Effect of Initial Assignment on Local Search Performance for Max Sat

Daniel Berend
Departments of Mathematics and of Computer Science,
Ben-Gurion University, Beer Sheva 84105, Israel
berend@cs.bgu.ac.il

Yochai Twitto
Department of Computer Science, Ben-Gurion University, Beer Sheva 84105, Israel
twittoy@cs.bgu.ac.il

Abstract

In this paper, we explore the correlation between the quality of initial assignments provided to local search heuristics and that of the corresponding final assignments. We restrict our attention to the Max r-Sat problem and to one of the leading local search heuristics – Configuration Checking Local Search (CCLS). We use a tailored version of the Method of Conditional Expectations (MOCE) to generate initial assignments of diverse quality.

We show that the correlation in question is significant and long-lasting. Namely, even when we delve deeper into the local search, we are still in the shadow of the initial assignment. Thus, under practical time constraints, the quality of the initial assignment is crucial to the performance of local search heuristics.

To demonstrate our point, we improve CCLS by combining it with MOCE. Instead of starting CCLS from random initial assignments, we start it from excellent initial assignments, provided by MOCE. Indeed, it turns out that this kind of initialization provides a significant improvement of this state-of-the-art solver. This improvement becomes more and more significant as the instance grows.

2012 ACM Subject Classification Theory of computation → Theory of randomized search heuristics; Theory of computation → Stochastic approximation

Keywords and phrases Combinatorial Optimization, Maximum Satisfiability, Local Search, Probabilistic Algorithms

Digital Object Identifier 10.4230/LIPIcs.SEA.2020.8

Funding This research was partially supported by the Milken Families Foundation Chair in Mathematics and by the Lynne and William Frankel Center for Computer Science.

Acknowledgements We thank Shaowei Cai for providing us access to the original authors' implementation of the CCLS solver used in Max Sat Evaluation 2016, and André Abramé for providing us access to the Abramé-Habet benchmark used (partially) in that evaluation. We also thank Gregory Gutin and Shahar Golan for their helpful comments on this paper.

1 Introduction

In the Maximum Satisfiability (Max Sat) problem [31], we are given a sequence of clauses over some boolean variables. Each clause is a disjunction of literals over different variables. A literal is either a variable or its negation. We seek a truth (`true`/`false`) assignment for the variables, maximizing the number of satisfied (made `true`) clauses.

In the Max r-Sat problem, each clause is restricted to consist of at most r literals. Here we restrict our attention to instances with clauses consisting of exactly r literals each (sometimes called Max Er-Sat). We denote by n the number of variables and by m the number of clauses. The density of the instance is $\alpha = m/n$. As is customary in the literature, we focus on the case where r and α are constant.

As Max r-Sat (for $r \geq 2$) is NP-hard [5], it cannot be exactly solved in polynomial time (unless $P = NP$), and one must resort to approximation algorithms and heuristics. Numerous methods have been suggested for solving Max r-Sat, e.g. [20, 44, 43, 32, 14, 37, 3, 19, 29], and an annual competition of solvers has been held since 2006 [4]. Satisfiability related questions attracted a lot of attention from the scientific community. As an example, one may consider the well-studied satisfiability threshold question for random instances [15, 25, 2, 35, 16, 21]. For a comprehensive overview of the whole domain of satisfiability we refer to [7, 22].

1.1 Local search

Local search heuristics [30] explore the assignment space. They usually start from a randomly generated assignment, and traverse the search space by flipping variables, usually one at a time. The leading solver Configuration Checking Local Search (CCLS) [32] follows this scheme and flips variables until some predefined number of flips is executed or the allotted time has been used up. Of course, if a satisfying assignment has been found, the execution is stopped as well.

CCLS performs two types of flips: random ones, with some predefined probability p, and greedy ones, with probability $1 - p$. Random flips just flip a randomly selected variable from a randomly selected unsatisfied clause. Greedy flips are ones that flip the seemingly best possible variable among all the variables whose configuration has been changed [32] and who satisfy at least one currently unsatisfied clause. This variable is the one with the maximum score out of those variables, i.e., the one whose flipping will lead to the maximum number of satisfied clauses. Ties are broken randomly.

Recent works, related to local search, configuration checking, CCLS, and algorithms of the same spirit, include [38, 11, 10, 33, 34, 1, 8, 9, 12, 42, 36, 45, 13, 24].

1.2 The Method of Conditional Expectations

The simple randomized approximation algorithm, which assigns to each variable a uniformly random truth value, independently of all other variables, satisfies $1-1/2^r$ of all clauses on the average. Furthermore, this simple algorithm can be easily derandomized using the Method of Conditional Expectations (MOCE) [23, 47], yielding an assignment that is guaranteed to satisfy at least this proportion of clauses.

In a sense, this method is optimal for Max 3-Sat, as no polynomial-time algorithm for Max 3-Sat can achieve a performance ratio exceeding $7/8$ unless $P = NP$ [28]. We note that, typically, this method yields assignments that are much better than this worst-case bound.

MOCE iteratively constructs an assignment by going over the variables in some (arbitrary) order. At each iteration, it sets the seemingly better truth value to the currently considered variable. This is done by comparing the expected number of satisfied clauses under each of the two possible truth values it may set to the current variable.

For a given truth value, the expected number of satisfied clauses is the sum of three quantities. The first is the number of clauses already satisfied by the values assigned to the previously considered variables. The second is the additional number of clauses satisfied by the assignment of the given truth value to the current variable. The third is the expected number of clauses that will be satisfied by a random assignment to all currently unassigned variables. The truth value, for which the sum in question is larger, is the one selected for the current variable. Ties are broken arbitrarily or randomly. The whole process is repeated until all variables are assigned.

Recent theoretical and empirical works related to MOCE, and algorithms of the same spirit, include [17, 39, 41, 40, 18].

1.3 Overview

In Section 2, we explore the correlation between the quality of the initial assignments provided to local search heuristics and the quality of the final assignments resulting from them. We restrict our attention to CCLS, and use a tailored version of MOCE to generate initial assignments of diverse quality, to accommodate the exploration of the correlation.

We show that there is a strong long-lasting correlation between the quality of the initial assignment, from which the local search heuristic starts, and the quality of the final assignment provided by it. This implies that, even when we delve deeper into the local search, we are still in the shadow of the initial assignment. Thus, the quality of the initial assignment is crucial under practical time constraints. The observed correlation decays slower for denser instances, and faster for sparser ones. We show that the correlation is statistically significant, and estimate the impact of the improvement in the quality of the initial assignment on the quality of the final assignment.

In Section 3, we demonstrate our point by improving CCLS. Instead of starting CCLS from a random initial assignment, we start it from excellent initial assignments, provided by MOCE. This kind of initialization provides a significant improvement of this state-of-the-art solver. Moreover, the improvement becomes more and more significant as the instance grows. It has been noticed in other problems, such as TSP and QAP, that local search heuristics yield excellent results when started from initial solutions selected greedily with respect to expectation [27, 26]. A summary and conclusions are presented in Section 4.

2 Correlation between the quality of initial and final assignments

In this section, we explore the correlation between the number of clauses unsatisfied by an initial assignment and the number of those unsatisfied by the corresponding final assignment, where the transition is by CCLS. We explore the ongoing correlation during the execution as well. We have chosen CCLS for its excellent performance; a local search heuristic of lower quality may well be expected to yield an even stronger correlation.

To generate initial assignments of diverse quality, we manipulate MOCE by adding to it a parameter that allows us to invert its decision regarding the truth value for the current variable. This parameter, to which we refer as the inversion probability, is the probability to assign to a variable the truth value opposite to the one chosen by MOCE. Namely, for a given inversion probability $0 \leq p \leq 1$, at each step, we assign to the current variable the truth value chosen by MOCE with probability $1 - p$, and the opposite truth value with probability p. Thus, for $p = 0$ the algorithm is simply MOCE, while for $p = 1$ it is "anti-MOCE". We refer to this tailored algorithm as PMOCE.

We have generated a benchmark, consisting of 5 families of instances of Max 3-Sat. Each of the families consists of 150 instances over 100,000 variables. The densities of the 5 families are 5, 7, 9, 12, 15. The instances in each family were generated uniformly at random as follows. The clauses of an instance were generated independently of each other. Each of the clauses was generated by selecting 3 distinct variables uniformly at random, and then negating each of them with probability 1/2, independently.

2.1 End-to-end correlation

In the following, we describe what we have done in the experiment for each family. For each instance in the family, we executed PMOCE with 51 inversion probabilities, ranging from 0 to 1 in steps of 0.02. Thus, we obtained 51 initial assignments with presumed diverse

Table 1 End-to-end correlation coefficients and regression slopes.

	correlation coefficient			regression slope	
density	**mean**	**std**	***p*-value**	**mean**	**std**
5	0.52	0.11	$1.7 \cdot 10^{-3}$	$0.5 \cdot 10^{-3}$	$0.1 \cdot 10^{-3}$
7	0.74	0.06	$3.6 \cdot 10^{-7}$	$1.5 \cdot 10^{-3}$	$0.2 \cdot 10^{-3}$
9	0.79	0.12	$2.1 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	$0.5 \cdot 10^{-3}$
12	0.73	0.17	$1.2 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$1.0 \cdot 10^{-3}$
15	0.83	0.08	$1.1 \cdot 10^{-5}$	$3.4 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$

quality. From each of these initial assignments, we started a local search using CCLS, and thus obtained 51 final assignments. By the end of the 51 executions, we had 51 pairs of numbers. Each pair consisted of the number of clauses unsatisfied by the initial assignment generated by PMOCE, and the number of unsatisfied clauses at the end of the search done by CCLS. The cutoff time of CCLS was set to 30 minutes, measured in CPU time.

For each instance, we calculated the correlation coefficient over the corresponding 51 pairs. After going over the whole family, we had 150 correlation coefficients – one for each instance. Then, we calculated the mean and standard deviation of these 150 values of correlation coefficients.

For each of the correlation coefficients, we also calculated the p-value. The p-value is the probability that we would have found this correlation, or a higher one, if the correlation coefficient was in fact zero (null hypothesis). If this probability is lower than the conventional 5% (i.e., the p-value is less than 0.05), the correlation coefficient is considered statistically significant. For each family, we calculated the average p-value over the 150 correlation coefficients as a measure of the statistical significance of the results.

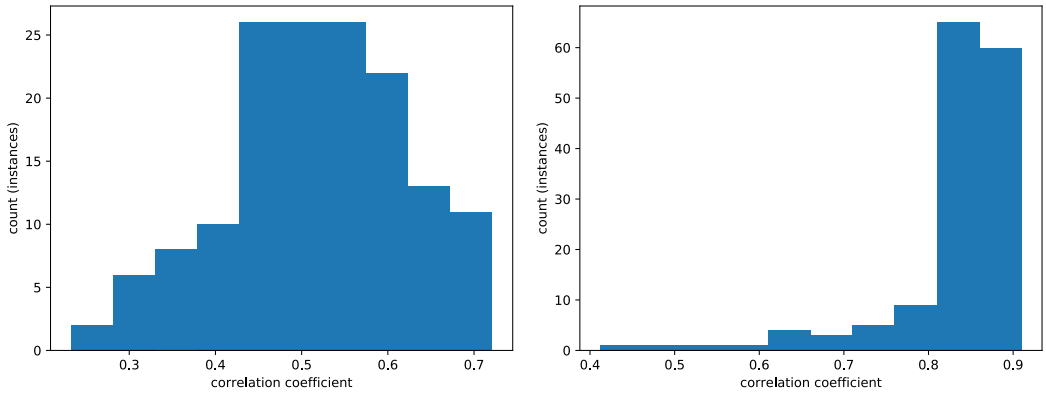
To measure the impact of the improvement of the quality of an initial assignment on the quality of the corresponding final assignment, we applied regression analysis. Specifically, we calculated the regression line of each of the instances of a family, and took its slope as a measure of the strength of the impact. We took the average of these 150 slopes as a measure of the strength of this impact in a given family.

The results are provided in Table 1. Each line summarizes the results of one family. For example, the first line summarizes the results of the family with density 5. In this family, instances are of 100,000 variables and 500,000 clauses. The mean correlation coefficient measured (over 150 random instances) was 0.52, with a standard deviation of 0.11. The mean p-value was $1.7 \cdot 10^{-3}$, and the mean and standard deviation of the regression slope were $0.5 \cdot 10^{-3}$ and $0.1 \cdot 10^{-3}$, respectively.

Figure 1 depicts histograms of the 150 end-to-end correlation coefficients of the family of density 5 (Figure 1a) and for the family of density 15 (Figure 1b).

The results show a strong positive correlation between the quality of the initial and final assignment for all densities. The correlation is stronger for denser families. The p-value is lower by far than the conventional 0.05, which indicates that the correlation coefficients obtained in the experiments are statistically very significant.

While the correlation is strong, the regression slope suggests that a large improvement in the initial assignment yields only a small improvement in the final assignment. As CCLS eventually converges to the optimal solution, there is little room for improvement by the end of its execution, so that this regression slope makes sense. Moreover, it is to be expected that the slope becomes even smaller as one runs CCLS longer.



(a) Family of density 5. **(b)** Family of density 15.

Figure 1 Histograms of end-to-end correlation coefficients.

Note that, after 30 minutes of execution, CCLS is way beyond its rapid improvement stage. In fact, it is deep in its convergence stage and shows relatively minor improvements as time goes by. This validates the correlation observed as meaningful.

Figure 2 depicts the number of unsatisfied clauses as a function of the number of flips made, for an arbitrary (but representative) instance from the family of density 15. The graph shows this number for inversion probability of 0 (MOCE) and 1 (anti-MOCE). We see that CCLS enters its convergence stage quite early in the execution.

We also emphasize the two phases seen in the graphs. The first phase is the rapid improvements phase. In this phase, the number of unsatisfied clauses is decreasing rapidly. This phase ends after about 100,000 flips. The second phase, which we call the convergence phase, continues from there onward. In this phase, the improvements are rarer and smaller.

2.2 Ongoing correlation

Besides the end-to-end correlation, we explored the ongoing correlation during the experiment. To this end, for each initial assignment, we recorded the minimum number of unsatisfied clauses found so far, not only at the end of the execution, but also after every 1000 flips made by CCLS. Then we calculated the correlation coefficient between the number of clauses unsatisfied by the initial assignment and the number of unsatisfied clauses recorded at each 1000 flips snapshot.

The number of flips made during the execution is very different for different families. In a denser instance, a flip takes longer, so that less flips are made. Even for instances of the same family, the number of flips varies. We provide statistics only up to the minimal number of flips made, over all instances in the family.

Figure 3 depicts the decay in the correlation as a function of time, where time is measured in number of flips made from the beginning of the local search. It seems that the number of flips is the natural time scale to measure the correlation decay. While the graphs are noisy, the trend is clear – the correlation gradually decays as a function of the number of flips made, and it does so slower for denser families. Moreover, as the density grows larger, the differences in the decay seem to be smaller and the graphs are almost overlapping.

Figure 3a shows the full results. It provides the graphs of correlation decay of all the families. In the figure, one may observe that the number of flips made in denser families is much smaller than the number of flips made in sparser ones. For example, the minimal

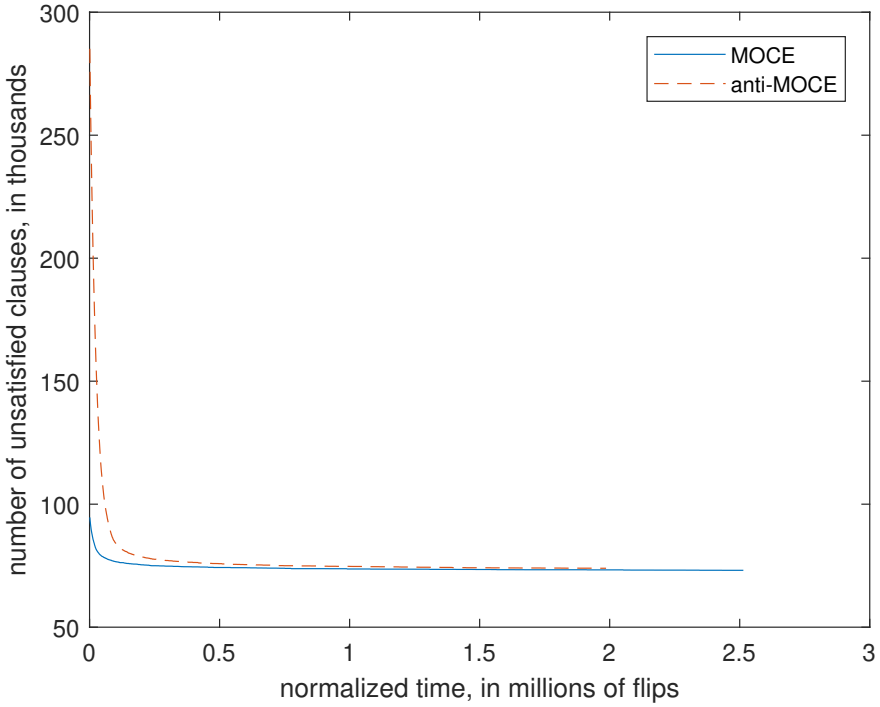


Figure 2 Number of unsatisfied clauses as function of the number of flips.

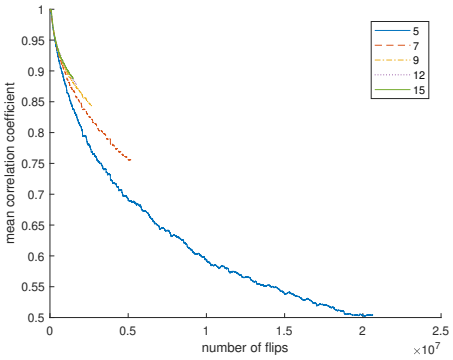
number of flips over all instances and inversion probabilities for the family of density 5 was about 20,600,000, while for the family of density 15 it was about 1,500,000. The reason is that, in denser families, each variable appears in a larger number of clauses, and a flip makes a larger number of variables available for selection subsequently. Thus, at each step, CCLS has to deal with a larger pool of candidates for flipping, which consumes more time per flip.

Figure 3b depicts the same graphs, but only up to about 1,500,000 flips, which is the place where the graph of the family of density 15 ends. In this figure, we see clearly the faster decay of the correlation in sparser families, as well as the smaller differences between the decay in denser families.

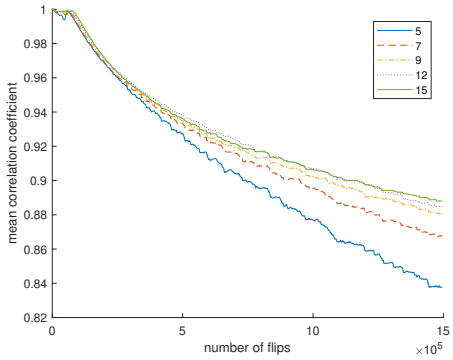
Figure 3c zooms in on the first 150,000 flips. During this stage, we observe a phenomenon of phase transition in the decay of the correlation. The empirical results suggest two phases of decay. The first phase starts at the beginning and ends after about 60,000-80,000 flips. In this phase, the correlation decays very slowly. This phase is characterized by a rapid decrease in the number of unsatisfied clauses, and is aligned with the rapid decrease shown in Figure 2.

The second phase is from about 60,000-80,000 flips onward. This phase is characterized by a faster decay in the correlation. It is aligned with the convergence stage of CCLS, shown in Figure 2, in which the number of unsatisfied clauses is decreasing slowly over time.

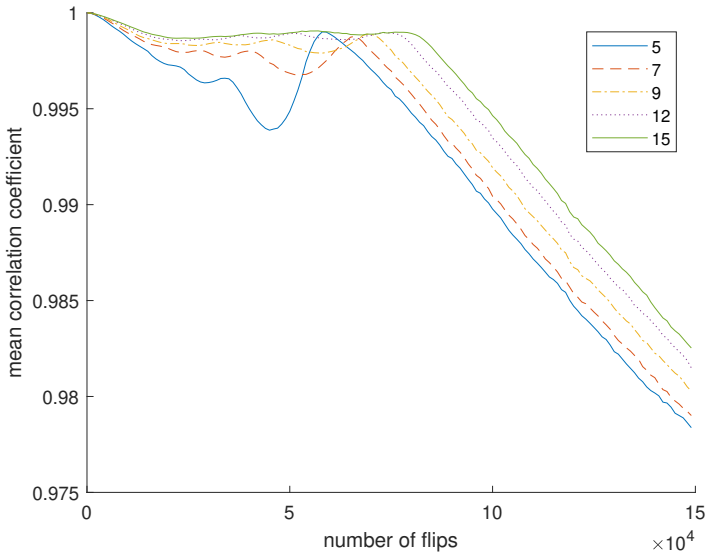
The position of the phase transition around 60,000-80,000 flips may be explained by the fact that the initial assignment provided by MOCE is expected to be at a distance of about 50,000 flips from an optimal solution. So the first 50,000 flips are significant. But, as about 30% of the flips of CCLS are random, and not all the flips are useful in general, this area stretches further to about 60,000-80,000 flips.



(a) Full graphs.



(b) Up to 1,500,000 flips.



(c) First 150,000 flips.

Figure 3 Ongoing correlation decay as a function of the number of flips.

During the first phase, the initial assignment is very important in determining the correlation. In all the executions, the decrease in the number of satisfied clauses is rapid and considerable at this phase. Thus, the different executions maintain their relative positions, which leads to a very slow decrease in the correlation. Afterward, the correlation decays at about the same speed, as can be seen in Figure 3c.

2.3 Experimentation information

The experiments described in this section were executed on a Sun Grid Engine (SGE) [46] managed cluster of 31 identical IBM m4 servers with Intel Xeon E5-2620@2.0GHz processors. Each of the servers consists of 24 computation cores and 64GB of working memory. Thus, we had 744 computation cores and 1984GB of working memory at hand.

We limited each of the jobs submitted to the cluster to use up to 3GB of working memory. Provided the load on the cluster, we managed to achieve a parallelization of about 300 times, thus reducing the experiments overall sequential time of approximately 2.18 years to around 2.66 days of parallel execution.

3 Improving CCLS

In this section, we study the improvement obtained by letting CCLS start its execution from good initial assignments, versus starting it from a random assignment (as done originally). Specifically, the good initial assignments we use are assignments provided by MOCE. We refer to the algorithm that starts from the assignment provided by MOCE as MOCE-CCLS. To emphasize the fact that the original CCLS algorithm starts from a random assignment, we will call it RAND-CCLS.

We first conducted experiments on several families of random instances. The families have been selected in a systematic way, so as to reveal trends in the performance, and connect it to the parameters of the family. Afterward, we conducted experiments on some public benchmarks. We show that MOCE-CCLS scales much better than RAND-CCLS. In particular, as the instance size grows, so does the performance improvement provided by MOCE-CCLS over RAND-CCLS.

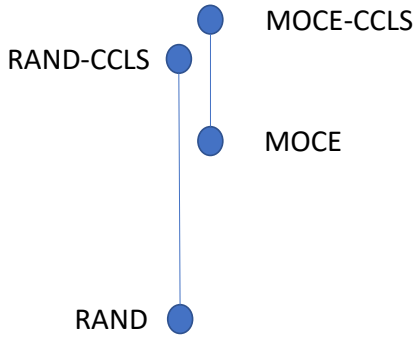


Figure 4 Performance diagram. Higher is better.

In Figure 4, we summarize qualitatively what we have observed in the experiments. The higher the algorithm appears in the diagram, the better it is. Inspecting the diagram, one can see that MOCE-CCLS performs much better than MOCE, which in turn shows performance very far away from the baseline reference RAND. MOCE-CCLS performs better than RAND-CCLS as well. The last statement holds significantly for large instances, while for small and medium instances MOCE-CCLS maintains or slightly improves the performance of RAND-CCLS.

3.1 Comparative performance on structured benchmarks

In this section we focus on random instances, for which the clauses are of length 3, the number of variables ranges from 10,000 to 1,000,000, and the density from 3 to 9. Such ranges allow us to systematically study the performance of the algorithms at hand on diverse families. For each family, we selected 100 instances uniformly at random, in the same way elaborated in Section 2.

For Max r-Sat, the reference baseline RAND unsatisfies $m/2^r$ clauses on average, with an approximate standard deviation of $\sqrt{m(1-1/2^r)/2^r}$ clauses [6]. For convenience, the (theoretical) average number of clauses unsatisfied by RAND for the families we studied (namely, $m/8$), is provided in Table 2. The rows correspond to the various numbers of variables, n, and the columns to the various densities, α.

Table 2 The number of clauses unsatisfied by RAND and MOCE.

n \ α	3		5		7		9	
	RAND	MOCE	RAND	MOCE	RAND	MOCE	RAND	MOCE
10000	3750	412	6250	1500	8750	2889	11250	4442
50000	18750	2078	31250	7459	43750	14409	56250	22209
100000	37500	4149	62500	14944	87500	28861	112500	44436
500000	187500	20790	312500	74702	437500	144296	562500	222074
1000000	375000	41559	625000	149383	875000	288572	1125000	444174
% unsat	12.5%	1.4%	12.5%	3%	12.5%	4.1%	12.5%	4.9%

Table 3 The improvement by executing CCLS after RAND and MOCE.

n \ α	3		5		7		9	
	(R−RC)/R	(M−MC)/M	(R−RC)/R	(M−MC)/M	(R−RC)/R	(M−MC)/M	(R−RC)/R	(M−MC)/M
10000	100.0%	100.0%	96.0%	83.6%	85.5%	56.2%	77.4%	42.9%
50000	100.0%	100.0%	95.5%	81.2%	84.8%	54.1%	76.7%	41.2%
100000	100.0%	100.0%	95.1%	79.9%	84.3%	52.9%	76.2%	40.2%
500000	100.0%	100.0%	90.4%	69.4%	77.1%	42.4%	68.3%	31.3%
1000000	80.6%	100.0%	48.7%	49.6%	37.4%	27.1%	31.6%	18.3%

Table 2 also presents the average number of clauses unsatisfied by MOCE. It turns out that this number scales linearly with the number of clauses, and thus can be described as a proportion of the number of clauses, for any fixed density. The proportion of clauses unsatisfied by MOCE, out of all clauses, was 1.4%, 3%, 4.1%, and 4.9% for the densities 3, 5, 7, and 9, respectively. For each family, the percentage of clauses unsatisfied by each of the algorithms is provided in the last line of the table.

MOCE is a linear time algorithm, and is extremely fast in practice. In fact, its execution time is but a few seconds for the larger instances we studied, and less than a second for the small and medium size instances.

Although MOCE returns excellent solutions, it benefits a lot from supplementing it with a highly performing local search. In fact, executing the local search part of CCLS (which we simply call CCLS), starting from the solution returned by MOCE, we obtained a significant improvement. Namely, the number of unsatisfied clauses is significantly reduced at the local search stage.

This improvement is summarized in Table 3. In this table, the columns shortly named "(M−MC)/M" present the relative improvement of MOCE-CCLS over MOCE. This relative improvement is the difference between the number of clauses unsatisfied by MOCE and the number of those unsatisfied by MOCE-CCLS, divided by the number of clauses unsatisfied by MOCE. In the table, we also present the improvement of supplementing RAND with CCLS (which is simply the standard version of CCLS), under the columns named "(R−RC)/R".

It is worth mentioning that this significant improvement comes with a caveat – a significant increase in the execution time. In fact, the results shown in Table 3 are based on 30 minutes executions of CCLS, after the initial solution (by either RAND or MOCE) has been obtained in just a few seconds. One more caveat is due to the fact that, as the instances grow larger, this improvement decreases. As the instance grows larger, the number of flips CCLS can perform during the allotted time decreases, and with it decreases the obtained improvement as well.

Table 4 MOCE-CCLS vs. RAND-CCLS.

n \ α	3			5		
	RC	MC	% improve	RC	MC	% improve
10000	0	0	NaN	248	246	0.81%
50000	0	0	NaN	1417	1403	0.99%
100000	0	0	NaN	3038	3002	1.18%
500000	0	0	NaN	29976	22894	23.63%
1000000	72642	0	100.00%	320674	75260	76.53%
n \ α	7			9		
	RC	MC	% improve	RC	MC	% improve
10000	1265	1264	0.08%	2546	2537	0.35%
50000	6647	6617	0.45%	13122	13052	0.53%
100000	13717	13588	0.94%	26770	26554	0.81%
500000	99976	83163	16.82%	178234	152512	14.43%
1000000	548044	210440	61.60%	769640	363037	52.83%

We conclude this section by comparing MOCE-CCLS and RAND-CCLS head to head. The comparison is provided in Table 4 (which is wrapped for readability). For each density, we provide the number of clauses unsatisfied by RAND-CCLS, the number of clauses unsatisfied by MOCE-CCLS, and the relative improvement of MOCE-CCLS over RAND-CCLS. The latter number is the difference between the number of clauses unsatisfied by RAND-CCLS and the number of those unsatisfied by MOCE-CCLS, divided by the number of clauses unsatisfied by RAND-CCLS.

The results demonstrate our point regarding the importance of the initial solution. Even after 30 minutes of local search, and using the excellent local search heuristics CCLS, the initialization with MOCE instead of RAND yields better solutions. Moreover, MOCE-CCLS proved to be much more scalable than RAND-CCLS. As the instance grows larger, the improvement of MOCE-CCLS over RAND-CCLS becomes more significant. Whereas, for small instances, MOCE-CCLS improves RAND-CCLS by less than 1%, for large instances the improvement exceeds 50%.

In view of the above, when using a local search algorithm for Max Sat, one should strive to start the search from very good assignments. This holds as long as it is not too much time consuming to attain such assignments.

3.1.1 Experimentation information

The experiments described in this section were carried out on the same infrastructure as in Section 2.3. Here as well, we limited each of the jobs submitted to the cluster to use up to 3GB of working memory. Provided the load on the cluster, we managed to achieve a parallelization of about 100 times, thus reducing the experiment overall sequential time of approximately 4.11 months to around 1.25 days of parallel execution.

3.2 Comparative performance on public benchmarks

The random instances of Maximum Satisfiability Evaluation 2016 were tailored mainly for complete solvers. Thus, they are very small and less adequate for evaluation of local search heuristics. Indeed, most of the solvers participating in that evaluation found solutions with the same number of unsatisfied clauses most of the time; the ranking was only according to the time they consumed to reach their best solutions. In our comparison of RAND-CCLS and MOCE-CCLS, the situation was no different.

In the following, we consider three additional benchmarks in the same spirit as the 2016 Evaluation – but larger ones. As we wanted to keep the exact same blend of instances, we created the new benchmarks by blowing up the original ones. We enlarged the number of variables and that of clauses in each instance, while keeping the density the same as in the evaluation. We created three expanded benchmarks by enlarging the original one by factors of 10, 100, and 1000.

We compared MOCE-CCLS and RAND-CCLS on the enlarged instances using the Instance Won measure. This measure is the one used in the Max Sat Evaluation [4] held in 2016, from which we took the original instances. We ran each of the two competitors on each of the instances for a few minutes (CPU time). For each instance, the winner is the competitor that provides the smaller number of unsatisfied clauses. Ties are broken by the time it took each competitor to arrive at its best solution. The overall winner is the heuristic that wins more instances.

While RAND-CCLS wins on the competition instances, it is enough to blow up the instances tenfold to have MOCE-CCLS achieve an overall draw. When scaling the instances by a factor of 100, MOCE-CCLS wins decisively, and when scaling by a factor of 1000, it beats RAND-CCLS by a knockout. In fact, MOCE-CCLS wins on the expanded benchmarks in terms of the number of unsatisfied clauses, and not merely by time. Namely, MOCE-CCLS provides solutions with a strictly smaller number of unsatisfied clauses.

Finally, we note that MOCE alone is not enough. It is the value from the combined solver MOCE-CCLS that leads to the extra satisfied clauses. Moreover, CCLS provides a significant improvement to the excellent initial solutions of MOCE. Thus, the state-of-the-art performance of MOCE-CCLS is attributed to both its ingredients: MOCE and CCLS.

4 Summary and conclusions

In this paper, we have explored the correlation between the quality of initial assignments provided to local search heuristics and that of the corresponding final assignments. We have shown that this correlation is significant and long-lasting. Thus, under practical time constraints, the quality of the initial assignment is crucial to the performance of local search heuristics.

We demonstrated our point by improving the state-of-the-art solver CCLS, by combining it with MOCE. Instead of starting CCLS from random initial assignments, we started it from excellent initial assignments, provided by MOCE. The combined MOCE-CCLS solver provided a significant improvement over CCLS. Moreover, MOCE-CCLS proved to be much more scalable. Namely, it handles larger instances better, and shows superior performance on them.

Given the above, we recommend MOCE-CCLS over RAND-CCLS. Furthermore, we recommend starting CCLS from solutions even better than those provided by MOCE, as long as such may be obtained in linear time or slightly longer (say, by a logarithmic factor).

References

1 André Abramé, Djamal Habet, and Donia Toumi. Improving configuration checking for satisfiable random k-sat instances. *Annals of Mathematics and Artificial Intelligence*, 79(1-3):5–24, 2017.

2 Dimitris Achlioptas and Yuval Peres. The threshold for random k-sat is $2^k \log 2 - o(k)$. *Journal of the American Mathematical Society*, 17(4):947–973, 2004.

3 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Sat-based maxsat algorithms. *Artificial Intelligence*, 196:77–105, 2013.

4 Josep Argelich, Chu Min Li, Felip Manyà, and Jordi Planes. Maxsat evaluations. URL: `http://www.maxsat.udl.cat/`.

5 Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, 2 edition, 2003.

6 Daniel Berend and Yochai Twitto. The normalized autocorrelation length of random max r-sat converges in probability to $(1 - 1/2^r)/r$. In *The 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016)*, pages 60–76. Springer, 2016.

7 Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of Satisfiability*, volume 185. IOS press, 2009.

8 Noureddine Bouhmala. A variable neighborhood walksat-based algorithm for max-sat problems. *The Scientific World Journal*, 2014, 2014.

9 Shaowei Cai, Zhong Jie, and Kaile Su. An effective variable selection heuristic in sls for weighted max-2-sat. *Journal of Heuristics*, 21(3):433–456, 2015.

10 Shaowei Cai, Chuan Luo, John Thornton, and Kaile Su. Tailoring local search for partial maxsat. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 2623–2629. AAAI Press, 2014.

11 Shaowei Cai and Kaile Su. Local search with configuration checking for sat. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 59–66. IEEE, 2011.

12 Shaowei Cai and Kaile Su. Local search for boolean satisfiability with configuration checking and subscore. *Artificial Intelligence*, 204:75–98, 2013.

13 Byungki Cha, Kazuo Iwama, Yahiko Kambayashi, and Shuichi Miyazaki. Local search algorithms for partial maxsat. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, page 263–268. AAAI Press, 1997.

14 Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse max-sat and max-k-csp. In *Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT 2015)*, pages 33–45. Springer, 2015.

15 Vašek Chvátal and Bruce Reed. Mick gets some (the odds are on his side) [satisfiability]. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

16 Amin Coja-Oghlan. The asymptotic k-sat threshold. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 804–813. ACM, 2014.

17 Don Coppersmith, David Gamarnik, MohammadTaghi Hajiaghayi, and Gregory B Sorkin. Random max sat, random max cut, and their phase transitions. *Random Structures & Algorithms*, 24(4):502–545, 2004.

18 Kevin P Costello, Asaf Shapira, and Prasad Tetali. Randomized greedy: new variants of some classic approximation algorithms. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 647–655. Society for Industrial and Applied Mathematics, 2011.

19 Jessica Davies and Fahiem Bacchus. Solving maxsat by solving a sequence of simpler sat instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming*, pages 225–239. Springer, 2011.

20 Pieter-Tjerk de Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

21 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 59–68, 2015.

22 Ding-Zhu Du, Jun Gu, and Panos M. Pardalos, editors. *Satisfiability Problem: Theory and Applications, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, March 11-13, 1996*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. DIMACS/AMS, 1997.

23 Paul Erdős and John L Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.

24 Paola Festa, Panos M Pardalos, Leonidas S Pitsoulis, and Mauricio GC Resende. Grasp with path-relinking for the weighted maximum satisfiability problem. In *International Workshop on Experimental and Efficient Algorithms*, pages 367–379. Springer, 2005.

25 Ehud Friedgut and Jean Bourgain. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.

26 Gregory Gutin and Abraham P Punnen. *The Traveling Salesman Problem and Its Variations*, volume 12. Springer Science & Business Media, 2006.

27 Gregory Gutin and Anders Yeo. Polynomial approximation algorithms for the tsp and the qap with a factorial domination number. *Discrete Applied Mathematics*, 119(1-2):107–116, 2002.

28 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

29 Federico Heras, Javier Larrosa, and Albert Oliveras. Minimaxsat: An efficient weighted max-sat solver. *Journal of Artificial Intelligence Research (JAIR)*, 31:1–32, 2008.

30 Holger H Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. Elsevier, 2004.

31 Chu Min Li and Felip Manyà. *MaxSAT, hard and soft constraints*, volume 185, pages 613–631. IOS Press, 2009.

32 Cheng Luo, Shanyong Cai, Wenchuan Wu, Zhong Jie, and Kuan-Wu Su. Ccls: An efficient local search algorithm for weighted maximum satisfiability. *IEEE Transactions on Computers*, 64(7):1830–1843, 2014.

33 Chuan Luo, Shaowei Cai, Kaile Su, and Wei Wu. Clause states based configuration checking in local search for satisfiability. *IEEE transactions on Cybernetics*, 45(5):1028–1041, 2014.

34 Chuan Luo, Shaowei Cai, Wei Wu, and Kaile Su. Focused random walk with configuration checking and break minimum for satisfiability. In *International Conference on Principles and Practice of Constraint Programming*, pages 481–496. Springer, 2013.

35 Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-sat from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

36 Patrick Mills and Edward Tsang. Guided local search for solving sat and weighted max-sat problems. *Journal of Automated Reasoning*, 24(1-2):205–223, 2000.

37 Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided maxsat resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2717–2723. AAAI Press, 2014.

38 Denis Pankratov and Allan Borodin. On the relative merits of simple local search methods for the max-sat problem. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT 2010)*, SAT'10, page 223–236, Berlin, Heidelberg, 2010. Springer-Verlag.

39 Matthias Poloczek. Bounds on greedy algorithms for max sat. In *Proceedings of the 19th European Conference on Algorithms*, ESA'11, page 37–48, Berlin, Heidelberg, 2011. Springer-Verlag.

40 Matthias Poloczek, Georg Schnitger, David P Williamson, and Anke Van Zuylen. Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. *SIAM Journal on Computing*, 46(3):1029–1061, 2017.

41 Matthias Poloczek and David P Williamson. An experimental evaluation of fast approximation algorithms for the maximum satisfiability problem. In *International Symposium on Experimental Algorithms*, pages 246–261. Springer, 2016.

42 Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, page 337–343. American Association for Artificial Intelligence, 1994.

43 Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1996.

44 Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.

45 Kevin Smyth, Holger H Hoos, and Thomas Stützle. Iterated robust tabu search for max-sat. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 129–144. Springer, 2003.

46 Sun grid engine (sge) quickstart. URL: `http://star.mit.edu/cluster/docs/0.93.3/guides/sge.html`.

47 Mihalis Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17(3):475–502, 1994.