Electronic Colloquium on Computational Complexity, Report No. 24 (2014)

0-1 Integer Linear Programming with a Linear Number of Constraints

Russell Impagliazzo* Shachar Lovett† Ramamohan Paturi*
Stefan Schneider*

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404, USA
E-Mail: {russell, slovett, paturi, stschnei}@cs.ucsd.edu

January 2014

We give an exact algorithm for the 0-1 Integer Linear Programming problem with a linear number of constraints that improves over exhaustive search by an exponential factor. Specifically, our algorithm runs in time $2^{(1-\text{poly}(1/c))n}$ where n is the number of variables and cn is the number of constraints. The key idea for the algorithm is a reduction to the *Vector Domination problem* and a new algorithm for that subproblem.

*This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
†Supported by NSF CAREER award 1350481.

ISSN 1433-8092

1 Introduction

A large number of real world problems from social sciences, economics, logistics and other areas are very naturally expressed as integer programs. Various variants of Integer Programming have been studied, such as bounds on the solution vector, pure or mixed integer programs, and linear, nonlinear or even nonconvex constraints, as well as a number of other restrictions on the constraints. Most forms of Integer Programming are **NP**-hard, with some variants in **P** (such as linear, totally unimodular constraints) and the variant with nonconvex constraints and unbounded solution vector is even undecidable. For a survey on the large body of work for solving various variants of Integer Programming exactly or approximately, we refer to the survey by Genova and Guliashki [3].

In this paper we concentrate on the special case of 0-1 Integer Linear Programming (ILP). Given n Boolean variables and m linear constraints, the problem is to find an assignment of either 0 or 1 to the variables such that all constraints are satisfied. For this special case, we omit the objective function to be optimized and only consider the problem of deciding if a set of constraints is feasible. Since any objective value can be at most exponentially large in the input size, binary search can reduce the optimization problem to the feasibility problem with only polynomial overhead.

The problem can trivially be solved in time $O(2^n \text{poly}(n, m))$ using exhaustive search. On the other hand, an algorithm to solve the problem in time $O\left(2^{(1-s)n}\right)$ for $s > 0$ when m is superlinear in n would contradict the Strong Exponential Time Hypothesis (**SETH**) [4], which says that for every $s > 0$ there is a k such that k-SAT cannot be solved in time $O\left(2^{(1-s)n}\right)$. CNFSAT on m clauses is a special case of 0-1 ILP on m constraints and the Sparsification Lemma [5] allows us to reduce k-SAT to CNFSAT on $c(k)n$ many clauses. Short of proving or disproving **SETH**, we can then ask the following question: Given a 0-1 integer linear program on n variables and a linear number of constraints, is it possible to decide feasibility faster than exhaustive search? We answer this question affirmatively. In particular, we prove the following.

Theorem 1.1. *Given a* 0-1 *integer program on n variables and $m = cn$ constraints, it is possible to decide the feasibility of the program in time $2^{(1-s(c))n}$ where $s(c) = \Omega\left(poly\left(\frac{1}{c}\right)\right)$.*

For the special case of CNF satisfiability, Schuler [7] gives an algorithm that improves over exhaustive search by an exponential factor if the number of clauses is linear. With savings (the $s(c)$ in above theorem) that are inverse polylogarithmic in c, as opposed to polynomial, Schuler's algorithm runs considerably faster.

Recently, Williams [8] gave an algorithm for 0-1 ILP that improves over exhaustive search even for a polynomial number of constraints. The algorithm runs in time $2^{(1-s)n}$. For $s = \frac{1}{\text{polylog}(m)}$. Since s is subconstant, even for linear m, Williams' result is not directly comparable to our result. Also note that a subconstant s does not contradict **SETH**.

Our result is a follow-up to earlier work of a subset of the authors [6], where we considered the more general class of depth two threshold circuits. Depth two threshold circuits generalize the problem from a conjunction of linear constraints to a threshold function of linear constraints. However, the algorithm for depth two threshold circuits we gave only improves over exhaustive search by an exponential factor if the number of nonzero coefficients (across all constraints) is linear in the number of inputs. Furthermore, the savings of our algorithm for depth two threshold circuit was exponential in $\frac{1}{c}$.

The key idea of our algorithm is to reduce the problem to the *Vector Domination problem*, the problem of finding a pair of vectors such that one vector dominates the other in every coordinate. As such, the main idea stays the same as in our previous work. The main technical contribution of the current paper is an algorithm for the Vector Domination problem that improves over the trivial algorithm for vectors of dimension $O(\log N)$ (where N is the number of vectors), whereas earlier algorithms, such as the one used for our previous result, only worked for dimensions up to $\delta \log N$, where δ is a sufficiently small number. Whereas algorithms for small dimensions have been discussed before, in particular Bentley [1] and Chan [2], we believe that we give the first algorithm improving over exhaustive search for all dimensions possible without refuting **SETH**.

2 ILP and the Vector Domination problem

In this section we give a reduction from 0-1 Integer Linear Programming to the *Vector Domination* problem.

In all definitions below, $u \geq v$ for two vectors u and v of the same dimensions is used to denote an element-wise comparison.

Definition 2.1. Given a matrix $M \in \mathbb{R}^{n\times m}$ and a vector $r \in \mathbb{R}^m$, the 0-1 *Integer Linear Programming* problem on n variables and m constraints is to find a vector $x \in \{0,1\}^n$ such that $Mx \geq r$.

Definition 2.2. Given two sets of d-dimensional real vectors A and B, the *Vector Domination Problem* is the problem of finding two vectors $u \in A$ and $v \in B$ such that $u \geq v$.

Both problems have a trivial exhaustive search algorithm. For 0-1 ILP, that algorithm runs in time $O\left(2^n \mathrm{poly}(n,m)\right)$, while the trivial algorithm for the Vector Domination problem runs in time $O(|A||B|d)$.

We can reduce 0-1 ILP to the Vector Domination problem, which shows that if the Vector Domination problem allows for an algorithm faster than exhaustive search, then so does 0-1 ILP. The reduction was introduced by Williams [9] for the special case of Boolean vectors (he called the problem the Cooperative Subset Query problem).

Lemma 2.1. *Suppose there is an algorithm for the Vector Domination problem for $|A| = |B| = N$ and $d = 2c\log N$ running in time $O\left(N^{2-s(c)}\right)$. Then a 0-1 Integer Linear program on n variables and $m = cn$ constraints can be solved in time $O\left(2^{(1-s(c)/2)n}\right)$.*

Proof. Let M and r be the matrix and vector respectively of the 0-1 integer linear program. Separate the variable set into two sets S_1 and S_2 of equal size. For every assignment to the variables in S_1 and S_2, we assign a cn-dimensional vector where every dimension corresponds to a constraint. Let α be an assignment to S_1 and let $a \in \mathbb{R}^{cn}$ be the vector with $a_j = \sum_{i\in S_1} M_{i,j}\alpha(x_i)$ and let A be the set of $2^{n/2}$ such vectors. For an assignment β to S_2, let b be the vector with $b_j = r_j - \sum_{xi\in S_2} M_{i,j}\beta(x_i)$ and let B be the set of all such vectors b.

An assignment to all variables corresponds to an assignment to S_1 and an assignment to S_2, and hence to a pair $a \in A$ and $b \in B$. The pair satisfies all inequalities if and only if a dominates b. We have $|A| = |B| = N = 2^{n/2}$ and $d = cn = 2c\log(N)$. Hence the Vector Domination problem, and therefore the 0-1 Integer Linear Program, can be solved in time $O\left(N^{2-s(c)}\right) = O\left(2^{(1-s(c)/2)n}\right)$. □

Note that the reduction can be trivially adapted to any ILP variant where the variables can take values from a constant size set.

3 An Algorithm for the Vector Domination Problem

The algorithm is a divide and conquer algorithm, splitting the two sets A and B into sets A^+,A^-, B^+ and B^- depending on if the first coordinate is larger or smaller than some value a. For our purposes, we choose a as a weighted median that is computed across both sets A and B.

The weighted median of a collection of real numbers with associated real weights is a number such the both the total weight of all numbers smaller than the median and the total weight of all numbers larger than the median are at most half of the total weight. Note that the weighted median can be computed in linear time using minor modifications to the standard algorithms to compute the unweighted median.

Lemma 3.1. *Let $A, B \subseteq \mathbb{R}^d$ with $|A| = |B| = N$ and $d = c \log N$. There is an algorithm for the Vector Domination problem that runs in time $O\left(N^{2-s(c)}\right)$, where $s(c)$ is polynomial in $\frac{1}{c}$.*

Proof. We assume w.l.o.g. that $c \geq 4$. Let $\varepsilon = \frac{1}{c^{15}}$, $\gamma = \frac{1}{15 \log c}$ and $t = \gamma \log N$.

Furthermore, let a be the weighted median of the first coordinates of $A \cup B$, where all numbers from A have weight $|B|$ and all number from B have weight $|A|$. Further let $A^+ \subseteq A$ consist of all vectors where the first coordinate is larger than a and A^- consist of all vectors where the first coordinate is smaller than a. Similarly, split B into two sets B^+ and B^-. For vectors where the first coordinate is exactly a, it is sufficient to split the vectors evenly between the two possible sets, as long as we do not at the same time add vectors to A^- and B^+. This rounding is equivalent to adding a small positive noise to all vectors in A and a small negative noise to all vectors in B.

Now a vector $u \in A$ can only dominate a vector $v \in B$ in one of three cases:

1. $u \in A^+$ and $v \in B^+$
2. $u \in A^-$ and $v \in B^-$
3. $u \in A^+$ and $v \in B^-$

Also, in the third case any vector in A^+ dominates any vector in B^- on the first coordinate, hence our recursive algorithm can recurse on $d - 1$ dimensions.

Since we split at a weighted median where both A and B have the same total weight, we have $\frac{|A^-|}{|A|} = \frac{|B^+|}{|B|} =: \varepsilon'$. We distinguish two cases, a *balanced* case where $\varepsilon' \geq \varepsilon$ and an *unbalanced* case where $\varepsilon' < \varepsilon$. In both the balanced and the unbalanced cases, we recurse on all three subproblems. In the balanced case, we additionally decrement t. Once $t = 0$ we solve the Vector Domination problem on the remaining vectors by exhaustive search.

To bound the runtime of above algorithm, we consider the recursion tree. We first bound the time spent on exhaustive search in leaves where $t = 0$. Each of the N^2 possible pairs of dominating vectors appears in at most one of the subcases. Furthermore, in the balanced case there are at least $\varepsilon^2 N^2$ pairs where one vector is in A^- and the other is in B^+ that are not considered in any subcase. Since there are t balanced cases on the path from the root to any leaf with $t = 0$, the total time spent on exhaustive search is bounded by

$$(1-\varepsilon^2)^t N^2 \leq e^{-\varepsilon^2 \gamma \log N} N^2 = N^{2-\log(e)\varepsilon^2\gamma} = N^{2-\frac{\log(e)}{15c^{30}\log(c)}} = N^{2-\mathrm{poly}(1/c)}$$

For $c = 4$, we get constant savings of 2^{-66}.

To bound the size of the recursion tree we bound the number of possible paths from the root to any leaf. On any path, there are at most d steps that decrease the dimension. Furthermore, there are at most t balanced cases on any path. For the unbalanced case, in both subproblems 1 and 2, where the dimension does not decrease, the number of pairs of vertices decreases by a factor of at most ε, hence this can happen at most $\frac{\log(N^2)}{\log(1/\varepsilon)}$ times along any path. Let $r = t + \frac{\log(N^2)}{\log(1/\varepsilon)} = \left(\gamma + \frac{2}{\log(1/\varepsilon)}\right) \log N = \frac{1}{5 \log(c)} \log(N)$. Taking into account that the time spent on computing the median in every node is linear in $|A|+|B|=O(N)$, the total time is bounded by

$$O(N)\binom{d+r}{r}2^r \le O(N)\left(e\left(1+5c\log(c)\right)\right)^{\frac{1}{5\log(c)}\log N} N^{\frac{1}{5\log(c)}}$$
$$= N^{1+\log(e(1+5c\log(c)))\frac{1}{5\log(c)}+\frac{1}{5\log(c)}}$$

The bound is monotonely decreasing for $c \ge 4$ and at $c = 4$ the bound is $O\left(N^{1.781}\right)$.

The overall runtime is therefore always dominated by the time spent on exhaustive search in the leaves, which gives us the claim immediately. □

The algorithm can be extended for the case where $|A| \neq |B|$. If $|A| = O(|B|)$, above analysis immediately gives us a runtime of $(|A||B|)^{1-s(c)/2}$. If $|A| \gg |B|$, we partition A into $\frac{|A|}{|B|}$ subsets of size $|B|$ each and solve the Vector Domination problem for each subset individually. The whole algorithm then runs in time $|B|^{2-s(c)}\frac{|A|}{|B|} = |A||B|^{1-s(c)}$.

Theorem 1.1 then follows directly from Lemmas 2.1 and 3.1.

4 Conclusions

We give a first algorithm for 0-1 integer programs on n variables and cn constraints that improves over exhaustive search by a factor $2^{\mathrm{poly}(1/c)n}$. The result does generalize to ILP where the variables are constrained by any finite set of values, not just Boolean variables.

Under the Strong Exponential Time Hypothesis, this is qualitatively optimal in the sense that we can only expect exponential improvement over exhaustive search if the number of constraints is linear. However, for the special case of formulas in conjunctive normal form the best algorithms achieve savings that are polylogarithmic in $\frac{1}{c}$ [7]. It is open if we can get the same for 0-1 ILP.

This result is not comparable to our earlier work [6]. While 0-1 ILP is a special case of depth two threshold circuits, our result requires a linear number of constraints instead of a linear number of wires. It is still open if we can find an algorithm for general depth two threshold circuits that works for a linear number of gates (i.e. constraints).

Lastly, there are countless problems that reduce to Integer Programming in a natural way. Some of these applications might benefit from this result. Alternatively, there might be problems that reduce directly to the Vector Domination problem and benefit from the corresponding subroutine.

References

[1] Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.

[2] Timothy M Chan. All-pairs shortest paths with real weights in O (n 3/log n) time. In *Algorithms and Data Structures*, pages 318–324. Springer, 2005.

[3] Krasimira Genova and Vassil Guliashki. Linear integer programming methods and approaches–a survey. *Cybernetics And Information Technologies*, 11(1), 2011.

[4] R. Impagliazzo and R. Paturi. The complexity of k-SAT. *Journal of Computer and Systems Sciences*, 62(2):367–375, March 2001. Preliminary version in *14th Annual IEEE Conference on Computational Complexity*, pages 237–240, 1999.

[5] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 1998. Preliminary version in *39th Annual IEEE Symposium on Foundations of Computer Science*, pp 653-662, 1998.

[6] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *Proceedings of the 54rd Annual IEEE Symposium on Foundations of Computer Science*, 2013.

[7] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *Journal of Algorithms*, 54(1):40–44, 2005.

[8] R. Williams. New algorithms and lower bounds for circuits with linear threshold gates. *ArXiv e-prints*, January 2014.

[9] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348:357–365, 2005.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de