A polynomial time algorithm for SAT

Ortho Flint

Abstract

The deterministic polynomial time algorithm that determines satisfiability of 3-SAT can be generalized for SAT.

1 Introduction

The proof for the deterministic polynomial time algorithm that determines satisfiability of 3-SAT found at: polynomial3sat.org, can be easily modified to prove a generalized version of the algorithm for SAT.

2 A polynomial time algorithm for SAT

Let K_q be a complete graph on q vertices, $q \geq 1$. Observe that every definition in the paper at polynomial3sat.org can be modified by simply replacing the term: edge-sequence with K_q-sequence and the term: vertex-sequence with K_1-sequence. Most importantly, the three lemmas and the theorem in the paper can also be modified by the very same replacements. Thus, we have a proof for a generalized version of the original algorithm for 3-SAT. We note that all the definitions, rules, etc., must be used in the generalized version of the algorithm. For example, K_q-sequences (and the K_1-sequences which are also constructed), must be LCR and K-rule compliant. And in the proof for 3-SAT the concept of literal triples for 3-SAT, would be the concept of literal $(q+1)$-tuples for $(q+1)$-SAT. Provided below are the modified versions of definition 2.2 and 2.11 respectively, from the paper. Note that a K_2-sequence is an edge-sequence.

Definition 2.1. *A K_q-sequence is an ordered sequence with elements* 1 *and* 0*. The ordering is an ordering of the clauses, with indexing: C_1, C_2, C_3, ... , C_c where a position in the sequence having a* 1 *or* 0 *entry, is associated with the literal corresponding to that position. A K_q-sequence I, for a K_q with endpoints labelled x_1, x_2, x_3, ... , x_q, where no x_i and its negation appear, the literals associated with the endpoints, is denoted by $I_{x_1,\ldots,x_q}$. The endpoints must always be from different clauses. We call the positions in $I_{x_1,\ldots,x_q}$ that correspond to a clause C_i the cell C_i. The cells containing the endpoints, x_1, x_2, x_3, ... , x_q, have only one entry that is* 1 *in the positions associated to x_1, x_2, x_3, ... , x_q. When a K_q-sequence is constructed, a given position in $I_{x_1,\ldots,x_q}$ is* 1 *if the associated literal is not a negation of the literals x_1, x_2, x_3, ... , x_q. The initial construction of $I_{x_1,\ldots,x_q}$ is subject to certain rules defined in* 2.8 *and* 2.9 *of the paper, which may produce more zero entries. Lastly, removing one or more cells from $I_{x_1,\ldots,x_q}$ is again a (sub) K_q-sequence, denoted by $I_{x_1,\ldots,x_q}{}^*$, if the cells containing the endpoints for $I_{x_1,\ldots,x_q}$ remain.*

Definition 2.2. *An S-set is a collection of K_q-sequences whose endpoints are from q clauses, where the literals associated with the endpoints are such that no x_i and its negation appear. The number of constructed K_q-sequences to be an S-set is the product of the sizes of the q clauses less any non K_q-sequence. ie. A non K_q-sequence is a K_q-sequence containing at least one x_i and its negation associated with the endpoints.*

As clause sizes increase, Comparing any two S-sets is more work in general, and the number of S-sets to Compare also increases. In other words, suppose c clauses are considered, then there are $\binom{c}{q}$ S-sets, thus the number of S-set comparisons for a **run** is $\binom{\binom{c}{q}}{2}$. For example, a 4-SAT $\mathcal{G}$, with c clauses requires S-sets containing K_3-sequences. So, an S-set could have as many as 4^3 K_3-sequences and the number of S-sets constructed for $\mathcal{G}$ would be $\binom{c}{3}$. Note well that only K_q-sequences and K_1-sequences for $(q+1)$-SAT are constructed. The latter is for our mechanism to determine possible unsatisfiability of the given SAT. If the given SAT is satisfiable, then a **round** one can be completed, where every K_q-sequence from a collection of equivalent S-sets $\mathcal{X}$, is such that a literal with a 1 entry in $I_{x_1,\ldots,x_q}$ belongs to at least one K_C with x_1, x_2, x_3, ... , x_q.

For 2-SAT

It can now be seen by the generalization that a 2-SAT $\mathcal{G}$, with c clauses, is processed by Comparing just the K_1-sequences between the c S-sets, one for each clause. Clearly, 1-SAT is trivial and it's always handled by pre-processing. ie. either one solution or no solution.

3 Final comments

It is the case that Comparing for SAT becomes more expensive as clause size increases relative to just converting to 3-SAT. However, the natural generalization of the algorithm for 3-SAT, could be exploited for efficiency purposes, by extracting information at chosen costs, for Comparing a SAT's corresponding 3-SAT. Below, is a scheme for converting SAT to 3-SAT.

Given a collection of clauses for some SAT, let $k \geq 4$ be the size of a clause C_i. Then the number of clauses of size 3 that will replace C_i when converting the SAT to a 3-SAT, is $k-2$. There is no need to replace clauses of size 2 or 3 from the given SAT.

If $C_i = (1, 2, 3, 4, 5)$ say, then it's replaced with $(5-2)$ clauses of the form: $(1, 2, x)$, $(-x, y, 3)$ and $(-y, 4, 5)$ where the *connectors*: x, $-x$, y and $-y$ must be singletons wrt. all the clauses constructed for the 3-SAT. For another example, let $C_i = (1, 2, 3, 4, 5, 6)$. Then it's replaced with $(6-2)$ clauses of the form: $(1, 2, x)$, $(-x, y, 3)$, $(-y, z, 4)$ and $(-z, 5, 6)$ where again the *connectors*: x, $-x$, y, $-y$, z and $-z$ must be singletons wrt. all the clauses constructed for the 3-SAT. So in general, if $C_i = (1, 2, 3, \ldots, r)$, then it's replaced with $(r-2)$ clauses of the form: $(1, 2, l_1)$, $(-l_1, l_2, 3)$, $(-l_2, l_3, 4)$, $\ldots$, $(-l_{r-4}, l_{r-3}, r-2)$, $(-l_{r-3}, r-1, r)$, where the *connectors*: l_1, $-l_1$, l_2, $-l_2$, l_3, $-l_3$, $\ldots$, l_{r-3} and $-l_{r-3}$, must be singletons wrt. all the clauses constructed for the 3-SAT.

In conclusion, equivalency is determined by Comparing S-sets for each SAT by: K_1-sequences for 2-SAT, K_2-sequences for 3-SAT, K_3-sequences for 4-SAT, $\ldots$, K_q-sequences for $(q+1)$-SAT. It should be clear by this generalization, that there is nothing special or unique about 3-SAT conceptually.