. M. Dittes. Optimization on rugged landscapes: A new general purpose monte carlo approach. *Physical Review Letters*, 76(25):4651–4655, June 1996.

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

[11] I. P. Gent, H. H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proceedings of AAAI99*, pages 654–660, 1999.

[12] I. P. Gent, E. MacIntyre, P Prosser, and T. 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.

[13] C.P. Gomes. Structure, duality, and randomization – common themes in AI and OR. In *Proceedings of AAAI00*, 2000.

[14] C.P. Gomes and B. Selman. Problem structure in the presence of perturbations. In *Proceedings of AAAI97*, 1997.

[15] C.P. Gomes and D. Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and Extensions*, 2002.

[16] A. Guerri and M. Milano. Learning techniques for automatic algorithm portfolio selection. In *Proc. of ECAI'04*, 2004.

[17] T. Hogg. Which search problems are random? In *Proceedings of AAAI98*, pages 438–443, 1998.

[18] T. Hogg, B. A. Huberman, and C. P. Williams. Phase transitions and the search problems. *Artificial Intelligence*, 81(1–2), 1996. Special issue on Phase Transitions and Search Problems.

[19] H. H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for sat. *Artificial Intelligence*, 112:213–232, 1999.

[20] H.H. Hoos. SAT-encodings, search space structure, and local search performance. In *Proceedings of IJCAI99*, pages 988–993, 1999.

[21] H.H. Hoos and T. Stützle. SATLIB: An online resource for research on SAT. In I. Gent, H. van Maaren, and T. Walsh, editors, *SAT2000: Highlights of Satisfiability Research in the Year 2000*, pages 283– 292. IOS Press, 2000.

[22] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[23] S. A. Kauffman. *At home in the universe.* Oxford University Press, New York, 1995.

[24] S. A. Kauffman and W. Macready. Technological evolution and adaptive organizations. *Complexity*, 26(2):26–43, March 1995.

[25] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems. In P. Van Henteryck, editor, *Proceedings of CP02 - Eighth International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science.* Springer, 2002.

[26] W. G. Macready and S. A. Kauffman A. G. Siapas. Criticality and parallelism in combinatorial optimization. *Science*, 271:56–59, January 1996.

[27] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of AAAI97/IAAI97*, pages 321–326, Menlo Park, 1997. AAAI Press.

[28] M. Milano and A. Roli. Boolean networks-based algorithms for the satisfiability problem. In *Electronic Proceedings of the Workshop on Experimental Analysis of Algorithms for Artificial Intelligence*, 1999. AI*IA working group on Knowledge Representation and Reasoning.

[29] M. Milano and A. Roli. Solving the Satisfiability Problem through Boolean Networks. In E. Lamma and P. Mello, editors, *AI*IA99: Advances in Artificial Intelligence*, volume 1792 of *Lecture Notes in Artificial Intelligence*, pages 72–83. Springer, 2000.

[30] D. G. Mitchell, B. Selman, and H. J. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of AAAI92*, pages 459–465. AAAI Press/MIT Press, July 1992.

[31] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137, July 1999.

[32] M.E.J. Newman, S.H. Strogatz, and D.J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64(2):026118–1–026118–17, 2001.

[33] E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham. Understanding random sat: Beyond the clauses-to-variables ratio. In M. Wallace, editor, *Proceedings of CP04 - Tenth International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *Lecture Notes in Computer Science*. Springer, 2004.

[34] A.J. Parkes. Distributed local search, phase transitions, and polylog time. In *Workshop "Stochastic Search Algorithms" at IJCAI01 (http://www.cs.ubc.ca/~hoos/IJCAI-01-WS/stoch.html)*, 2001.

[35] I. Rish and R. Dechter. Resolution versus search: Two strategies for SAT. *J. Automated Reasoning*, 24:225–275, 2000.

[36] A. Roli. Criticality and parallelism in GSAT. *Electronic Notes in Discrete Mathematics*, 9, 2001.

[37] A. Roli. Criticality and parallelism in structured sat instances. In P. Van Henteryck, editor, *Proceedings of CP02 - Eighth International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 714–719. Springer, 2002.

[38] A. Roli. Impact of structure in parallel local search for SAT. In *SAT 2002 – Fifth International Symposium on the Theory and Applications of Satisfiability Testing*, Cincinnati, Ohio, USA, 2002.

[39] A. Roli. Metaheuristics and structure in satisfiability problems. Technical Report DEIS-LIA-03-005, University of Bologna (Italy), May 2003. PhD Thesis - LIA Series no. 66.

[40] A. Roli. Problem structure and search: Empirical results and open questions. In *Proc. of CP-AI-OR'03 – Fifth Int. Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*, Montreal (Canada), 2003.

[41] A. Roli and C. Blum. Critical parallelization of local search for MAX-SAT. In *Proceedings of AI*IA 2001 - 7th Congress of Italian Association of Artificial Intelligence.*, 2001.

[42] A. Roli and F. Zambonelli. Emergence of macro spatial structures in dissipative cellular automata. In *Proc. of ACRI2002: Fifth International Conference on Cellular Automata for Research and Industry*, volume 2493 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2002.

[43] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for local search. In *Proceedings of AAAI94, Seattle/WA, USA*, pages 337–343, 1994.

[44] B. Selman, H. J. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of AAAI92*, pages 440–446, Menlo Park, California, 1992. American Association for Artificial Intelligence, AAAI Press.

[45] B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1–2):17–29, 1996.

[46] A. Strohmaier. Multi-flip networks: Extending symmetric networks to real parallelism. Technical report aida-96-08, FG Intellektik, TU Darmstadt, 1996.

[47] T. Walsh. Search in a small world. In *Proceedings of IJCAI99*, pages 1172–1177, 1999.

[48] T. Walsh. Search on high degree graphs. In *Proceedings of IJCAI01*, 2001.

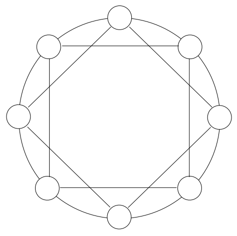


Figure 32: Example of lattice graph. Each node has 4 neighboring nodes.

[49] J.P. Watson, L. Barbulescu, A.E. Howe, and L.D. Whitley. Algorithm Performance and Problem Structure for Flow-Shop Scheduling. In *Proceedings of AAAI99*, 1999.

[50] D.J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness.* Princeton University Press, 1999.

[51] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[52] C. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:72–117, 1994.

[53] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of IJCAI03*, 2003.

[54] M. Yagiura and T.Ibaraki. Analyses on the 2 and 3-flip neighborhoods for the MAXSAT. *Journal of Combinatorial Optimization*, 3:95–114, 1999.

Appendix A

In this section, we describe the procedure used to generate constant-degree k-SAT instances.

The starting graph is a *lattice graph*. Lattice graphs have a very regular topology and every node is connected to a fixed (usually quite small) number of neighbors. Examples of lattice graphs are ring lattices (also called cycles) and hypercubes. For instance, a lattice graph in which every node has four neighbors is depicted in Fig.32. Observe that the graph can be seen as a circular structure with adjunctive links connecting neighbors at distance 2.

Once obtained the graph with the given topology, we have to assign variables to nodes and to generate the clauses of the formula. The first step can be completed very easily by assigning variables in order: variable x_i is assigned to node i, for $i = 1, \ldots, n$. The generation of clauses, i.e., of a formula that can be mapped into the given lattice graph, is a bit more complex. First of all, we remind that the graph associated to a SAT instance, as previously defined, corresponds to a set of SAT instances. Therefore, it is important to define a given structure for the formula. Our choice is to follow the usual experimental settings for random generated SAT instances: 3-SAT formulas with controlled ratio m/n, where n is the number of variables and m the number of clauses.

In the following, we describe the algorithm to generate 3-SAT instances with given ratio m/n on a lattice graph. The generalization of the algorithm to k-SAT instances is straightforward. The high level algorithm is described in Alg.3. The algorithm is structured in two phases. In the first phase, a minimal set of clauses is generated to obtain a formula that can be represented by the given lattice graph. In the second phase, the additional required number of clauses is generated by adding clauses randomly chosen from the first set and by randomly changing the sign of literals.

In the first phase, clauses of three literals are constructed, by taking in turn each variable as a *pivot* and adding two subsequent variables (see Fig.33 and Fig.34). In order to avoid repetitions

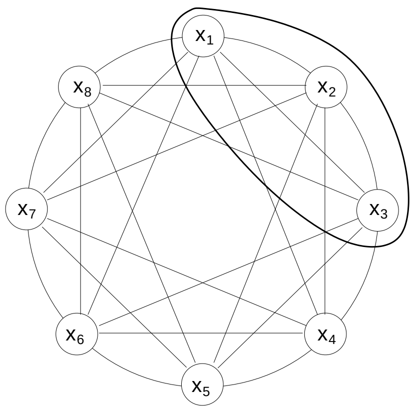


Figure 33: Construction of the first clause involving variable x_1.

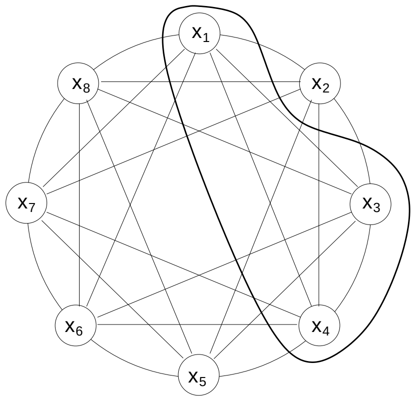


Figure 34: Construction of the second clause involving variable x_1.

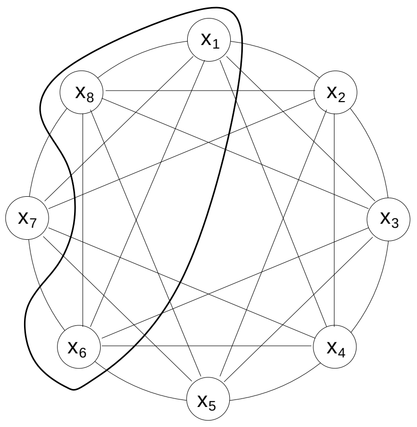


Figure 35: Construction of the third clause involving variable x_1. The pivot is variable x_6.

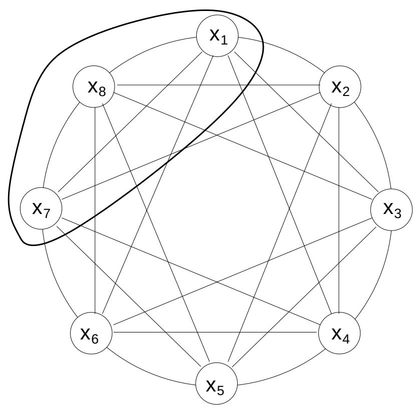


Figure 36: Construction of the fourth clause involving variable x_1. The pivot is variable x_7.

Algorithm 3 Generation of a 3-SAT instance on a lattice graph

```
INPUT: $n$, $m$, $\gamma$ {$\gamma$ is the number of neighbors}
OUTPUT: 3-SAT formula $\Phi = \{C_1, \ldots, C_m\}$ with $n$ variables and $m$ clauses associated to a
lattice graph with $n$ nodes with $\gamma$ neighbors each.

Build a lattice graph $G(n, \gamma)$ (on a circle) with $n$ nodes with $\gamma$ neighbors each;
Assign variables (clockwise) to nodes;
$\Phi \leftarrow \emptyset$
for $i = 1$ to $n - 1$ do
  The neighbors of $x_i$ are $\mathcal{N}^+ = \{x_{i+1}, \ldots, x_{i+\gamma/2}\}$ (mod $n$) and $\mathcal{N}^- = \{x_{i-1}, \ldots, x_{i-\gamma/2}\}$
  (mod $n$);
  for each pair $x_j, x_{j+1}$ in $\mathcal{N}^+$ do
    Construct the clause $C = x_i \vee x_j \vee x_{j+1}$
    Negate each variable in $C$ with probability 0.5;
    $\Phi \leftarrow \Phi \cup C$
  end for
end for
{Now the number of clauses is $|\Phi| = n(\gamma/2 - 1)$ }
while $|\Phi| < m$ do
  repeat
    Pick randomly a clause $C'$ in $\Phi$;
    Negate each variable in $C'$ with probability 0.5;
  until a new clause $C'$ is generated
  $\Phi \leftarrow \Phi \cup C'$
end while
```

of clauses, for every variable x_i only subsequent variables $x_j, j > i$ (modulo n) are considered. Indeed, given the symmetry of the graph, the clauses involving the symmetric part of neighbors will be generated by using those neighbors as pivot (see Fig.35 and Fig.36).

A complete example of a lattice-3-SAT instance with $n = 6$, $m = 12$ and nodes with 4 neighbors is the following:

$\Phi = \{(\neg x_1 \vee x_2 \vee x_3), (x_2 \vee \neg x_3 \vee x_4), (\neg x_3 \vee x_4 \vee x_5), (\neg x_4 \vee \neg x_5 \vee x_6), (x_5 \vee x_6 \vee x_1), (x_6 \vee x_1 \vee x_2), (\neg x_5 \vee x_6 \vee \neg x_1), (x_3 \vee x_4 \vee \neg x_5), (\neg x_5 \vee x_6 \vee x_1), (x_2 \vee x_3 \vee \neg x_4), (x_6 \vee \neg x_1 \vee x_2), (\neg x_2 \vee x_3 \vee x_4)\}$.

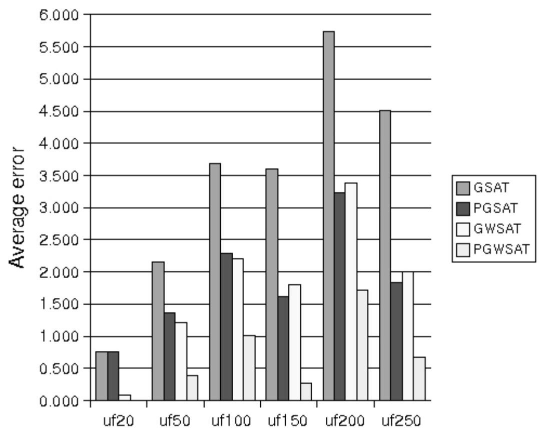


Figure 37: Comparison of the average error (number of unsatisfied clauses) of the four algorithms run on random instances (one for each size). Results are averaged over 1000 trials.

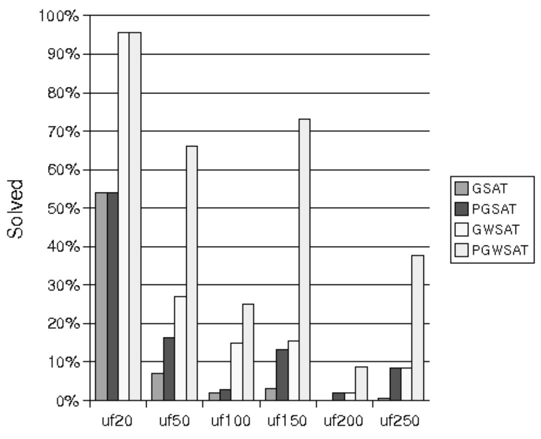


Figure 38: Success rate of the four algorithms run on random instances (one for each size). Results are averaged over 1000 trials.

Appendix B

Concerning criticality and parallelism and noise, we performed preliminary experiments on the combination of PGSAT with GWSAT, which is the combination of GSAT and 'random walk'. With a predefined probability wp, instead of performing a GSAT move, an unsatisfied clause is randomly chosen and a variable within it is selected at random to be flipped. In [43] it is shown that the introduction of noise enables the search to escape from confined areas of the search space and produces a more robust and efficient algorithm than GSAT. Our purpose was to test whether parallel moves and noise conflict or combine their positive effects[18].

The algorithm we implemented, PGWSAT, is the combination of PGSAT and random walk: with probability wp, a repairing move is performed and with probability $1 - wp$, a usual PGSAT move is taken. The optimal value for both wp (noise) and τ (parallelism) have been tuned as if the strategies were run independently. The results reported in Fig.39 show that PGWSAT behaves in the same way as PGSAT[19]. Furthermore, very interesting is the performance of PGWSAT, which outperforms GSAT, PGSAT and GWSAT both in terms of average solution quality and success rate, as shown in Fig.37 and Fig.38.

[18]I thank Holger Hoos for suggesting me to explore in this direction.

[19]At least, for random instances.

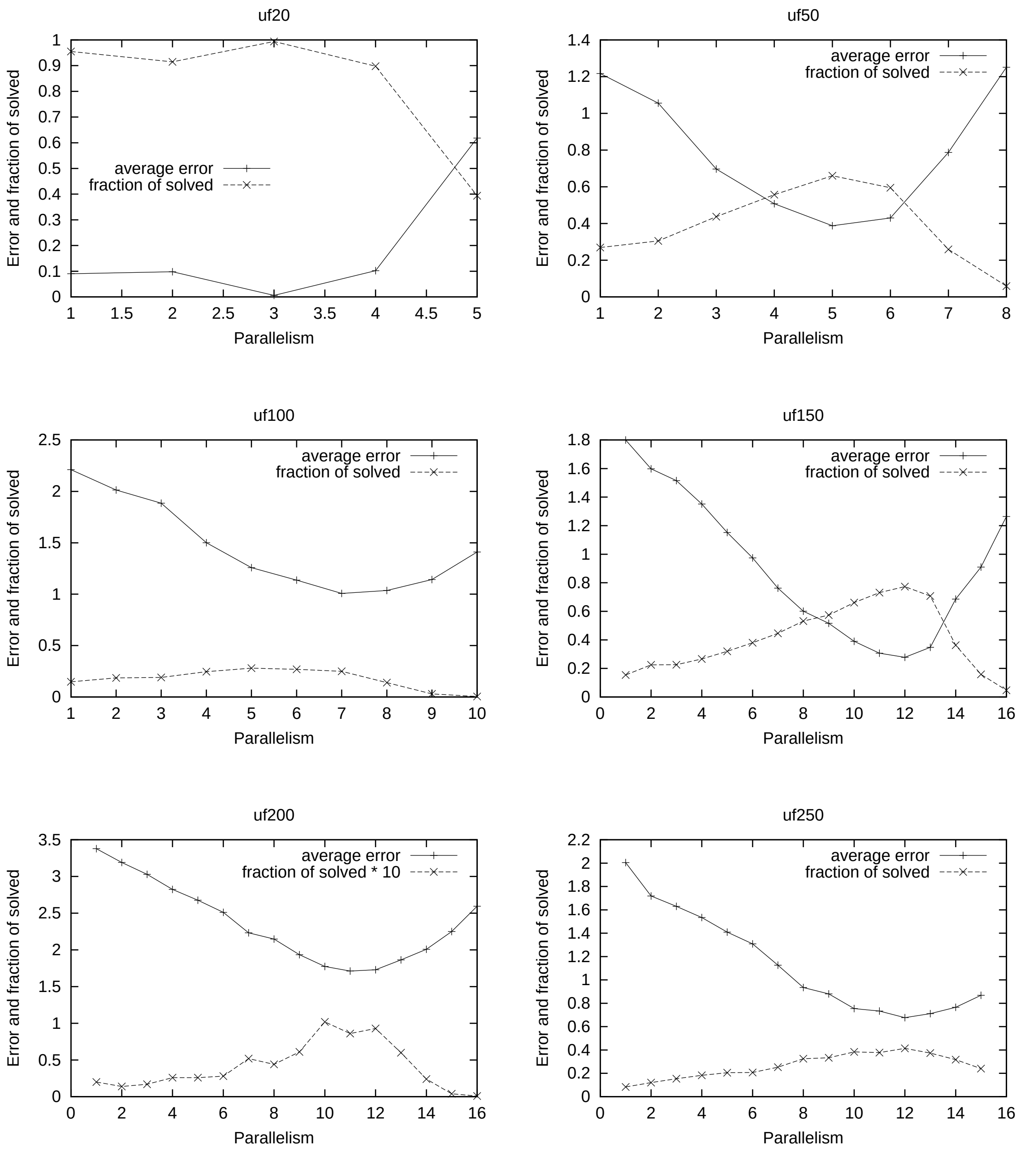


Figure 39: Average error (number of unsatisfied clauses) and fraction of solved instances (rescaled when necessary) vs. parallelism (τ) for random 3-SAT instances with $20, 50, 100, 150, 200$ and 250 variables. Results are averaged over 1000 trials.