SCIENCE CHINA
Information Sciences

• LETTER •

November 2019, Vol. 62 219106:1–219106:3
https://doi.org/10.1007/s11432-018-9786-2

A Kernighan-Lin inspired algorithm for MAX-SAT

Noureddine BOUHMALA

Department of Maritime Innovation and Technology, Institute of Micro-Systems, University of South-Eastern Norway, Borre 3184, Norway

Received 6 May 2018/Revised 15 August 2018/Accepted 15 October 2018/Published online 24 September 2019

Citation Bouhmala N. A Kernighan-Lin inspired algorithm for MAX-SAT. Sci China Inf Sci, 2019, 62(11): 219106, https://doi.org/10.1007/s11432-018-9786-2

Dear editor,

In general, an instance of the Boolean satisfiability (SAT) problem is defined by a set of Boolean variables $V = \{v_1, \ldots, v_N\}$, a set of clauses $C = \{c_1, c_2, \ldots, c_M\}$, and a Boolean formula $\Phi : \{0,1\}^N \rightarrow \{0,1\}$. The formula Φ is in a conjunctive normal form (CNF) if it is a conjunction of clauses. Each clause, in turn, is a disjunction of literals, and a literal is a variable or its negation. The task is to determine whether there exists an assignment of values to the variables under which Φ evaluates to true. Such an assignment, if it exists, is known as a satisfying assignment for Φ, and Φ is called satisfiable. Otherwise, Φ is said to be unsatisfiable. Because we have two choices for each of the n Boolean variables, the size of the search space S becomes $|S| = 2^N$, i.e., the size of the search space grows exponentially with the number of variables. Most SAT solvers use a CNF representation of the formula Φ. For example, $P \vee Q$ is a clause containing two literals, namely P and Q. The clause $P \vee Q$ is satisfied if either P or Q is true. The formula Φ with N variables and M clauses creates a landscape in the space Boolean formula $\{0,1\}^N \times \{0,1,\ldots,M\}$. The 2^N truth assignments correspond to the points $\{0,1\}^N$, and the height of each point is a number between 0 and M corresponding to the number of clauses in Φ that are violated. MAX-SAT is the optimized variant of SAT, where the task is to determine whether there exists an assignment of truth values of the variables that satisfies the minimum number k of unsatisfied clauses. The quality of the solution is evaluated in terms of the number of unsatisfied clauses or the total weight of the unsatisfied clauses in the case where weights are associated with clauses using an evaluation function $f(A)$ to be minimized; this function is defined as

$$f(A) = \sum_{i=1}^{M} w_{c_i} |\{c | \neg \mathrm{sat}(A, c_i) \wedge c_i \in \Phi\}, \qquad (1)$$

where A is an assignment, $\neg\mathrm{sat}(A, c_i)$ denotes that the clause c_i is unsatisfied, and w_{c_i} represents the weight associated with c_i. For unweighted problem instances, w_c is equal to 1 for all clauses.

The algorithm. The popular Kernighan-Lin (KL) [1] algorithm, which belongs to the variable depth search class, is adapted and used to solve MAX-SAT problems. Its advantage compared to simple local search methods is that it allows moves that worsen the quality of the solution as long as the overall result is an improvement. KL still remains the most successful approaches used for the graph partitioning problem.

The KL method proposed in this study is based on a two-phase strategy. The first phase applies KL until a local optimum solution is reached, whereas the second phase performs a kick to allow KL to proceed with the search each time it is stuck at a local optimum solution. KL starts by labeling all variables as tabu-inactive (false). KL goes through several phases. During each phase, a k-change is built sequentially by modifying the truth value of one variable at a time by performing a flip. A flip consists of changing the truth value

Email: nb@usn.no

of a variable v_i from val to $1 - \text{val}$. KL identifies the variable having the highest score. The highest score of a variable v_i is the one that results in a maximal decrease or minimal increase in the number of unsatisfied clauses if v_i is flipped. If there are several variables with the same maximal score, one of them is randomly selected according to a uniform distribution. Let v_i be the variable to be flipped. The score of v_i is the difference between the number of unsatisfied clauses before performing the flip minus the number of unsatisfied clauses after the flip is performed. Therefore, a positive number indicates a decrease in the number of unsatisfied clauses, whereas a negative number corresponds to an increase in the number of false clauses. If a variable has already been chosen, it can no longer be considered and it is labeled as tabu active (true). One phase of the algorithm terminates when all variables are set to tabu active (true). The algorithm identifies the subset of variables having the highest cumulative score. The cumulative score is the sum of each of the scores made by altering the value of one variable. If such a subset exists, the solution is updated by substituting all the variables in the subset with their new truth values. The variables not belonging to the subset will retain their old values. The best assignment is updated whenever the current assignment has a better cost. The assignment with the lowest cost found during the current phase is used as the starting solution for the next phase. However, if such a subset fails to exist, the algorithm proceeds with the second phase referred to as the kick phase. The kick phase chooses between performing a random flip or a kick depending on whether the result of the division (random() mod 2) is an even or odd number. A random flip consists of choosing a random variable and changing its truth value. A kick is based on a large r, where an r flip is a random number drawn between 1 and N.

Experimental results. Table 1 compares KL to the iterated robust tabu search algorithm [2] using UBCSAT (version 1.1) [3], an implementation and experimentation environment for stochastic local search algorithms for SAT and MAX-SAT solvers. The number of runs was set to 20 for both algorithms.

For each algorithm, the average solution quality (Q) and average time (T) are shown. The first 14 benchmark instances are real problems that arise in difficult integer factorization problems translated into the MAX-SAT format.

One observes that IRoTS cannot find the best solution for all the difp instances.

KL has a better value compared to IRoTS for all difp instances. The difference in quality ranges between 24% and 83%. The quality superiority of KL comes at the expense of a higher running time (approximately 2–3 times slower). The second benchmark instance set is of unweighted MAX-SAT problems. KL delivers solutions of equal quality compared to IRoTS at the expense of a higher running time (up to 8 times slower). The same observation applies to the final weighted instances.

Note that the majority of the running time for KL is spent finding the appropriate kick to lead KL to a better local optimum, which explains the variability in the running time between the instances.

Conclusion. Herein, an algorithm based on the Kernighan-Lin heuristic for MAX-SAT was introduced. This algorithm is based on a two-stage strategy. Each time KL reaches a local optimum configuration, the algorithm chooses between performing a simple random move or a kick. Experimental comparisons demonstrated the superiority of KL compared to IRoTS for real problems.

KL manages to produce similar results compared to IRoTS for both unweighted and weighted test cases.

Currently, a faster implementation of KL using a multilevel scheme is being developed to improve

Table 1 Performance comparisons

	Inst	KL		IRoTS	
		Q	T	Q	T
Real	difp-19-0	7	89.5	21	66.5
	difp-19-1	5	60.1	19	47.8
	difp19-3	7	105.1	23	68.1
	difp-19-99	16	112.8	26	52.7
	difp20-0	7	109.5	18	41.1
	difp20-1	8	103.1	26	73.2
	difp20-2	20	98.1	26	62.1
	difp20-3	6	117.3	34	60.3
	difp20-99	18	114.7	26	52.8
	difp-21-0	8	102.7	31	48.2
	difp21-1	9	85.1	34	42.7
	difp21-2	20	117	33	78.3
	difp21-3	22	119.4	34	69.3
	difp21-99	8	79.2	33	60.3
Unweighted	v100c1200	865	2.31	865	0.13
	v100c1300	933	6.76	933	0.16
	v100c1400	1018	6.31	1018	0.15
	v100c1500	1186	3.36	1186	0.19
	v120c1400	959	4.06	959	0.16
	v120c1500	1119	7.25	1119	0.14
	v120c1600	1150	0.85	1150	0.16
	v140c1500	1073	0.86	1073	0.17
	v140c1600	1143	12.14	1143	0.17
Weighted	v70c700	110	1.60	110	0.16
	v70c800	132	2.54	132	0.16
	v70c900	157	2.76	157	0.15
	v70c1000	214	6.51	214	0.18

its running time as well as its solution quality.

References

1 Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J, 1970, 49: 291–307

2 Smyth K, Hoos H H, Stützle T. Iterated robust tabu search for MAX-SAT. In: Proceedings of the 16th Canadian Society for Computational Studies of Intelligence Conference on Advances in Artificial Intelligence, Halifax, 2003. 129–144

3 Tompkins A D, Hoos H H. UBCSAT: an implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In: Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing, Vancouver, 2004. 306–320