Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 348 (2005) 357–365
Theoretical Computer Science
www.elsevier.com/locate/tcs

A new algorithm for optimal 2-constraint satisfaction and its implications☆

Ryan Williams

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

We present a novel method for exactly solving (in fact, counting solutions to) general constraint satisfaction optimization with at most two variables per constraint (e.g. MAX-2-CSP and MIN-2-CSP), which gives the first exponential improvement over the trivial algorithm. More precisely, the runtime bound is a constant factor improvement in the base of the exponent: the algorithm can count the number of optima in MAX-2-SAT and MAX-CUT instances in $O(m^3 2^{\omega n/3})$ time, where $\omega < 2.376$ is the matrix product exponent over a ring. When the constraints have arbitrary weights, there is a $(1+\varepsilon)$-approximation with roughly the same runtime, modulo polynomial factors. Our construction shows that improvement in the runtime exponent of either *k-clique* solution (even when $k = 3$) or *matrix multiplication over GF(2)* would improve the runtime exponent for solving 2-CSP optimization.

Our approach also yields connections between the complexity of some (polynomial time) high-dimensional search problems and some NP-hard problems. For example, if there are sufficiently faster algorithms for computing the diameter of n points in ℓ_1, then there is an $(2-\varepsilon)^n$ algorithm for MAX-LIN. These results may be construed as either lower bounds on the high-dimensional problems, or hope that better algorithms exist for the corresponding hard problems.

Keywords: Exact algorithms; Constraint satisfaction; MAX-2-SAT; MAX-CUT

1. Introduction

The extent to which NP-hard problems are indeed hard to solve remains largely undetermined. For some problems, it intuitively seems that the best one can do is examine every candidate solution, but this intuition has been shown to fail in many scenarios. The fledgling development of improved exponential algorithms in recent times suggests that for many hard problems, something substantially faster than brute-force search can be done, even in the worst case. However, some fundamental problems have persistently eluded researchers from better algorithms. One popular example in the literature is MAX-2-SAT.

There has been notable theoretical interest in discovering a procedure for MAX-2-SAT running in $O((2-\varepsilon)^n)$ steps on all instances, for some $\varepsilon > 0$. Unlike problems such as Vertex Cover and k-SAT, where analysis of branch-and-bound techniques (or random choice of assignments) sufficed for improving the naïve time bounds (e.g. [23,20,13]), MAX-SAT has been surprisingly difficult to attack. Previous work has only been able to show either a $(2-\varepsilon)^m$ time bound, where m is the number of clauses, or an approximation scheme running in $(2-\varepsilon)^n$ [12,8] (but $\varepsilon \to 0$ as the

☆ Supported by the NSF ALADDIN Center (NSF Grant No. CCR-0122581) and an NSF Graduate Research Fellowship.
E-mail address: ryanw@cs.cmu.edu.

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.023

quality of the approximation goes to 1). Of course, the number of clauses can, in general, be far higher than the number of variables, so the $(2-\varepsilon)^m$ bounds only improve the trivial bound on some instances. The current best worst-case bound for MAX-2-SAT explicitly in terms of m is $\tilde{O}(2^{m/5})$, [1] by [9] (so for $m/n > 5$, this is no better than 2^n). This result followed a line of previous papers on bounds in terms of m [11,19,10,6,4,17]; a similar line has formed for MAX-CUT [28,16]. In terms of partial answers to the problem, [21] showed that when every variable occurs at most three times, MAX-2-SAT remains NP-complete, but has an $O(n3^{n/2})$ algorithm. While definite stepping stones, these results were still distant from an improved exponential algorithm. Consequently, several researchers (e.g. [21,1,8,29]) explicitly proposed a $(2-\varepsilon)^n$ algorithm for MAX-2-SAT (and/or MAX-CUT) as a benchmark open problem in exact algorithms.

1.1. Outline of our approach: split and list

Most exact algorithms for NP-hard problems in the literature involve either a case analysis of a branch-and-bound strategy [9], repeated random choice of assignments [20], or local search [25]. Our design is a major departure from these approaches, resembling earlier algorithms from the 1970s [14,26]. We *split* the set of n variables into k partitions (for $k \geqslant 3$) of (roughly) equal size, and *list* the $2^{n/k}$ variable assignments for each partition. From these $k2^{n/k}$ assignments, we build a graph with weights on its nodes and edges, arguing that a optimum weight k-clique in the graph corresponds to a optimum solution to the original instance. The weights are eliminated using a polynomial reduction, and a fast k-clique algorithm on undirected graphs yields the improvement over 2^n. To get a $(1+\varepsilon)$-approximation when constraints have arbitrary weights, we can adapt results concerning approximate all pairs shortest paths [30] for our purposes.

We will also investigate the possibility of efficient split-and-list algorithms for more general problems such as SAT and MAX-LIN-2 (satisfying a maximum number of linear equations modulo 2). In particular, we will demonstrate some connections between this question and problems in high-dimensional geometry. For example, if a furthest pair out of n d-dimensional points in ℓ_1 norm can be found faster than its known solutions (say, in $O(poly(d) \cdot n^{2-\varepsilon})$ time), then there exists a $(2-\varepsilon)^n$ split-and-list algorithm for MAX-LIN-2.

1.2. Notation

$\mathbb{Z}^+$ and $\mathbb{R}^+$ denote the set of positive integers and the set of positive real numbers, respectively.

Let $V = \{x_1, \ldots, x_n\}$ be a set of variables over (finite) domains $D_1, \ldots, D_n$, respectively. A *k-constraint* on V is defined as a function $c : D_1 \times \cdots \times D_n \to \{0, 1\}$, where c only depends on k variables of V (one might say c is a *k-junta*). For a k-constraint c, define $\mathsf{vars}(c) \subseteq V$ to be this k-set of variables. Partial assignments a to variables of V are given by a sequence of assignments $x_{i_1} := v_2, x_{i_2} := v_2, \ldots, x_{i_k} := v_k$, where $i_j \in [n]$ and $v_{i_j} \in D_{i_j}$. A partial assignment *a satisfies* a constraint c if $\mathsf{vars}(c)$ is a subset of the variables appearing in a, and $c(a) = 1$ (the restriction of c to the variables in a evaluates to 1, on the variable assignments given by a).

Given a set S, $S^{m \times n}$ is the set of $m \times n$ matrices with entries taken from S.

Throughout, ω refers to the smallest real number such that for all $\varepsilon > 0$, matrix multiplication over a ring can be performed in $O(n^{\omega+\varepsilon})$ time. Note that $\omega < 2.376$, cf. [7]. We will discuss three types of matrix product in the paper; unless otherwise specified, the default is matrix product over the ring currently under discussion. The other two are the distance product ($\otimes_d$) on $\mathbb{Z} \cup \{-\infty, \infty\}$, and Boolean matrix product ($\otimes_b$) on 0–1 matrices. Let $A, B \in (\mathbb{Z} \cup \{-\infty, \infty\})^{n \times n}$. $A \otimes_d B$ is matrix product over the (min, +)-semiring; that is, the usual + in matrix product is replaced with the min operator, and $\times$ is replaced with addition. When A and B are 0–1 matrices, the Boolean product $\otimes_b$ replaces + with $\vee$ (OR) and $\times$ with $\wedge$ (AND).

2. Fast *k*-clique detecting and counting

We briefly review an algorithm [18] for detecting if a graph has a k-clique in less than n^k steps.

Theorem 1 (*Nesetril and Poljak [18]*). *Let $r \in \mathbb{Z}^+$. Then $3r$-clique on undirected graphs is solvable in $O(n^{\omega r})$ time.*

[1] The $\tilde{O}$ suppresses polynomial factors.