

Lower bounds for splittings by linear combinations^{*}

Dmitry Itsykson^{**} and Dmitry Sokolov

Steklov Institute of Mathematics at St.Petersburg
27, Fontanka, St.Petersburg, Russia, 191023

Splitting is the one of the most frequent methods for exact algorithms for NP-hard problems. It considers several cases and recursively executes on each of that cases. For the CNF satisfiability problem the classical splitting algorithms are so called DPLL algorithms (by authors Davis, Putnam, Logemann and Loveland) in which splitting cases are values of a variable. A very natural extension of such algorithms is a splitting by a value of some formula. In this paper we consider an extension of DPLL that allows splitting by linear combinations of variables over $\mathbb{F}_2$. There is a polynomial time algorithm that check whether a system of linear equations has a solution and whether a system of linear equations contradicts a clause. Thus the running time of an algorithm that solves CNF-SAT using splitting by linear combinations (in the contrast to splitting by arbitrary functions) is at most the size of its splitting tree up to a polynomial factor.

Formulas that encode unsatisfiable systems of linear equations are hard for resolution and hence for DPLL [Tseitin 1968]. Systems of linear equations are also hard satisfiable examples for myopic and drunken DPLL algorithms [Aleknovich et al. 2005], [Itsykson 2010]. Hard examples for myopic DPLL with a cut heuristic are also based on linear systems [Itsykson, Sokolov 2011]. We show that a splitting by linear combinations helps to solve explicitly encoded linear systems over $\mathbb{F}_2$ in polynomial time.

For every CNF formula ϕ we denote by $\phi^\oplus$ a CNF formula obtained from ϕ by substituting $x_1 \oplus x_2$ for each variable x . Urquhart shows that for unsatisfiable ϕ the running time of any DPLL algorithm on $\phi^\oplus$ is at least $2^{d(\phi)}$, where $d(\phi)$ is the minimal depth of the recursion tree of DPLL algorithms running on the input ϕ [Urquhart 2011]. Urquhart also gives an example of Pebbling contradictions $Peb(G_n)$ such that $d(Peb(G_n)) = \Omega(n/\log n)$ and there is a DPLL algorithm that solves $Peb(G_n)$ in $O(n)$ steps. Thus $Peb^\oplus(G_n)$ is one more example that is hard for DPLL algorithms but easy for DPLL with splitting by linear combinations.

The recent algorithm by Seto and Tamaki [Seto, Tamaki, 2013] solves satisfiability of formulas over full binary basis using a splitting by linear combination of variables. The similar idea was used by Demenkov and Kulikov in the simplified lower bound $3n - o(n)$ for circuit complexity over full binary basis [Demenkov,

^{*} The full version of the paper is accepted to MFCS 2014. Research is partially supported by the RFBR grant 14-01-00545, by the President's grant MK-2813.2014.1 and by the Government of the Russia (grant 14.Z50.31.0030).

^{**} dmitrits@pdmi.ras.ru

[Kulikov 2011]. The common idea of [Seto, Tamaki 2013] and [Demenkov, Kulikov 2011] is that a restricting a circuit with a linear equation may significantly reduce the size of the circuit.

Our results. We prove an exponential lower bound on the size of a splitting tree by linear combinations for 2-fold Tseitin formulas that can be obtained from ordinary Tseitin formulas by substituting every variable by the conjunction of two new variables. The plan of the proof is following: let for every unsatisfiable formula ϕ a search problem $Search_\phi$ be the problem of finding falsified clause given a variable assignment. We prove that it is possible to transform a splitting tree T into a randomized communication protocol for the problem $Search_\phi$ of depth $O(\log |T| \log \log |T|)$ if some variables are known by Alice and other variables are known by Bob. And finally, we note that a lower bound on the randomized communication complexity of the problem $Search_{TS_{G,c}^2}$ for a 2-fold Tseitin formula $TS_{G,c}^2$ follows from [Kalyanasundaram, Schintger 1992] and [Beame et al. 2007].

We also give an elementary proof of the lower bound $2^{\frac{n-1}{2}}$ on the size of linear splitting trees of formulas PHP_n^m that encode the pigeonhole principle.

It is well known that the behavior of DPLL algorithms on unsatisfiable formulas corresponds to tree-like resolution proofs. We consider the extension of the resolution proof system that operates with disjunctions of $\mathbb{F}_2$ -linear equalities. A system Res-Lin contains the weakening rule and the resolution rule. We also consider a system Sem-Lin that is a semantic version of Res-Lin; Sem-Lin contains semantic implication rule with two premises instead of the resolution rule. We prove that this two systems are polynomially equivalent and they are implication complete. We also show that tree-like versions of Res-Lin and Sem-Lin are equivalent to splitting by linear combinations; the latter implies that our lower bounds hold for tree-like Res-Lin and Sem-Lin. Raz and Tzameret studied a system $R(lin)$ that operates with disjunctions of linear equalities with *integer coefficients* [Raz, Tzameret 2008]. It is possible to p-simulate Res-Lin in $R(lin)$ but the existence of the simulation in the other direction is an open problem.

Further research. The main open problem is to prove a superpolynomial lower bound in the DAG-like Res-Lin. One of the ways to prove a lower bound is to simulate the Res-Lin system with another system for which a superpolynomial lower bound is known. It is impossible to simulate Res-Lin with Res(k) (that extends Resolution and operate with k-DNF instead of clauses) and in PCR (Polynomial Calculus + Resolution) over field with $char \neq 2$ because there are known exponential lower bounds in Res(k) and PCR for formulas based on systems of linear equations [Razborov 2003]. It is interesting whether it is possible to simulate Res-Lin with Polynomial Calculus (or PCR) over $\mathbb{F}_2$ or with the system $R^0(lin)$ that is a subsystem of $R(lin)$ with known exponential lower bounds based on the interpolation.

Another open problem is to prove lower bounds for splitting by linear combinations on satisfiable formulas, for example, for algorithms that arbitrary choose a linear combination for splitting and randomly choose a value to investigate first.