On the Complexity of Integer Programming with Fixed-Coefficient Scaling

Jorke M. de Vlas
Linköping Universitet, Sweden

Abstract

We give a polynomial time algorithm that solves a CSP over $\mathbf{Z}$ with linear inequalities of the form $c^{a_1}x - c^{a_2}y \leq b$ where x and y are variables, a_1, a_2 and b are parameters, and c is a fixed constant. This is a step in classifying the complexity of CSP(Γ) for first-order reducts Γ from $(\mathbf{Z}, <, +, 1)$. The algorithm works by first reducing the infinite domain to a finite domain by inferring an upper bound on the size of the smallest solution, then repeatedly merging consecutive constraints into new constraints, and finally solving the problem using arc consistency.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Complexity theory and logic

Keywords and phrases constraint satisfaction problems, integer programming, CSP dichotomy

Digital Object Identifier 10.4230/LIPIcs.CP.2024.35

Category Short Paper

Funding This work is partially supported by the Swedish Research Council (VR) under grant VR-2021-04371.

Acknowledgements I wish to thank P. Jonsson and G. Osipov for discussions and feedback.

1 Introduction

Many computational problems in theoretical computer science can be formulated as a Constraint Satisfaction Problem or CSP. In these problems the goal is to find an assignment of values to a set of variables from a given domain that satisfies some given constraints that impose relations on subsets of variables. After fixing a domain and the allowed constraint types, we are left with a computational problem and want to determine its computational complexity. In general, we are interested in determining whether a given CSP is solvable in polynomial time or NP-hard.

For Boolean domains, Schaefer's dichotomy theorem gives a full classification [9]. This result has recently been extended to arbitrary finite domains [3, 12]. For infinite domains, there is no known dichotomy and this is an area of active research [1]. We focus on TVPI, the infinite-domain CSP which consists of linear inequalities with two variables per inequality. In general, this CSP is NP-complete [6, Theorem F], but it admits many subclasses whose tractability is unknown.

Inequalities in TVPI consist of the form $ax + by \leq d$ for variables x, y and integer coefficients a, b, d. Several tractable subclasses of TVPI focus on restricting the linear coefficients a and b. Restricting to $\{\pm 1\}$, we obtain the tractable UTVPI [5]. Restricting to $\{\pm 1, \pm 2\}$, we obtain the class BTVPI, which was recently shown to also be tractable [10].

An intractable subclass of TVPI is the CSP that concerns monotone inequalities, where the coefficients a and b have different signs [6]. This class admits a pseudo-polynomial time algorithm using a rounded version of a technique known as Fourier-Motzkin elimination, which iteratively combines pairs of inequalities into new inequalities [4]. Another technique used in constraint programming is arc consistency, which iteratively reduces variable domains until a fixed state is found. Similarly to FM-elimination, finding a fixed state is NP-hard in general [2] but admits a pseudo-polynomial time algorithm [11].

Our contribution is an algorithm that combines these techniques into a polynomial-time algorithm for another subclass of TVPI. Our class concerns monotone inequalities whose coefficients are restricted to powers of c for some fixed constant $c > 0$. That is, inequalities of the form $c^{a_1}x - c^{a_2}y \leq b$. After dividing by $c^{\min(a_1,a_2)}$, we may assume that a_1 or a_2 is zero. Formally, for any $c \in \mathbf{N}$ we define R_c as the set of inequalities of the form $x \leq c^a y + b$ or $c^a x + b \leq y$ with $a \in \mathbf{N}$ and $b \in \mathbf{Z}$, and then consider $\mathrm{CSP}(\mathbf{Z}; R_c)$. In order to keep the size of the coefficients polynomial, we require that each a is given in unary. To simplify some arguments, we view the $\mathrm{CSP}(\mathbf{Z}, R_c)$ instance as a directed, weighted graph with colored arcs. We model a constraint of the form $x \leq c^a y + b$ as a *red* arc from x to y with weight $-a$, and a constraint of the other form as a *blue* arc from x to y with weight a.

The paper is structured as follows. In Section 2 we give a high-level outline of the algorithm and its main ideas. In Section 3 we give a detailed explanation of all the substeps. Finally, in Section 4 we combine all these steps to show that our algorithm is correct and satisfies the claimed runtime. To begin, we fix a value for c and an instance of $\mathrm{CSP}(\mathbf{Z}, R_c)$. For this instance, let n be the number of variables, m the number of constraints, and W the maximal value of any coefficient in any constraint. Since each a is given in unary, $\log(W)$ is polynomial in the input size.

2 Algorithm outline

The algorithm consists of four main ideas. The first one is boundedness: if the instance admits an integer solution, then the size of the "smallest" integer solution will not be too large. More precisely, we will prove that there exists an integer Ω, whose size is polynomial in the input, such that: if there exists an integer solution to the CSP, then there exists an integer solution where every variable is bounded in absolute value by 2^Ω. In particular, this implies that to solve the CSP we only need to look for solutions in a bounded subset of $\mathbf{Z}$. Let $I_\Omega := \{-2^\Omega, -2^\Omega + 1, \ldots, 2^\Omega\}$ be this subset.

The second idea is to use arc consistency. If we know a domain of possible values for some variable x, then any arc between x and some other variable y translates this into a domain of possible values for y. By starting with the initial domain $x \in I_\Omega$ for every variable x and repeatedly checking consistency along all arcs, we will eventually settle in a state where some domain is empty or where all arcs are consistent. Now, the instance has a solution if and only if all domains are nonempty: if a domain is empty the instance is obviously unsolvable, and if all domains are nonempty then we can set every variable to the smallest remaining value. This is a solution, since the smallest remaining value on the larger side of the inequality is (by arc consistency) larger than some remaining value on the smaller side of the inequality, and thus also larger than the smallest remaining value of that side. In more technical terms, min is a polymorphism.

Since the complete domains can be exponentially large, we only keep track of a description of them. It suffices to only consider integer lower bounds, so for every variable x we keep track of a lower bound $x \geq \ell, \ell \in \mathbf{Z}$ which increases during the arc consistency steps. A domain is now considered empty if we find a lower bound of at least 2^Ω. One arc consistency step now goes as follows: loop over every arc (x, y). The current lower bound at the starting vertex x implies a new lower bound on the end vertex y. If this lower bound is stricter then the bound we had for y, update y with the new bound.

This arc consistency procedure terminates in at most $n2^{\Omega+1}$ iterations: every iteration increases one of the n lower bounds by at least one, and each lower bound is contained in I_Ω. Unfortunately, this is still exponentially many. To reduce the required number of iterations, we introduce the third main idea: edge shortening.

If we have an arc from x to y and an arc from y to z, then any lower bound on x implies a lower bound on y which in turn gives a lower bound on z. In some cases (but not all!) we could have obtained this lower bound on z directly from the one on x by adding a new constraint between x and z which is functionally equivalent to the path from x to z via y. This means that if we would have added this new arc before applying the arc consistency procedure, we would have needed one iteration less to obtain the lower bound on z.

By this reasoning, we see that the arc consistency algorithm improves if we first preprocess the graph to introduce as many new arcs as possible. We do this by performing multiple edge shortening iterations. In one iteration we check every pair of consecutive arcs and, if possible, add a new arc. We will later show how to determine when this arc exists and what its parameters are. We will also prove that one edge shortening iteration reduces the required number of arc consistency steps by (roughly) one third and that after enough edge shortening iterations, the required number of arc consistency steps reduces to just two.

The problem with edge shortening is that we might introduce exponentially many new arcs and that the arcs might have exponentially large parameters. To solve this, we apply the fourth main idea: compression. We will show that any set of constraints S can be transformed into a new set S' whose total bitsize is polynomial in Ω such that any variable assignment $s \in (I_\Omega)^n$ satisfies S if and only if it satisfies S'. The intuition is that since we only concern solutions bounded by 2^Ω, then the set of variable assignments that satisfy an arc whose coefficients are much larger that 2^Ω can also be described using an arc with smaller coefficients. Furthermore, we only need at most one arc of each weight between every two variables: if there are multiple arcs of the same weight, we only need the one with the most restrictive constant term.

Overall, we obtain the following algorithm:

1. Perform the boundedness step: compute Ω.
2. Alternate between edge shortening iterations and compression steps polynomially many times (we later specify exactly how many).
3. Perform two arc consistency steps.
4. If there is a lower bound of at least 2^Ω, output NO SOLUTION. Otherwise, output the current values of the lower bounds.

In the next section, we will describe the four main steps in more detail and prove why they work.

3 Main steps

Boundedness

We will first show how to construct the global bound Ω using results from literature. Our goal is to apply the main theorem of [8], which states that if a CSP consisting of linear equations over the integers has n variables, m constraints, and coefficients bounded by a, and admits a solution consisting of positive integers, then it admits one where each integer in the solution is at most $n(ma)^{2m+1}$.

CSP(R_c) is different: our constraints are inequalities instead of equalities and the solution may contain negative integers. Fortunately, we can easily transform it into this standard form [7, Section 2.2] by using slack variables for the inequalities and expressing each unbounded variable as the difference between two positive variables. This modification adds $m + n$ additional variables and n additional constraints, so the resulting CSP has $2n + m$ variables, $m + n$ constraints, and coefficients bounded by W.

We can now apply the above theorem and find a new solution to the modified instance where each variable is positive and bounded by $\Omega' := (2n+m)((m+n)W)^{2(m+n)+1}$. This translates into a solution of the original instance where each variable may be negative again but still has an absolute value of at most Ω'. We conclude that $\Omega := \log(\Omega') = \mathcal{O}((m+n)(\log(W(m+n))$ suffices. Overall, we obtain the following proposition.

▶ **Proposition 1.** *There exists an integer $\Omega \in \mathbf{N}$, which only depends on and is polynomial in n, m and $\log(W)$, such that: if our $\mathrm{CSP}(\mathbf{Z}, R_c)$ instance has at least one integer solution, then there exists an integer solution where each variable has an absolute value of at most 2^Ω.*

Arc consistency

The second step is to apply arc consistency by propagating lower bounds along the constraint arcs. On a blue arc $c^a x + b \leq y$, a lower bound $x \geq \ell$ implies a lower bound $y \geq c^a \ell + b$. On a red arc $x \leq c^a y + b$ a lower bound $x \geq \ell$ implies a lower bound $y \geq \left\lceil \frac{\ell - b}{c^a} \right\rceil$. This chains across consecutive arcs: for any path p, a lower bound on the starting vertex results in a lower bound on the ending vertex. In general, we can identify any path with a valid lower bound by propagating the trivial lower bound $x \geq -2^\Omega$ on the starting vertex x along the path.

Conversely, every lower bound that can be obtained using arc consistency corresponds to a path; every arc consistency step effectively appends one arc to the path. Hence, we obtain an equivalence between paths and lower bounds. When viewing a path in the context of a lower bound chain, we will refer to it as a *propagation path*. We now give some properties of propagation paths.

▶ **Proposition 2.** *Propagation paths satisfy the following properties.*

- **(i)** *Let $x \geq \ell$ be some lower bound reachable by propagation paths. The length of the shortest propagation path resulting in this lower bound (or a better one) is at most $n2^{\Omega+1}$.*
- **(ii)** *Let p be a propagation path of length k ending in some lower bound $x \geq \ell$. Then, the arc consistency procedure will find this lower bound (or a better one) in at most k iterations.*

Proof. We prove the parts in order.

- **(i)** If a propagation path of minimal length contains a vertex multiple times, then the corresponding lower bound on that vertex should increase otherwise the subcycle between these occurences is redundant and the path is not minimal. So, every vertex occurs at most $|I_\Omega| = 2^{\Omega+1}$ times, which bounds the length of the path to $n2^{\Omega+1}$.
- **(ii)** We use induction on the length of p. If p has length zero, then the lower bound must be the initial lower bound $x \geq -2^\Omega$ which is found before any arc consistency iteration. Now suppose p has length $k > 0$. Let $x' \geq \ell'$ be the bound that corresponds to p with the final constraint removed. By the induction hypothesis, after $k-1$ steps of the arc consistency procedure we will have found a bound $x' \geq \ell'$ or better. One more arc consistency iteration will then propagate this lower bound into a lower bound of at least ℓ for x. ◀

A corollary of this proposition is termination of the arc consistency procedure: every lower bound that can be found by lower bound propagation will be found in at most $n2^{\Omega+1}$ iterations, so after at most this many iterations the procedure terminates.

Edge shortening

We will now describe in which cases we can concatenate consecutive arcs into a new arc.

▶ **Proposition 3.** *Let e_1 and e_2 be two consecutive arcs. Let x, y, z be their endpoints such that e_1 runs from x to y and e_2 runs from y to z. Suppose that e_1 and e_2 are colored in one of the following ways: (red, red), (blue, blue), or (blue, red). Then there exists an arc e_3 from x to z such that the set of integer values for x and z that satisfy the constraint from e_3 is the same as the set of integer values for x and z for which there exists an integer value for y such that both the constraints e_1 and e_2 are satisfied.*

Proof. We begin with the case where e_1 and e_2 are both red. Let $x \leq c^{a_1}y + b_1$ and $y \leq c^{a_2}z + b_2$ be the inequalities associated with e_1 and e_2, respectively. These inequalities combine into

$$x \leq c^{a_1}y + b_1 \leq c^{a_1}\left(c^{a_2}z + b_2\right) + b_1 = c^{a_1+a_3}z + c^{a_1}b_2 + b_1$$

We now define e_3 as this constraint: $a_3 := a_1 + a_2$ and $b_3 = c^{a_1}b_2 + b_1$. One direction is easy: if for some tuple of integer values for x and z there exists a y such that e_1 and e_2 are satisfied, then this tuple by construction satisfies e_3 as well. For the inverse direction, let (v_x, v_z) be a tuple of integer values for x and z that satisfy e_3. Then, we can set y to the value $v_y := c^{a_2}v_z + b_2$ which is integer. Now, the triplet (v_x, v_y, v_z) satisfies e_1 and e_2: e_1 rewrites to the inequality for e_3 and e_2 is satisfied by construction.

The other two cases are very similar. If e_1 and e_2 are both blue, we have the inequalities $c^{a_1}x + b_1 \leq y$ and $c^{a_2}y + b_2 \leq z$ which merge into $c^{a_1+a_2}x + c^{a_2}b_1 + b_2 \leq z$, so we set e_3 to this inequality. If (v_x, v_z) is a tuple of integer assignments for x and z that satisfies e_3, then we can set $v_y := c^{a_1}v_x + b_1$ to find a triplet of integer assignments for e_1 and e_2.

Finally, if e_1 is blue and e_2 is red, then we have the inequalities $c^{a_1}x + b_1 \leq y$ and $y \leq c^{a_2}z + b_2$. These merge into $c^{a_1}x + b_1 \leq c^{a_2}z + b_2$. This is not yet the proper form to become a valid constraint, but we can modify it a bit. We consider two cases: either $a_1 > a_2$ or $a_1 \leq a_2$. In the first case, we can rewrite the inequality to $c^{a_1-a_2}x + \frac{b_1-b_2}{c^{a_2}} \leq z$. Since z and $c^{a_1-a_2}x$ are always integer, this inequality is equivalent to $c^{a_1-a_2}x + \left\lceil \frac{b_1-b_2}{c^{a_2}} \right\rceil \leq z$. This inequality has the proper form for a blue edge: $a_3 := a_1 - a_2$ is a nonnegative integer and $b_3 := \left\lceil \frac{b_1-b_2}{c^{a_2}} \right\rceil$ is integer as well.

In the case where $a_1 \leq a_2$ we can do something analogous: we instead rewrite the equation to $x \leq c^{a_2-a_1}y + \left\lfloor \frac{b_2-b_1}{c^{a_1}} \right\rfloor$ to obtain an equation for a red edge. The correctness proof of the (blue, red) case is also analogous: if v_x and v_z are integer values for x and z that satisfy e_3, then we can set y to $c^{a_1}x + b_1$ or $c^{a_2}z + b_2$ (both work!) to obtain a triple that satisfies e_1 and e_2. ◀

▶ Remark 4. The case (red,blue) does not work since we cannot always set y to an integer value. For example, in the case $c = 2$ one might think that the arcs $x \leq 2y$ and $2y \leq z$ combine into the arc $x \leq z$, but this does not satisfy the requirements: setting x and z both to 1 satisfies the inequality $x \leq z$ but there is no integer value for y that satisfies $x \leq 2y \leq z$.

We will now consider the effect of edge shortening on the length of propagation paths.

▶ **Proposition 5.** *Let $x \geq \ell$ be a lower bound that is reachable by a propagation path of length k. Then, apply one iteration of the edge shortening procedure. Now, the bound $x \geq \ell$ will be reachable by a propagation path of length at most $\lceil 2k/3 \rceil$.*

Proof. The main idea is the following observation: for any three consecutive arcs, the edge shortening iteration will shorten at least one pair. Indeed, if the first pair is (red, blue), the only configuration in which we cannot apply edge shortening, then the second pair will be (blue, blue) or (blue, red) which are both shortable. Now let p be the path of length k used to reach the lower bound and partition it into triplets of consecutive arcs. Then, after applying the edge shortening operation, we can replace every triplet with just two arcs: the arc obtained from the shortening and the arc not used in the shortening. Overall, this will reduce p to a path of length $\lceil 2k/3 \rceil$, where the ceiling function is required to account for the (at most two) leftover arcs that do not fit in the triplets. ◀

▶ **Corollary 6.** *Let $x \geq \ell$ be a lower bound that is reachable by a propagation path. After polynomially many edge shortening iterations, this lower bound will be reachable by a propagation path of length at most 2.*

Proof. Let k be the length of the initial propagation path used to obtain the lower bound $x \geq \ell$. By Proposition 2(i), k is at most $n2^{\Omega+1}$. Every edge shortening iteration replaces k with $\lceil 2k/3 \rceil$. In particular, if $k \geq 3$ then $\lceil 2k/3 \rceil \leq 2k/3 + 2/3 \leq 2k/3 + 2k/9 = 8k/9$. So, after $\log_{9/8}(n2^{\Omega+1})$ iterations, k will be reduced to at most 3. One more iteration then reduces k to at most 2. This shows that after $\log_{9/8}(n2^{\Omega+1}) + 1$ iterations, which is polynomially many, the lower bound $x \geq \ell$ is reachable by a path of length 2. ◀

▶ Remark 7. We cannot apply more edge shortening iterations to reduce the length even further: if a propagation path of length 2 is colored in the colors (red, blue) then additional edge shortening iterations have no effect on this path.

Compression

We now show how to bound the number and size of parallel edges.

▶ **Proposition 8.** *Let S be a set of constraints. If we restrict each variable x to the domain $x \in I_\Omega$, then we can modify S into an equivalent set of constraints S' satisfying:*

- **(i)** *The (absolute) weight of each edge is bounded by $\Omega + 1$.*
- **(ii)** *There is at most one edge of each weight between any two variables.*
- **(iii)** *The constant coefficient of each constraint is at most $2^{2\Omega+2}$.*

In particular, the total bitsize of S' is polynomially bounded in Ω.

Proof. We only consider red arcs; blue arcs are analogous but with the direction of the inequalities reversed. Let e be a red arc, say $x \leq c^a y + b$. We perform three simplification steps on e to satisfy the three required properties.

- **(i)** Suppose e satisfies $a > \Omega + 1$. If $c = 1$, the value of a does not matter so we can set it to 0. If $c \geq 2$, then there is at most one value of y such that the right hand side $c^a y + b$ falls in the interval $|x| \leq 2^\Omega$. Let v_y be this value, and let $v_x := c^a v_y + b$ be the corresponding upper bound for x. The set of pairs (x, y) satisfying the constraint can now be described as: $y < v_y$ is impossible, $y = v_y$ implies $x \leq v_x$, and if $y > v_y$, there are no restrictions on x. This can also be achieved with the constraint $x \leq 2^{\Omega+1}(y - v_y) + v_x = 2^{\Omega+1}y - 2^{\Omega+1}v_y + v_x$ which does have weight $\Omega + 1$ so we simply replace e with this new constraint. This argument is visualized in Figure 1: it shows the shape of the feasible region for some constraints, and in particular it shows that the feasible region for a constraint with $a > \Omega + 1$ has the same shape as one with $a = \Omega + 1$.

(ii) Suppose there is another edge e' of the same weight, say $x \leq c^a y + b'$. Assume without loss of generality that $b \leq b'$. Then the inequality for e' follows directly from the one for e, so e' is redundant and can be removed from S.

(iii) Suppose that $|b| \geq 2^{2\Omega+2}$, and recall that $|x| \leq 2^\Omega$, $|y| \leq 2^\Omega$ and $0 \leq a \leq 2^{\Omega+1}$. We find that the left hand side of $x \leq c^a y + b$ is at most 2^Ω in absolute value while the right hand side is at least $2^{2\Omega+2} - 2^{\Omega+1} \cdot 2^\Omega = 2^{2\Omega+1}$ in absolute value. This shows that the constraint is either always satisfied or always unsatisfied (depending on the sign of b). So, we find that either the instance is infeasible or the constraint can be removed without affecting feasibility. ◀

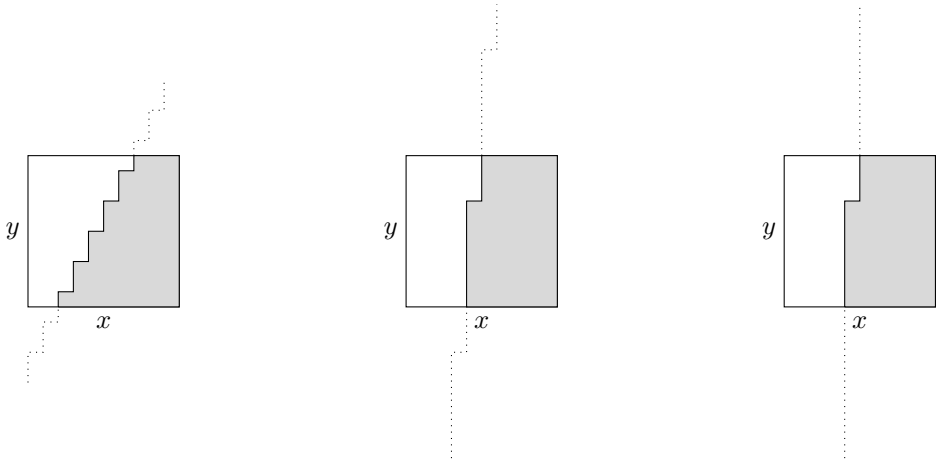

Figure 1 Some possible shapes for the feasible region of a single constraint. The square is the domain $(I_\Omega)^2$ and the wiggled line is the inequality. From left to right, the coefficients satisfy $1 < a < \Omega + 1$, $a = \Omega + 1$ and $a > \Omega + 1$.

4 Combining the parts

We will now give a formal proof of correctness for the algorithm described in Section 1 using the propositions from Section 3. This is split into two theorems, one for correctness and one for the runtime.

▶ **Theorem 9.** *The algorithm described in Section 1 returns an integer solution to the* CSP(R_c) *instance if one exists. Otherwise, it will output NO SOLUTION.*

Proof. We first show that if the algorithm returns some variable assignment, then this is indeed a solution. The final step asserts that all values of this assignment are at most 2^Ω, so by Proposition 8 the correctness is not affected by the compression steps. We now claim that the returned variable assignment satisfies all constraints of the modified instance obtained from the edge shortening iterations and compression steps: suppose to the contrary that there is an unsatisfied constraint. This means that one additional arc consistency step would have updated some lower bound to a new value, and most importantly that this new lower bound is a bound reachable by propagation paths. Propositions 3 and 8 show that this lower bound is also reachable in the original instance. Proposition 2(i) shows that the length of the shortest propagation path to reach this lower bound is at most $n2^{\Omega+1}$. Corollary 6 now shows that the edge shortening steps reduce it to one of length at most 2. Proposition 2(ii) then shows that this lower bound was already found in the two arc consistency steps. This is

a contradiction: we conclude that the returned variable assignment satisfies all constraints of the modified instance. Since all original constraints are either still present in the modified instance or replaced by an equivalent or stricter constraint, the solution must also satisfy the original instance. This completes the first half of the proof.

We now show that if a solution exists, the algorithm will find it. By the corollary after Proposition 2, just using the arc consistency procedure would determine existence of a solution in $n2^{\Omega+1}$ steps. In particular, the assigned values in this solution are the result of a propagation path of length at most $n2^{\Omega+1}$. By Corollary 6, the edge shortening iterations reduce all these propagation paths to paths of length at most two, so by Proposition 2(ii), every assigned value will be found in the two arc consistency steps. Therefore, the solution will be found by the algorithm. This completes the second half of the proof, and consequently the full proof. ◀

▶ **Theorem 10.** *The algorithm described in Section 1 runs in polynomial time.*

Proof. We first note that Ω is polynomial in the size of the input. After each compression step, Proposition 8 shows that the current instance contains at most $2\Omega + 3$ constraints between any two variables, so at most $(2\Omega + 3)n^2$ in total, which is polynomially many. Denote this value by N. Furthermore, the coefficients of every constraint have polynomial bitsize of $\mathcal{O}(\Omega)$, so most arc operations takes $\mathcal{O}(\Omega)$ time.

Each edge shortening iteration adds at most one new arc for any two consecutive arcs, taking $\mathcal{O}(\Omega)$ time per arc pair and resulting in at most N^2 new arcs after this step. The compression step afterwards takes $\mathcal{O}(\Omega)$ time per arc to reduce the number of arcs back down to N again. Together, one edge shortening iteration and compression step take $\mathcal{O}(N^2\Omega)$ time. One exception is the first compression step: since there are initially m arcs instead of at most N, and coefficients are initially bounded by W instead of 2^Ω, the first step takes $\mathcal{O}(\log(W)m^2)$ time. Since these two steps are repeated $\log_{9/8}(n2^{\Omega+1}) + 1 = \mathcal{O}(\Omega \log(n))$ times, the overall runtime of these steps is $\mathcal{O}(\Omega^2 \log(n)N^2 + \log(W)m^2)$; still polynomial.

Finally, each arc consistency step checks every arc at most once, so this adds another $\mathcal{O}(N\Omega)$ time to the computation. Overall, we conclude that the algorithm runs in $\mathcal{O}(\Omega^2 \log(n)N^2 + \log(W)m^2)$ time. When transformed back to the original input parameters, this becomes $\mathcal{O}(\Omega^2 \log(n)(\Omega n^2)^2 + \log(W)m^2) = \mathcal{O}(\Omega^4 n^4 \log(n) + \log(W)m^2)$. Since $\Omega = \mathcal{O}((m+n)\log(W(m+n)))$ the $\log(W)m^2$ vanishes and the total runtime is $\mathcal{O}((m+n)^4 \log(W(m+n))^4 n^4 \log(n))$. ◀

5 Discussion

To conclude the paper, we have shown that the CSP with constraints of the form $c^a x + b \leq y$ and $x \leq c^a y + b$ is solvable in polynomial time. We stress that the given time complexity is far from optimal; a better runtime analysis and some optimizations will reduce the runtime significantly.

We end with some possible future research directions.

1. What are the implications on the infinite domain CSP-dichotomy over $\mathbf{Z}$? In particular, does our method generalize to CSPs whose parameters are restricted to a larger domain?
2. What is the complexity of the optimization variant of this CSP? The current algorithm outputs a solution where all variable assignments are minimal and can easily be modified into an algorithm where these are maximal. However, there is no simple reduction to find the optimal solution where we want some variables to be large and others to be small.
3. How far can we reduce the time complexity of this algorithm?

References

1. Manuel Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction*. Lecture Notes in Logic. Cambridge University Press, 2021.
2. Lucas Bordeaux, George Katsirelos, Nina Narodytska, and Moshe Vardi. The complexity of integer bound propagation. *J. Artif. Intell. Res. (JAIR)*, 40:657–676, October 2011. doi:10.1613/jair.3248.
3. Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.
4. Dorit S. Hochbaum and Joseph (Seffi) Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6):1179–1192, 1994. doi:10.1137/S0097539793251876.
5. Joxan Jaffar, Michael J. Maher, Peter J. Stuckey, and Roland H. C. Yap. Beyond finite domains. In Alan Borning, editor, *Principles and Practice of Constraint Programming*, pages 86–94, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
6. J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, 1985. doi:10.1137/0214016.
7. K.G. Murty. *Linear Programming*. Wiley, 1983.
8. Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, October 1981. doi:10.1145/322276.322287.
9. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. Association for Computing Machinery. doi:10.1145/800133.804350.
10. Piotr Wojciechowski and K. Subramani. A faster algorithm for determining the linear feasibility of systems of btvpi constraints. In Leszek Gąsieniec, editor, *SOFSEM 2023: Theory and Practice of Computer Science*, pages 313–327, Cham, 2023. Springer International Publishing.
11. Zhang Yuanlin and Roland H. C. Yap. Arc consistency on n-ary monotonic and linear constraints. In Rina Dechter, editor, *Principles and Practice of Constraint Programming – CP 2000*, pages 470–483, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
12. Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, October 2017. doi:10.1109/FOCS.2017.38.