From Elliott-MacMahon to an algorithm for general linear constraints on naturals*

Eric Domenjoud[1] and Ana Paula Tomás[2]

[1] CRIN/CNRS & INRIA-Lorraine
Eric.Domenjoud@loria.fr
[2] LIACC/University of Porto
apt@ncc.up.pt

Abstract. We describe a new algorithm for solving a conjunction of linear diophantine equations, inequations and disequations in natural numbers. We derive our algorithm from one proposed by Elliott in 1903 for solving a single homogeneous equation. This algorithm was then extended to solve homogeneous systems of equations by MacMahon. We show how it further extends to an algorithm which solves general linear constraints in nonnegative integers and allows a parallel implementation. This algorithm provides a parametric representation of the solutions from which minimal solutions may be extracted immediately. Moreover, it may be easily implemented in parallel. It has however one drawback: it is redundant which means that the same minimal solution is usually generated many times. We show how this redundancy may be eliminated at the cost of an increase in the space complexity.

keywords: integer programming, linear diophantine constraints

Introduction

The problem of solving linear equations with integer coefficients over the natural numbers plays an important role in computer science. In automated deduction, this problem is at the heart of the associative commutative unification algorithm. It appears also in many other fields like data flow analysis, Petri nets or even abstract interpretation in logic programming.

Since 1978 when Huet [9] proposed his algorithm for solving a single homogeneous equation, much effort has been spent on trying to solve efficiently this problem [3, 4, 5, 7, 11]. In the last few years, several efficient algorithms have been proposed which solve systems of such equations. However, it seems that very few people have been aware of an old algorithm proposed in 1903 by Elliott [6] and later extended by MacMahon [10]. This algorithm has nice features which allowed us to extend it to solve more elaborate problems involving inequations ($\geq, \leq, <, >$) and also disequations ($\neq$). This problem was addressed very recently in two different works. One of them [2] extends the algorithm of

* This work was partly supported by the SOL project, HCM *#CHRX CT92 0053*

A. Fortenbacher to solve systems of equations and inequations while the other one [1] proposes a new method for solving an arbitrary conjunction of equations, inequations and disequations. This second algorithm is however a bit different since it does not compute a basis of the set of solutions, but rather a parametric representation of this set. This parametric representation consists in a finite set of parametric expressions which together generate all nonnegative solutions when the parameters range over the natural numbers. In addition, this parametric representation is unambiguous in the sense that each solution is generated exactly once. For instance, such a parametric representation of the solutions of $2x = y + z$ may be

$$\begin{cases} x = u_1 + u_2 \\ y = 2u_1 \\ z = 2u_2 \end{cases} \qquad \begin{cases} x = u_1 + u_2 + 1 \\ y = 2u_1 + 1 \\ z = 2u_2 + 1 \end{cases}$$

while the basis of the set of solutions is $\{(1, 2, 0), (1, 1, 1), (1, 0, 2)\}$. Note that if we are not interested in unambiguity, the basis of the set of solutions always provides a parametric representation. Unfortunately, when disequations are involved, such a basis does not exist in general. Thus a parametric representation seems to be the best one may hope. However, depending on the application we have in mind, some parametric representations may be more useful.

In this paper, we derive from the algorithm of Elliott-MacMahon a new algorithm which, given a problem $\mathcal{P}$ containing d disequations, computes at once a parametric representation of the solutions of each of the 2^d problems obtained by replacing each disequation $\alpha X + \beta \neq 0$ in $\mathcal{P}$, alternatively with $\alpha X + \beta > 0$ and $\alpha X + \beta < 0$. The advantage of this representation is that each of the solution sets of these 2^d problems may now be represented through a finite basis. Moreover, this basis may be extracted immediately from the parametric representation we obtain. Note that this last property does not hold in general for any parametric representation, even if only equations are considered. For instance,

$$\begin{cases} x = 2u + 1 \\ y = 2u \end{cases} \qquad \begin{cases} x = 2u + 2 \\ y = 2u + 1 \end{cases}$$

is a parametric representation of the solutions of $x - y - 1 = 0$. If we want to describe these solutions through minimal ones, we need both the minimal solutions of this equation and the minimal nonzero solutions of its homogeneous part $x - y = 0$. But the parametric representation does not *contain* the solution $(x, y) = (1, 1)$ which is the only minimal nonzero solution of $x - y = 0$. The method we present does not have this problem.

The paper is organised as follows. In section 1, we introduce the main notions about linear diophantine problems. In section 2 we describe the original algorithm of Elliott-MacMahon for homogeneous equations and reformulate it as a transformation process on constrained parametric expressions. In sections 3 and 4, we extend the algorithm to solve first non-homogeneous equations and then arbitrary diophantine constraints involving equations, inequations and disequations. Finally, in section 5, we show how we can eliminate the redundancy from this algorithm.

1 Preliminaries

We introduce here basic notions about linear diophantine constraints and their solutions. In the sequel, $\mathbb{N}$ is the set of natural numbers and $\mathbb{Z}$ is the set of integers.

A linear diophantine constraint has the form $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \# 0$ where $\#$ is any predicate in $\{=, \geq, \leq, >, <, \neq\}$ and $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Z}$. It is homogeneous if $\beta = 0$. We also write such a constraint as $\alpha X + \beta \# 0$ where α and X are tuples. The constraint is an *equation* if $\#$ is $=$, an *inequation* if $\# \in \{\geq, \leq, >, <\}$, and a *disequation* if $\#$ is $\neq$. For the sake of simplicity, we restrict our attention to the case where $\# \in \{=, \geq, \neq\}$. This is not really a restriction since in integers, any inequation may be reduced to $\alpha X + \beta \geq 0$.

A system of linear diophantine constraints is a finite conjunction of constraints. We also write it as $AX + B \# 0$ where A is a matrix of which the entry $\alpha_{i,j}$ is the coefficient of x_j in the i^{th} constraint, B is a tuple of integers, and $\#$ is a tuple of predicates. We also use the notation $A^1 x_1 + \cdots + A^n x_n + B \# 0$ where A^i denotes the i^{th} column of A. A system $AX + B \# 0$ of equations and inequations is homogeneous if $B = 0$. The *homogeneous part* of $AX + B \# 0$ is the homogeneous system $AX \# 0$.

We are only interested in nonnegative solutions of the constraints. Let $\mathcal{P} \equiv AX + B = 0 \wedge A'X + B' \geq 0$ be a system of equations and inequations and $\mathcal{P}^H$ its homogeneous part. A nonzero solution S of $\mathcal{P}^H$ is minimal if it is not the sum of two nonzero solutions of $\mathcal{P}^H$. A solution S of $\mathcal{P}$ is minimal if it is not the sum of a solution of $\mathcal{P}$ and a nonzero solution of $\mathcal{P}^H$. If Y is a tuple of n new variables where n is the number of inequations in $\mathcal{P}$ then a nonzero solution S of $\mathcal{P}^H$ is minimal if and only if $(S, A'S)$ is minimal in the componentwise ordering in the set of nonzero solutions of $AX = 0 \wedge A'X - Y = 0$. A solution S of $\mathcal{P}$ is minimal if and only if it is minimal in the componentwise ordering in the set of solutions of $AX + B = 0 \wedge A'X - Y + B' = 0$. If $\mathcal{N} = \{N_1, \ldots, N_p\}$ is the set of minimal solutions of $\mathcal{P}$ and $\mathcal{H} = \{H_1, \ldots, H_q\}$ is the set of minimal nonzero solutions of $\mathcal{P}^H$, the set of all solutions of $\mathcal{P}$ is $\{N_i + H_1\lambda_1 + \cdots + H_q\lambda_q \mid i = 1, \ldots, p,\ \lambda_1, \ldots, \lambda_q \in \mathbb{N}\}$. We write this set as $\mathcal{N} + \mathcal{H}^*$ (by convention, $\varnothing^* = \{0\}$).

We shall also be interested in another description of the solutions. A parametric representation of the solutions of $AX + B \# 0$ is a finite set of expressions of the form $X = S_1 u_1 + \cdots + S_k u_k + D$ such that (i) for any natural numbers $\lambda_1, \ldots, \lambda_k$, $S_1\lambda_1 + \cdots + S_k\lambda_k + D$ is a solution of $AX + B \# 0$; (ii) for each solution S of $AX + B \# 0$, there exists at least one expression $X = S_1 u_1 + \cdots + S_k u_k + D$ in the set and natural numbers $\lambda_1, \ldots, \lambda_k$, such that $S = S_1\lambda_1 + \cdots + S_k\lambda_k + D$. For instance, with the previous notations, the set $\{X = N_i + H_1 u_1 + \cdots + H_q u_q \mid i = 1, \ldots, p\}$ is a parametric representation of the solutions of $\mathcal{P}$.

2 Elliott-MacMahon's method (1903/1916)

We describe here the method of Elliott for solving a single homogeneous linear diophantine equations. We give no correctness proof. The interested reader may refer to [6] for more details.

2.1 The original method

Given an homogeneous diophantine equation

$$a_1 x_1 + \cdots + a_n x_n = 0 \tag{1}$$

where $a_1, \ldots, a_n \in \mathbb{Z}$, Elliott considers the generating function

$$\prod_{i=1}^{n} (1 + Y_i \lambda^{a_i} + Y_i^2 \lambda^{2a_i} + \cdots) = \prod_{i=1}^{n} \frac{1}{1 - Y_i \lambda^{a_i}}$$

where Y_i's and λ are formal indeterminates. The expansion of this function is the sum of all terms of the form $Y^\gamma \lambda^\alpha$ where $\gamma \in \mathbb{N}^n$ and $a_1\gamma_1 + \cdots + a_n\gamma_n = \alpha$. Thus a tuple $\gamma \in \mathbb{N}^n$ is a solution of (1) if and only if the expansion contains the term Y^γ. The idea of Elliott is to transform this generating function until factors not containing λ appear. For this, he uses the identity

$$\frac{1}{1-R}\,\frac{1}{1-S} = \frac{1}{1-RS}\left(\frac{1}{1-R} + \frac{1}{1-S} - 1\right) \tag{2}$$

At each transformation step, the generating function is a sum of terms of the form $\pm \prod (1 - Y^\gamma \lambda^\alpha)^{-1}$. He selects one of these terms and two of its factors to which he applies the transformation above and distributes. This yields three new terms. When there is no *good choice* left, the process stops. He then collects in all terms the factors which do not contain λ. The set of corresponding exponents of Y contains all minimal nonzero solutions of (1).

The main question here is what a *good choice* is. The criterion for choosing the factors must fulfill two requirements: it must ensure both termination and completeness of the algorithm. By completeness, we mean that when the algorithm stops, all minimal nonzero solutions have been found. Elliott chooses two factors with respectively the smallest negative and the largest positive exponent for λ. With this criterion, termination is easily proved. At each step, either the largest difference between two exponents of λ decreases, or this difference does not change and the number of exponents that are equal to the largest one or to the smallest one decreases. A complete proof may be found in [6].

Example 1. *Let us consider the equation $2x_1 + x_2 - 3x_3 = 0$. The generating function is*

$$\frac{1}{1-Y_1\lambda^2}\,\frac{1}{1-Y_2\lambda}\,\frac{1}{1-Y_3\lambda^{-3}}$$

Applying the transformation to the first and third factors, we get

$$\begin{aligned}
& \frac{1}{1-Y_2\lambda}\,\frac{1}{1-Y_1Y_3\lambda^{-1}}\left(\frac{1}{1-Y_1\lambda^2} + \frac{1}{1-Y_3\lambda^{-3}} - 1\right) \\
= & \frac{1}{1-Y_2\lambda}\,\frac{1}{1-Y_1Y_3\lambda^{-1}}\,\frac{1}{1-Y_1\lambda^2} + \frac{1}{1-Y_2\lambda}\,\frac{1}{1-Y_1Y_3\lambda^{-1}}\,\frac{1}{1-Y_3\lambda^{-3}} \\
- & \frac{1}{1-Y_2\lambda}\,\frac{1}{1-Y_1Y_3\lambda^{-1}}
\end{aligned}$$

which may be further transformed by selecting for instance the second and third factors in the first term. In the end, we get a sum of 19 terms among which for instance

$$\frac{1}{1-Y_1^3Y_3^2}\,\frac{1}{1-Y_1Y_2Y_3}\,\frac{1}{1-Y_1Y_3\lambda^{-1}}$$

Since the first and second factors do not contain λ, $(3,0,2)$ *and* $(1,1,1)$ *are solutions of the equation. The set of all such tuples is* $\{(3,0,2),(1,1,1),(0,3,1)\}$ *which contains all minimal nonzero solutions.*

Extending this algorithm to solve systems of homogeneous equations is quite straightforward and such an extension was described by MacMahon [10]. The coefficients a_i of the equation (1) are now tuples in $\mathbb{Z}^m$ where m is the number of equations in the system, and instead of a single indeterminate, λ is a tuple $(\lambda_1,\ldots,\lambda_m)$. Again, the criterion for choosing two factors to be combined must ensure both termination and completeness. The method proposed by MacMahon consists actually in handling the equations one by one. First eliminate λ_1 and then λ_2, considering only factors not containing λ_1 and so on until λ_m.

2.2 Reformulation as constraint transformations

Let us show how this algorithm may be reformulated in a way which allows for simplifications and extensions. Each term generated during the transformation process has the form

$$\pm\frac{1}{1-Y^{M^1}\lambda^{A^1}}\cdots\frac{1}{1-Y^{M^k}\lambda^{A^k}} \tag{3}$$

The solutions of (1) found by transforming further this term have the form $X = M^1u_1+\cdots+M^ku_k$ where $u_1,\ldots,u_k$ are natural parameters satisfying $A^1u_1+\cdots+A^ku_k = 0$. If we interpret each term in this way, we may see the whole process as transforming constrained parametric expressions of the form

$$X = M^1u_1+\cdots+M^ku_k \parallel A^1u_1+\cdots+A^ku_k = 0 \tag{4}$$

with $\forall i$, $M^i \in \mathbb{N}^n$. This may also be written in matrix form as $X = MU \parallel AU = 0$ where M and A are matrices the columns of which are respectively the M^i's and A^i's. The process is initialised with $X = U \parallel \mathcal{A}U = 0$ where $\mathcal{A}$ is the matrix of the problem we started from. The algorithm transforms this expression until a disjunction of solved forms is found. A solved form is a constrained parametric expression with an empty constraint. We write it simply as $X = MU$. We shall describe the algorithm through transformation rules operating on finite disjunctions of constrained parametric expressions. For each rule, we just describe the way it transforms one term of the disjunction.

One may check that the interpretations of the three terms resulting from the application of (2) to (3) are actually obtained by choosing $u_i \le u_j$, $u_i \ge u_j$ or $u_i = u_j$ and performing the corresponding replacements in (4). This means replacing u_j with $u_i+u'_j$ or u_i with $u_j+u'_i$ or u_j with u_i where u'_i and u'_j are new parameters. The choice $u_i = u_j$ is obviously useless since it is contained in both other choices. After the replacement of u_i with $u_j + u_i'$, u_i completely disappears so that we may rename u_i' to u_i. The same holds for u_j'. In the end, we get a transformation rule which transforms a constrained parametric expression into a disjunction.

$$P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow u_i + u_j] \vee (P \parallel \mathcal{C})[u_j \leftarrow u_j + u_i]$$

$(P \parallel \mathcal{C})[u_i \leftarrow u_i + u_j]$ denotes the replacement of u_i with $u_i + u_j$ everywhere in $P \parallel \mathcal{C}$. Since $u_i \geq u_j \vee u_i \leq u_j$ always holds, this transformation always preserves the set of solutions regardless of how we choose u_i and u_j. We have nevertheless to choose them so as to ensure termination. We take here the same criterion as Elliott-MacMahon for choosing the two parameters u_i and u_j. Their coefficients are respectively the least negative and the greatest positive one in the first constraint. If this first constraint is $\alpha_1 u_1 + \cdots + \alpha_k u_k = 0$ where $[\alpha_1 \ \cdots \ \alpha_n]$ is the first row of A, the condition may be reformulated as $\alpha_i \alpha_j = \min\{\alpha_p \alpha_q\} < 0$. The corresponding transformation rule for constrained parametric expressions is given below. $\mathcal{C}'$ denotes the conjunction of remaining constraints.

$$
\begin{array}{ll}
(R_1) & P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow u_i + u_j] \vee (P \parallel \mathcal{C})[u_j \leftarrow u_j + u_i] \\
 & \text{if } \mathcal{C} \equiv \alpha_1 u_1 + \cdots + \alpha_n u_n = 0 \wedge \mathcal{C}' \text{ and } \min\{\alpha_p \alpha_q\} = \alpha_i \alpha_j < 0
\end{array}
$$

This rule is however not sufficient to get solved forms. For instance, it does not apply to $X = u_1 \parallel 2u_1 = 0$ although it is unsolved. This happens when the coefficients of the first constraint are either all nonnegative or all nonpositive. In this case, all parameters with nonzero coefficients may be set to 0. We are then left with a constraint $0 = 0$ which is trivially satisfied and may be removed. This is the purpose of the two rules below.

$$
\begin{array}{lll}
(R_2) & P \parallel \mathcal{C} & \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0] \\
 & \multicolumn{2}{l}{\text{if } \mathcal{C} \equiv \alpha_1 u_1 + \cdots + \alpha_n u_n = 0 \wedge \mathcal{C}',\ \alpha_i \neq 0 \text{ and either } \alpha \geq 0 \text{ or } \alpha \leq 0} \\
(R_3) & P \parallel 0 = 0 \wedge \mathcal{C}' & \vdash \quad P \parallel \mathcal{C}'
\end{array}
$$

where $\alpha \geq 0$ (resp. ≤ 0) means $\forall i,\ \alpha_i \geq 0$ (resp. ≤ 0).

Theorem 1. *Repeated application of rules R_1, R_2 and R_3 to $X = U \parallel \mathcal{A}U = 0$ terminates and returns solved forms.*

Proof. The termination proof is exactly the same as for Elliott-MacMahon's algorithm. We omit it. To prove the second part of the theorem, we have to show that the algorithm does not stop before solved forms are reached. This means that as long as an expression $P \parallel \mathcal{C}$ is not in solved form, one rule may be applied to it. If $P \parallel \mathcal{C}$ is not in solved form, $\mathcal{C}$ has the form $\alpha_1 u_1 + \cdots + \alpha_k u_k = 0 \wedge \mathcal{C}'$. If at least one α_i is negative and one α_j is positive, R_1 applies. Otherwise, α_i's are either all nonnegative or all nonpositive. If at least one of them is different from 0, R_2 applies, otherwise, R_3 applies. □

Since each transformation rule preserves the set of solutions of $\mathcal{A}X = 0$, when the algorithm stops, we are left with solved forms the disjunction of which is equivalent to the initial problem $\mathcal{A}X = 0$. We want now to extract minimal solutions from this parametric representation.

Theorem 2. *Let $X = S_{i,1}u_1 + \cdots + S_{i,n_i}u_{n_i}$, $i = 1, \ldots, r$ be the solved forms obtained from $X = U \parallel \mathcal{A}U = 0$. The set $\cup_{i=1,\ldots,r}\{S_{i,1}, \ldots, S_{i,n_i}\}$ contains all minimal nonzero solutions of $\mathcal{A}X = 0$.*

Proof. Let S be a minimal nonzero solution of $\mathcal{A}X = 0$. There must be a solved form $X = S_{i,1}u_1 + \cdots + S_{i,n_i}u_{n_i}$ and naturals $z_1, \ldots, z_{n_i}$ such that $S = S_{i,1}z_1 + \cdots + S_{i,n_i}z_{n_i}$. Since S is minimal, it must be one of the $S_{i,j}$'s. □

It is also possible to see the algorithm as operating on matrices. Given a constrained parametric expression $X = MU \parallel AU = 0$, let $X' = M'U' \parallel A'U = 0$ be obtained by applying to it one of the rules R_1, R_2, R_3. If R_1 is applied, then M' and A' are obtained by adding either the i^{th} column to the j^{th} one in both A and M, or the j^{th} column to the i^{th} one. If R_2 is applied, M' and A' are obtained by removing the i^{th} column from both M and A. Finally, if R_3 is applied, $M' = M$ and A' is obtained by removing the first row of A. In this setting, the algorithm is initialised with $M = I$ and $A = \mathcal{A}$ where I denotes the identity matrix. It stops when A has no row left.

Example 2. *We consider again the equation $2x_1 + x_2 - 3x_3 = 0$. Figure 1 shows in a tree form all applications of R_1. At each node $X = MU \parallel AU = 0$, we only represent the matrices A and M. The line delimits the matrices. The arrows above A indicate the selected parameters. Instead of showing the application of R_2 and R_3, we frame the solutions found in the end. Other columns correspond to parameters set to 0 by R_2 leaving constraints of the form $0 = 0$ which are deleted by R_3. By collecting the solutions found in the leaves, we get as before the set $\{(3,0,2), (1,1,1), (0,3,1)\}$.*

On this example, one may notice the main drawback of the method: its redundancy. Indeed, the same solution is in general found several times. This redundancy is due to the way we branch in the rule R_1. The two branches correspond respectively to the choices $u_i \geq u_j$ and $u_i \leq u_j$. We then find the solutions corresponding to $u_i = u_j$ in both branches. We discuss in section 5 a method which avoids this behaviour.

3 Extension to non-homogeneous equations

Up to now, we only dealt with solving homogeneous equations. Since our final aim is to solve general constraints, we first have to be able to handle the constant term $\mathcal{B}$ in $\mathcal{A}X + \mathcal{B} \# 0$. We start by showing how the previous algorithm may be extended to solve non-homogeneous equations of the form $\mathcal{A}X + \mathcal{B} = 0$. We show in the next section that the algorithm we get solves also the general case.

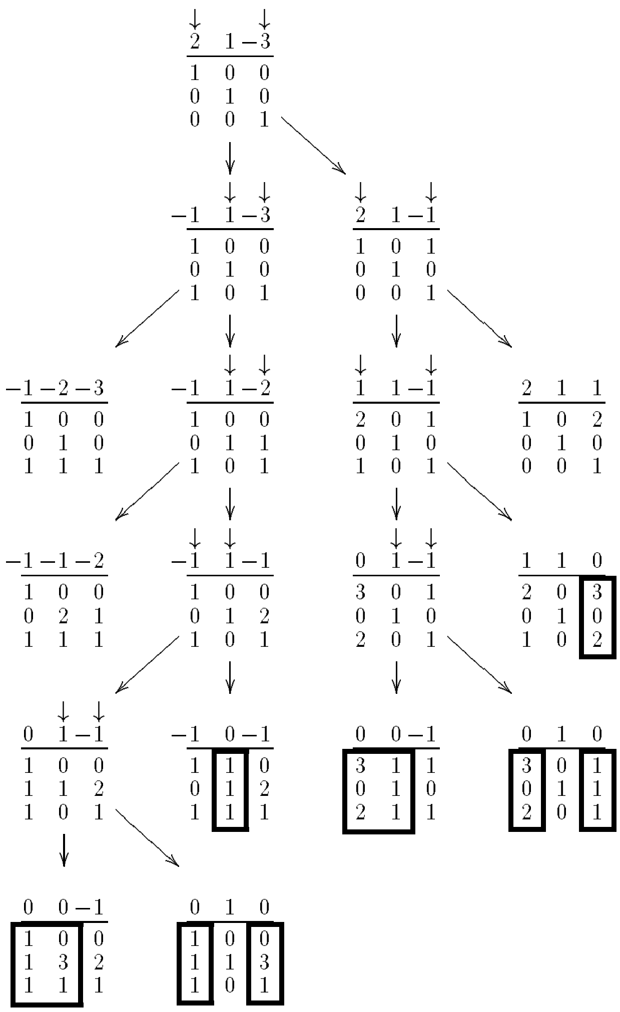

Fig. 1. solving $2x_1 + x_3 - 3x_3 = 0$

The set of solutions of $\mathcal{A}X + \mathcal{B} = 0$ is $\mathcal{N} + \mathcal{H}^*$ where $\mathcal{N}$ is the set of minimal solutions and $\mathcal{H}$ is the set of minimal nonzero solutions of the homogeneous part $\mathcal{A}X = 0$. We show how the previous algorithm may be extended to get a parametric representation of the solutions of $\mathcal{A}X + \mathcal{B} = 0$ and how from this representation, we get immediately $\mathcal{N}$ and $\mathcal{H}$.

The constrained expressions we consider have the form $X = MU + D \parallel AU + B = 0$ and the process is initialised with $X = U \parallel \mathcal{A}U + \mathcal{B} = 0$. Provided we change slightly the condition of R_1, the transformations we gave in section 2 are still valid but do not suffice anymore. Indeed, we could not get solved forms of $2x_1 + x_2 - 3 = 0$ for instance. Therefore, we introduce a new transformation, corresponding to the choice $u_i = 0$ or $u_i \geq 1$ for some parameter u_i. The solutions are obviously preserved because $u_i = 0 \vee u_i \geq 1$ always holds in natural numbers. The meaning of $u_i \geq 1$ is $u_i = u'_i + 1$ for some new parameter u'_i. Again, replacing u_i with $u'_i + 1$ makes u_i disappear so that we may rename u'_i to u_i. We must also take into account the fact that a constraint may now have no solution. For instance, $u + 1 = 0$ has no solution in natural numbers. Thus, we add a deletion rule which removes expressions with an unsatisfiable constraint. The modified set of rules is given below.

(R_1) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow u_i + u_j] \vee (P \parallel \mathcal{C})[u_j \leftarrow u_j + u_i]$
if $\mathcal{C} \equiv \alpha U + \beta = 0 \wedge \mathcal{C}'$, $\alpha_i \alpha_j < 0$, $\forall p, q$, $\alpha_i \alpha_j \leq \alpha_p \alpha_q$, $\forall p$, $\alpha_i \alpha_j \leq \alpha_p \beta$

(R_2) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0]$
if $\mathcal{C} \equiv \alpha U + \beta = 0 \wedge \mathcal{C}'$, $\beta = 0$, $\alpha_i \neq 0$ and either $\alpha \geq 0$ or $\alpha \leq 0$

(R_3) $P \parallel 0 = 0 \wedge \mathcal{C}' \quad \vdash \quad P \parallel \mathcal{C}'$

(R_4) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0] \vee (P \parallel \mathcal{C})[u_i \leftarrow u_i + 1]$
if $\mathcal{C} \equiv \alpha U + \beta = 0 \wedge \mathcal{C}'$, $\alpha_i \beta < 0$, $\forall p, q$, $\alpha_i \beta \leq \alpha_p \alpha_q$, $\forall p$, $\alpha_i \beta \leq \alpha_p \beta$

(R_5) $P \parallel \alpha U + \beta = 0 \wedge \mathcal{C}' \quad \vdash \quad \perp$
if either $\alpha \geq 0$ and $\beta > 0$, or $\alpha \leq 0$ and $\beta < 0$

Like for the homogeneous case, these rules may be seen as operating on matrices. When applied to $X = MU + D \parallel AU + B = 0$, the effect of the new rule R_4 is to either delete the i^{th} column of both M and A or to add this column to the constant term. The process is then initialised with $M = I$, $D = 0$, $A = \mathcal{A}$ and $B = \mathcal{B}$ where $\mathcal{A}X + \mathcal{B} = 0$ is the problem we started from. The effect of R_5 is obvious and R_1, R_2 and R_3 act as before.

Theorem 3. *Repeated application of rules R_1, R_2, R_3, R_4 and R_5 to $X = U \parallel \mathcal{A}U + \mathcal{B} = 0$ terminates and returns solved forms.*

Proof. The same as for theorem 1 provided in $\alpha U + \beta$ we consider β as a coefficient. □

When the algorithm stops, we are left with a finite disjunction of parametric expressions which is equivalent to the initial problem $\mathcal{A}X + \mathcal{B} = 0$. We want to find the minimal solutions.

Theorem 4. *Let $X = S_{i,1}u_1 + \cdots + S_{i,n_i}u_{n_i} + D_i$, $i = 1, \ldots, r$ be the solved forms obtained after transformation of $X = U \parallel \mathcal{A}U + \mathcal{B} = 0$. We have*

1. *the set $\{D_1, \ldots, D_r\}$ contains all minimal solutions of $\mathcal{A}X + \mathcal{B} = 0$.*
2. *if $\mathcal{A}X + \mathcal{B} = 0$ has a solution then the set $\bigcup_{i=1,\ldots,r}\{S_{i,1}, \ldots, S_{i,n_i}\}$ contains all minimal nonzero solutions of $\mathcal{A}X = 0$*

Before giving the proof, let us point out that this theorem is less trivial than it seems. We gave in the introduction an example of a parametric representation which does not have this property.

Proof. (1) follows immediately from the definition of minimal solutions.

(2) we show that, provided $\mathcal{A}X + \mathcal{B} = 0$ has a solution, the transformation rules preserve the set of solutions of $\mathcal{A}X = 0$. We get then the result using the same argument of minimality as for the proof of theorem 2.

This is obvious for R_1, R_2 and R_3 since they only operate on the homogeneous part and were already shown to preserve its solutions. The result also obviously holds for R_5 which applies only to unsatisfiable constraints. The only problem comes from R_4. When R_4 is applied to $X = MU + D \parallel AU + B = 0$, two cases may occur. If $AU + (A^i + B) = 0$ is satisfiable, the set of solutions of the homogeneous part is preserved since this homogeneous part does not change. Otherwise, we must show that any solution of $X = MU \parallel AU = 0$ is a solution of $X = M^1u_1 + \cdots + M^{i-1}u_{i-1} + M^{i+1}u_{i+1} + \cdots + M^n u_n \parallel A^1u_1 + \cdots + A^{i-1}u_{i-1} + A^{i+1}u_{i+1} + \cdots + A^n u_n = 0$, or equivalently $AU = 0 \Rightarrow u_i = 0$. We show it by contradiction. Assume S is a solution of $AU = 0$ and $s_i \geq 1$. Now let V be a solution of $AU + B = 0$. We have $A(V + S - e_i) + A^i + B = 0$ where e_i is the i^{th} vector of the canonical basis. Since $s_i \geq 1$, $V + S - e_i$ is nonnegative so that it is a solution of $AU + A^i + B = 0$, contradicting the assumption that it is unsatisfiable. □

4 Extension to general linear constraints

We now extend the previous algorithm to solve a finite conjunction of linear constraints of the form $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \# 0$ where $\#$ is any predicate in $\{=, \geq, \leq, <, >, \neq\}$. We want to solve such a problem directly, without introducing disjunctions or slack variables which would increase the complexity. As we shall see, the transformation rules described so far suffice to solve the problem. We just have to change their conditions to take the new predicates into account. For the sake of simplicity, we restrict our attention to predicates in $\{=, \geq, \neq\}$. This weakens in no way the method since $L \leq 0$, $L < 0$ and $L > 0$ are respectively equivalent to $-L \geq 0$, $-L - 1 \geq 0$ and $L - 1 \geq 0$.

Due to the presence of disequations, it is not possible anymore to represent the set of solutions as $\mathcal{N} + \mathcal{H}^*$ where $\mathcal{N}$ and $\mathcal{H}$ are finite sets. For instance, the solutions of $x - y \neq 0$ do not have such a representation. However, it is always possible to represent in this way the solutions of a problem containing only equations and inequations. Now a disequation $\alpha X + \beta \neq 0$ is clearly equivalent to the disjunction $\alpha X + \beta > 0 \vee \alpha X + \beta < 0$ so that a problem containing d disequations is equivalent to a disjunction of 2^d problems with pairwise disjoint sets of solutions which may be represented by means of two sets of minimal solutions. We do not want to introduce explicitly these disjunctions since it would lead to do most of the work several times. However, what our algorithm computes is a representation of each of these 2^d sets. Only, we handle directly the disequations and *dispatch* the solutions afterwards.

4.1 Parametric representation of the solutions

We first give the new set of transformation rules and then prove that we indeed get the representation we seek. We have to be careful about disequations when we adapt the condition of R_3. This rule is intended to remove a satisfied constraint. As already said, the disequation $L \neq 0$ is actually meant as the disjunction $L > 0 \vee L < 0$. We want to remove it only when either $L > 0$ or $L < 0$ is satisfied for all values of the parameters. For instance, the trivially satisfied disequation $2u - 1 \neq 0$ must not be removed since neither $2u - 1 < 0$ nor $2u - 1 > 0$ is satisfied.

We also have to modify slightly R_4 in order to be able to solve disequations of the form $\alpha U \neq 0$ where $\alpha \neq 0$ and either $\alpha \geq 0$ or $\alpha \leq 0$. The criterion we used until now, consisted in always choosing coefficients with opposite signs. In this case, this is not possible anymore although the disequation is neither satisfied nor unsatisfiable. We apply then R_4 by choosing any parameter u_i with a nonzero coefficient. This makes the disequation satisfied. This case is handled by the rule R'_4 in the table below which gives the modified set of rules.

(R_1) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow u_i + u_j] \vee (P \parallel \mathcal{C})[u_j \leftarrow u_j + u_i]$
if $\mathcal{C} \equiv \alpha U + \beta \# 0 \wedge \mathcal{C}'$, $\alpha_i\alpha_j < 0$, $\forall p, q,\ \alpha_i\alpha_j \leq \alpha_p\alpha_q$, $\forall p,\ \alpha_i\alpha_j \leq \alpha_p\beta$

(R_2) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0]$
if $\mathcal{C} \equiv \alpha U = 0 \wedge \mathcal{C}'$, $\alpha_i \neq 0$ and either $\alpha \geq 0$ or $\alpha \leq 0$
or $\mathcal{C} \equiv \alpha U \geq 0 \wedge \mathcal{C}'$, $\alpha_i < 0$ and $\alpha \leq 0$

(R_3) $P \parallel c \wedge \mathcal{C}' \quad \vdash \quad P \parallel \mathcal{C}'$
if $c \equiv 0 = 0$
or $c \equiv \alpha U + \beta \geq 0$ and $\alpha \geq 0$ and $\beta \geq 0$
or $c \equiv \alpha U + \beta \neq 0$ and either $\alpha \geq 0$ and $\beta > 0$ or $\alpha \leq 0$ and $\beta < 0$

(R_4) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0] \vee (P \parallel \mathcal{C})[u_i \leftarrow u_i + 1]$
if $\mathcal{C} \equiv \alpha U + \beta \# 0 \wedge \mathcal{C}'$, $\alpha_i\beta < 0$, $\forall p, q,\ \alpha_i\beta \leq \alpha_p\alpha_q$, $\forall p,\ \alpha_i\beta \leq \alpha_p\beta$

(R'_4) $P \parallel \mathcal{C} \quad \vdash \quad (P \parallel \mathcal{C})[u_i \leftarrow 0] \vee (P \parallel \mathcal{C})[u_i \leftarrow u_i + 1]$
if $\mathcal{C} \equiv \alpha U \neq 0 \wedge \mathcal{C}'$, $\alpha_i \neq 0$ and either $\alpha \geq 0$ or $\alpha \leq 0$

(R_5) $P \parallel \mathcal{C} \quad \vdash \quad \perp$
if $\mathcal{C} \equiv \alpha U + \beta = 0 \wedge \mathcal{C}'$ and either $\alpha \geq 0$ and $\beta > 0$ or $\alpha \leq 0$ and $\beta < 0$
or $\mathcal{C} \equiv \alpha U + \beta \geq 0 \wedge \mathcal{C}'$, $\alpha \leq 0$ and $\beta < 0$,
or $\mathcal{C} \equiv 0 \neq 0 \wedge \mathcal{C}'$

Theorem 5. *The application of the transformation rules R_1, R_2, R_3, R_4, R'_4 and R_5 to $X = U \parallel \mathcal{A}U + \mathcal{B} \# 0$ terminates and returns solved forms.*

Proof. The proof is again similar to the proof of theorem 1. We just have to be a bit careful about the rule R'_4. In the second alternative of this rule, none of the complexities considered by Elliott decreases. But after this rule, only R_3 may apply to $(P \parallel \mathcal{C})[u_i \leftarrow u_i + 1]$, which removes the first constraint and clearly reduces the problem. □

Example 3. *Let us consider the problem*

$$\mathcal{P} \equiv \begin{cases} 2x_1 + x_2 - 3 \geq 0 \\ x_1 - 2x_2 + 1 \neq 0 \end{cases}$$

Figure 2 shows all applications of the transformation rules in a tree form. Each node $X = MU + D \parallel AU + B \# 0$ *is represented in matrix form as* $\begin{array}{c|c} A & B \,\# \\ \hline M & D \end{array}$. *Struck out constraints are the ones removed by* R_3, *crossed leaves are the ones that are deleted by* R_5 *and framed leaves are solved forms.*

4.2 Partition of the set of solutions

In the remaining of this section, $\mathcal{P} \equiv \mathcal{A}X + \mathcal{B} \# 0$ is a problem containing d disequations. When the transformation process stops, we are left with a finite disjunction of solved forms $\mathcal{F}_1, \ldots, \mathcal{F}_N$ which is equivalent to $\mathcal{P}$. Each $\mathcal{F}_i$ has the form $X = S_{i,1}u_1 + \cdots + S_{i,n_i}u_{n_i} + D_i$. As already said, $\mathcal{P}$ is equivalent to a disjunction of 2^d problems obtained by replacing each disequation in $\mathcal{P}$ with either $>$ or $<$. Given a tuple of predicates $\Delta = [\delta_1, \ldots, \delta_d] \in \{>, <\}^d$, $\mathcal{P}_\Delta$ denotes the problem obtained by replacing $\neq$ with δ_i in the i^{th} disequation of $\mathcal{P}$. Let $\mathcal{S}_\Delta$ denote the set of solutions of $\mathcal{P}_\Delta$. We show that from $\mathcal{F}_1, \ldots, \mathcal{F}_N$, we get a representation of each $\mathcal{S}_\Delta$. For this, we first need a lemma.

Lemma 6. *Let* $\alpha X + \beta \neq 0$ *be a disequation in* $\mathcal{P}$. *For each* $i = 1, \ldots, N$, *either* $\alpha D_i + \beta > 0$ *and* $\forall k = 1, \ldots, n_i$, $\alpha S_{i,k} \geq 0$ *or* $\alpha D_i + \beta < 0$ *and* $\forall k = 1, \ldots, n_i$, $\alpha S_{i,k} \leq 0$.

Proof. The property is preserved by each transformation rule. That is to say, let $P \equiv X = M^1u_1 + \cdots + M^r u_r + D \parallel \mathcal{C}$ be such that either $\alpha D + \beta > 0$ and $\forall k$, $\alpha M^k \geq 0$ or $\alpha D + \beta < 0$ and $\forall k$, $\alpha M^k \leq 0$. Then for any expression $X = M'^1 u_1 + \cdots + M'^{r'} u_{r'} + D' \parallel \mathcal{C}'$ obtained by applying one of the rules to P, we have either $\alpha D' + \beta > 0$ and $\forall k$, $\alpha M'^k \geq 0$ or $\alpha D' < 0$ and $\forall k$, $\alpha M'^k \leq 0$. Now from the condition of the rule R_3, the property has to hold when the disequation is removed from the constraints. So that it still holds in the end. □

This lemma implies that for each $\mathcal{F}_i$, if D_i is a solution of $\mathcal{P}_\Delta$, then any solution of $\mathcal{F}_i$ is a solution of $\mathcal{P}_\Delta$. Note that testing whether D_i is a solution of $\mathcal{P}_\Delta$ is straightforward. Since we already know that D_i is a solution of $\mathcal{P}$, we just have to look at the sign of $\alpha D_i + \beta$ for each disequation $\alpha D_i + \beta \neq 0$ in $\mathcal{P}$. We get then the following theorem.

Theorem 7. *For each* $\Delta \in \{>, <\}^d$,

1. $\mathcal{P}_\Delta \iff \bigvee_{D_i \in \mathcal{S}_\Delta} \mathcal{F}_i$.
2. *The set* $\mathcal{N}_\Delta = \{D_i \mid D_i \in S_\Delta\}$ *contains all minimal solutions of* $\mathcal{P}_\Delta$.
3. *If* $\mathcal{P}_\Delta$ *has a solution, the set* $\mathcal{H}_\Delta = \bigcup_{D_i \in \mathcal{S}_\Delta} \{S_{i,1}, \ldots, S_{i,n_i}\}$ *contains all minimal nonzero solutions of its homogeneous part.*

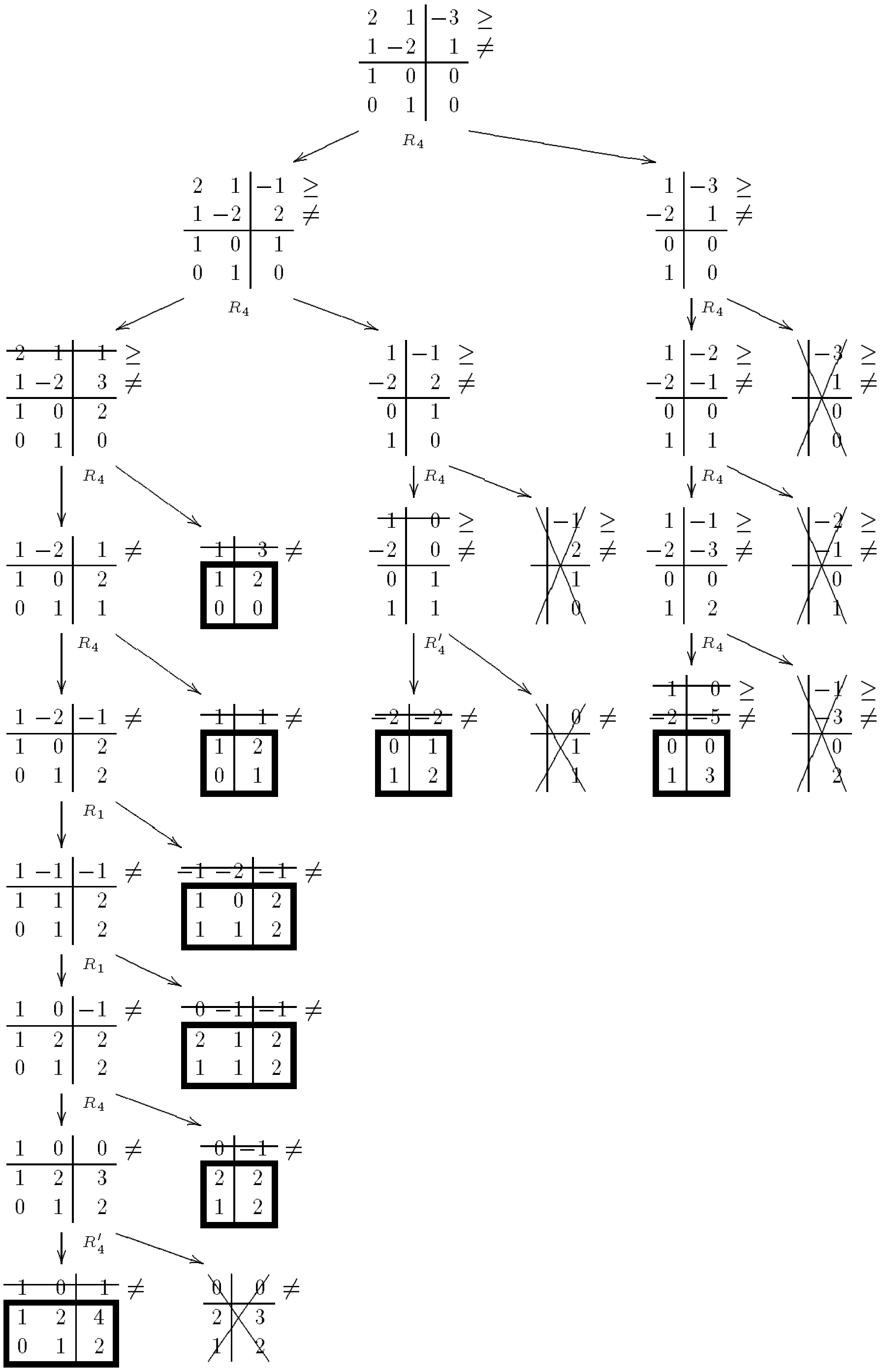

Fig. 2. solving $2x_1 + x_2 - 3 \geq 0 \wedge x_1 - 2x_2 + 1 \neq 0$

Proof. From the lemma, we immediately get $\mathcal{P}_\Delta \Leftarrow \vee_{D_i \in \mathcal{S}_\Delta} \mathcal{F}_i$. For the converse, let S be a solution of $\mathcal{P}_\Delta$. S is then a solution of $\mathcal{P}$ and must be solution of some $\mathcal{F}_i$. Now D_i belongs to some $\mathcal{S}_{\Delta'}$ but again from the lemma, all solutions of $\mathcal{F}_i$ must then belong to $\mathcal{S}_{\Delta'}$, including S. Since $\mathcal{S}_\Delta$'s are pairwise disjoint, we must have $\Delta' = \Delta$, which finishes the proof of (1). The proof of (2) and (3) is similar to the proof of theorem 4. □

Example 4 (example 3 continued). *Since we have one disequation, we have two sets of solutions, $S_>$ and $S_<$. Collecting the solutions in the solved forms, we get*

$$\mathcal{N}_> = \{(2,0),(2,1),(4,2)\} \qquad \mathcal{H}_> = \{(1,0),(2,1)\}$$
$$\mathcal{N}_< = \{(2,2),(1,2),(0,3)\} \qquad \mathcal{H}_< = \{(2,1),(1,1),(0,1)\}$$

$(4,2)$ *is a non-minimal solution of $\mathcal{P}_>$ so that we may remove it from $\mathcal{N}_>$. Other solutions are minimal so that we get*

$$\mathcal{S}_> = \{(2,0),(2,1)\} + \{(1,0),(2,1)\}^*$$
$$\mathcal{S}_< = \{(2,2),(1,2),(0,3)\} + \{(2,1),(1,1),(0,1)\}^*$$

4.3 Testing for minimality

From what precedes, we get for each $\mathcal{S}_\Delta$ a representation as $\mathcal{N}_\Delta + \mathcal{H}_\Delta^*$. However, both $\mathcal{N}_\Delta$ and $\mathcal{H}_\Delta$ may contain elements which are not minimal. We may filter such non-minimal elements and remove them without affecting $\mathcal{S}_\Delta$. For this, we must extend each solution S by adding a slack variable for each constraint which is not an equation. We show that this may be done in a way which does not depend on Δ. It turns out to be useful especially for the solutions of the homogeneous parts. Indeed, although $\mathcal{S}_\Delta$ and $\mathcal{S}_{\Delta'}$ are disjoint if $\Delta \neq \Delta'$, it may happen that $\mathcal{H}_\Delta$ are $\mathcal{H}_{\Delta'}$ are not disjoint. In such a case, we have to compute the extension of a solution only once.

Let S be a solution of $\mathcal{P}_\Delta$. If the i^{th} non-equality constraint of $\mathcal{P}$ is an inequation $\alpha X + \beta \geq 0$, then the value of the i^{th} slack variable is simply $\alpha S + \beta$. When the i^{th} non-equality constraint is a disequation, things are not so simple anymore. The value of the slack variable depends on whether S is a solution of $\alpha X + \beta > 0$ or of $\alpha X + \beta < 0$. Let us first assume S is a solution of $\alpha X + \beta > 0 \iff \alpha X + \beta - 1 \geq 0$. The value of the slack variable is then $\alpha S + \beta - 1$. Now if S is a solution of $\alpha X + \beta < 0 \iff -\alpha X - \beta - 1 \geq 0$, the value of the slack variable is $-\alpha S - \beta - 1$. Thus in both cases, the value of the slack variable is $|\alpha S + \beta| - 1$. Now the offset 1 is subtracted for each solution so that we may omit it without affecting the comparisons. The value we take for the slack variable is then $|\alpha S + \beta|$. Note that this value is actually the same as the one we take for inequations. Now for an equation $\alpha X + \beta = 0$, the slack is 0 which is again $|\alpha S + \beta|$. The same holds for the homogeneous part so that we have:

Theorem 8.
- *S is a minimal solution of $\mathcal{P}_\Delta$ if and only if $(S, |\mathcal{A}S + \mathcal{B}|)$ is minimal in $\{(X, |\mathcal{A}X + \mathcal{B}|) \mid X \in \mathcal{N}_\Delta\}$.*
- *S is a minimal nonzero solution of the homogeneous part of $\mathcal{P}_\Delta$ if and only if $(S, |\mathcal{A}S|)$ is minimal in $\{(X, |\mathcal{A}X|) \mid X \in \mathcal{H}_\Delta\}$.*

5 Avoiding redundancy

As we noted before, the algorithm presented so far shows some redundancies that result from the same subproblem being solved several times. This is mainly due to the branching introduced by R_1 and R_4. Indeed, rule R_1 can be seen as a transformation of $P \parallel \mathcal{C}$ into $(P \parallel \mathcal{C} \wedge u_i \geq u_j) \vee (P \parallel \mathcal{C} \wedge u_i \leq u_j)$, which results in $(P \parallel \mathcal{C} \wedge u_j = u_i)$ being solved twice. If the algorithm is not to be implemented in parallel this feature becomes clearly a drawback. The first idea which comes in mind to overcome this problem is to replace this branching with $(P \parallel \mathcal{C} \wedge u_i \geq u_j) \vee (P \parallel \mathcal{C} \wedge u_i < u_j)$. But as long as we explicitely branch, we do not avoid performing twice a combination which may be needed in both branches. The changes we propose to avoid explicit branching are based on the equivalence between $X = MU \parallel AU = 0$ and

$$X = (M^i + M^j)u + MU \parallel (A^i + A^j)u + AU = 0 \wedge (u_i = 0 \vee u_j = 0)$$

If this equivalence is used as a transformation rule, the constraint $u_i = 0 \vee u_j = 0$ implies that (u_i, u_j) should not be selected afterwards. If it was, the parameter associated to the new $(A^i + A^j)$, say v, would be constrained by $v = 0$ and hence of no use to the solution. Moreover, the algorithm would not terminate.

These constraints, that *forbid some combinations*, propagate in a quite easy way. If u_i is rewritten as $u_i = u'_i + u'$ then $u_i = 0 \vee u_j = 0$ propagates as $(u'_i = 0 \vee u_j = 0) \wedge (u' = 0 \vee u_j = 0)$ since all the parameters are nonnegative integers. In general, if u_p and u_q are selected and $(u_p = 0 \vee K_p) \wedge (u_q = 0 \vee K_q)$ must hold, then for the new parameterisation, say $u_p = u'_p + u$ and $u_q = u'_q + u$, it must be $(u'_p = 0 \vee K_p) \wedge (u'_q = 0 \vee K_q) \wedge (u = 0 \vee (K_p \wedge K_q))$, and additionally $u'_q = 0 \vee u'_p = 0$. We consider expressions of the form $P \parallel \mathcal{C} \parallel K$ where K denotes the additional constraints.

The criterion for selecting the two parameters to be replaced is slightly modified so that (u_i, u_j) is not chosen if $u_i = 0 \vee u_j = 0$ is asserted. $X = MU \parallel AU = 0 \parallel K$ rewrites as

$$X = (M^i + M^j)u + MU \parallel (A^i + A^j)u + AU = 0 \parallel K' \wedge (u_i = 0 \vee u_j = 0)$$

if $u_i \neq 0 \wedge u_j \neq 0$ is satisfiable under K, and $0 > \alpha_i\alpha_j = \min\{\alpha_p\alpha_q \mid u_p \neq 0 \wedge u_q \neq 0 \text{ satisfiable under } K\}$. Here, K' denotes the propagation of the forbidden combinations K, that is $K[u_i \leftarrow u_i + u, u_j \leftarrow u_j + u]$. It is not difficult to conclude that at each step, either

$$\min\{\alpha_p\alpha_q \mid u_p \neq 0 \wedge u_q \neq 0 \text{ satisfiable under } K\}$$

increases, or the number of (u_p, u_q) such that $u_p \neq 0 \wedge u_q \neq 0$ is satisfiable under K and $\alpha_p\alpha_q$ is minimum decreases. This implies termination.

Now we are going to explain how this basic idea is adapted to the general case. When solving non-homogeneous problems, say $AU + B \# 0$, redundancy is also due to R_4. Since R_4 involves B which had no associated parameter, we could not express the subsequent forbidden combinations as before. The trick is to associate a parameter to B, say t, representing the initial problem by

$$X = U + 0t \parallel \mathcal{A}U + \mathcal{B}t \# 0 \parallel t = 1$$

Then, the effect of R_4 can be seen as rewriting $X = MU + NT \parallel AU + BT \# 0 \parallel K' \wedge \sum t_i = 1$ into

$$X = MU + NT + (M^i + N^j)t \parallel AU + BT + (A^i + B^j)t \# 0 \parallel K'' \wedge \sum t_i + t = 1$$

where $K'' = K'[u_i \leftarrow u_i + t, t_j \leftarrow t_j + t] \wedge (u_i = 0 \vee t_j = 0)$. From $\sum t_i = 1$ it follows that $t_j = 0 \vee t_k = 0$, for all j and k, and thus a selection of two t's must not be eligible. If $K' \wedge \sum t_i = 1$ is denoted just as K it becomes apparent that R_1 and R_4 may now be merged in a single rule. Also, R_4' could be merged in the same scheme.

Let us give an example to illustrate what we have been describing. The following tables show schematically how the rules would operate on $3x - 2y - 1 = 0$. The double vertical lines make the separation between A and B, and between M and N. Below the last horizontal line we have represented the products between pairs of coefficients, and between each coefficient and b.

When u_i and u_j are selected, $\alpha_i\alpha_j$ is negative and minimum. A new column is introduced which is the sum of the columns associated to u_i and u_j. A new row is also introduced for we are not taking symmetry into account. The two entries $\alpha_i\alpha_j$ are crossed, indicating that the pair is no longer eligible. So, the table on the right results from the selection of u_1 and u_2. The one on the left was the starting table.

Coefs	3	−2	−1
Tuples	1	0	0
	0	1	0
Prods	9	−6	−3
	−6	4	2

Coefs	3	−2	1	−1
Tuples	1	0	1	0
	0	1	1	0
Prods	9	×	3	−3
	×	4	−2	2
	3	−2	1	−1

Crosses propagate: the result of adding something to a cross is a cross. When the selection is u_i and t_k, the sum of the columns is placed on the right-hand side and we do not introduce a new row. Just one entry has to be crossed. Since -3 is the least product, the next selection is u_1 and t_1, resulting the table below on the left. When no negative product occurs, the rules described so far do not apply, and the table is as shown on the right.

3	−2	1	−1	2
1	0	1	0	1
0	1	1	0	0
9	×	3	×	6
×	4	−2	2	×
3	−2	1	−1	2

3	−2	1	−1	0	−1	2	0
1	0	1	1	2	0	1	1
0	1	1	2	3	0	0	1
9	×	3	×	×	×	6	×
×	4	×	2	×	2	×	×
3	×	1	×	0	×	2	0
×	2	×	1	0	1	×	×
×	×	0	0	0	0	×	0

The equivalent problem the last table encodes can be denoted as before

$$X = \begin{bmatrix} 1 & 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} U + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} T$$

$$\| \begin{bmatrix} 3 & -2 & 1 & -1 & 0 \end{bmatrix} U + \begin{bmatrix} -1 & 2 & 0 \end{bmatrix} T = 0$$

$$\| \sum t_i = 1 \ \wedge \ \begin{cases} u_1 = 0 \vee u_2 = u_4 = u_5 = t_1 = t_3 = 0 \\ u_2 = 0 \vee u_1 = u_3 = u_5 = t_2 = t_3 = 0 \\ u_3 = 0 \vee u_2 = u_4 = t_1 = 0 \\ u_4 = 0 \vee u_1 = u_3 = t_2 = t_3 = 0 \\ u_5 = 0 \vee u_1 = u_2 = t_2 = 0 \end{cases}$$

and the constraint part may be viewed as the disjunction

$$\begin{aligned} &(3u_1 + u_3 + 2 = 0 \| t_2 = 1 \wedge u_2 = u_4 = u_5 = t_1 = t_3 = 0) \\ \vee\, &(-2u_2 - u_4 - 1 = 0 \| t_1 = 1 \wedge u_1 = u_3 = u_5 = t_2 = t_3 = 0) \\ \vee\, &(u_3 = 0 \| t_3 = 1 \wedge u_1 = u_2 = u_4 = t_1 = t_2 = 0) \\ \vee\, &(-u_4 - 1 = 0 \| t_1 = 1 \wedge u_1 = u_2 = u_3 = t_2 = t_3 = 0) \end{aligned}$$

It can be noted that only the third equation is satisfiable. Hence, the solution to the original problem is

$$X = \begin{bmatrix} 2 \\ 3 \end{bmatrix} u_5 + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The transformation rules that are then applied to get solved forms translate somehow the former R_2, R_3 and R_5. In order to describe them more easily, let L^i_α, L^i_β and R^i_α denote the sets

$$\begin{aligned} L^i_\alpha &= \{u_q \mid u_q \neq 0 \wedge u_i \neq 0 \text{ satisfiable under } K\} \\ L^i_\beta &= \{t_r \mid t_r \neq 0 \wedge u_i \neq 0 \text{ satisfiable under } K\} \\ R^i_\alpha &= \{u_q \mid u_q \neq 0 \wedge t_i \neq 0 \text{ satisfiable under } K\} \end{aligned}$$

Given a constraint $c \wedge \mathcal{C}' \| K$ where $c \equiv \alpha U + \beta T \,\#\, 0$,

- $u_i \leftarrow 0$ applies if either $L^i_\beta = \varnothing$ or $\alpha_i \neq 0$, $\neg(\alpha_i \,\#\, 0)$, $\forall u_q \in L^i_\alpha$, $\alpha_i \alpha_q \geq 0$ and $\forall t_r \in L^i_\beta$, $\alpha_i \beta_r \geq 0$,
- $t_i \leftarrow 0$ applies if $\neg(\beta_i \,\#\, 0)$ and either
  - $\# \in \{=, \geq\}$ and $\forall u_q \in R^i_\alpha$, $\alpha_q \beta_i \geq 0$
  - or $\# \equiv \neq$ and $\forall u_q \in R^i_\alpha$, $\alpha_q = 0$,
- $c \wedge \mathcal{C}' \| K$ is reduced to $\mathcal{C}' \| K$ if $\forall i$, $\beta_i \,\#\, 0 \wedge \forall u_q \in R^i_\alpha$, $\alpha_q \beta_i \geq 0$.

Finally, $P \| \mathcal{C} \| K \vdash \perp$ if K is unsatisfiable, which happens when all t_i's have been set to 0.

Let us see how these rules apply to the example used in the previous section:

$$\begin{cases} 2x + y - 3 \geq 0 \\ x - 2y + 1 \neq 0 \end{cases}$$

If just the first constraint is represented, at the starting point we have the table on the left. After the second transformation the one in the middle, and then $t_1 \leftarrow 0$ applies.

2	1	-3	$\geq$
1	0	0	
0	1	0	
4	2	-6	
2	1	-3	

2	1	-3	-1	-2	$\geq$
1	0	0	1	0	
0	1	0	0	1	
4	2	$\times$	-2	$\times$	
2	1	$\times$	-1	-2	

2	1	-1	-2	$\geq$
1	0	1	0	
0	1	0	1	
4	2	-2	$\times$	
2	1	-1	-2	

The following tables are respectively the last for $\geq$ and the starting one for $\neq$. Because K is kept when $c \wedge \mathcal{C}' \parallel K$ is rewritten as $\mathcal{C}' \parallel K$, the crosses are kept.

2	1	1	0	0	$\geq$
1	0	2	1	0	
0	1	0	1	3	
4	2	2	$\times$	$\times$	
2	1	1	0	0	

1	-2	3	0	-5	$\neq$
1	0	2	1	0	
0	1	0	1	3	
1	-2	3	$\times$	$\times$	
-2	4	-6	0	10	

After some steps we get the table displayed below on the left, which has no negative products. However, since a disequation is being solved, R'_4 applies twice, and then the substitutions $t_2 \leftarrow 0$ and $t_6 \leftarrow 0$, yielding the table on the right.

u_1	u_2	u_3	u_4	t_1	t_2	t_3	t_4	t_5	t_6	
1	-2	-1	0	3	0	-5	1	-1	0	$\neq$
1	0	1	2	2	1	1	2	2	3	
0	1	1	1	0	1	3	1	2	2	
1	$\times$	$\times$	0	3	$\times$	$\times$	1	$\times$	$\times$	
$\times$	4	2	$\times$	$\times$	0	10	$\times$	2	$\times$	
$\times$	2	1	0	$\times$	$\times$	$\times$	$\times$	1	0	
0	$\times$	0	0	$\times$	$\times$	$\times$	0	$\times$	0	

u_1	u_2	u_3	u_4	t_1	t_3	t_4	t_5	t_7	t_8	
1	-2	-1	0	3	-5	1	-1	-2	-1	$\neq$
1	0	1	2	2	1	2	2	2	4	
0	1	1	1	0	3	1	2	1	3	
1	$\times$	$\times$	0	3	$\times$	1	$\times$	$\times$	$\times$	
$\times$	4	2	$\times$	$\times$	10	$\times$	2	4	$\times$	
$\times$	2	1	0	$\times$	$\times$	$\times$	1	$\times$	1	
0	$\times$	0	0	$\times$	$\times$	0	$\times$	$\times$	0	

The constraint may be removed. In order to get the minimal solutions we proceed as in the previous section.

An aspect is worth mentioning here. The method we have proposed avoids redundancy at the cost of increasing the space complexity of the algorithm. This may become quite critical since the number of minimal solutions may be quite large, and the method solves one constraint at a time. This implies that when the i^{th} constraint is removed there are at least as many parameters as the minimal solutions of the subsystem solved so far. We say *at least* because actually there may be several non-minimal solutions which may remain until the whole system is solved. Partial solutions that are not minimal when compared with other partial solutions cannot be simply removed due to the way forbidden combinations are propagated to the next combinations.

On the other hand, although the use of pruning by solutions is an obvious improvement, it is not enough to make the Elliott-MacMahon algorithm competitive with other algorithms [8]. The major cause of lack of efficiency is inherent to constraints being solved one by one.

Conclusions

We have presented algorithms that solve systems of linear diophantine equations, inequations and disequations, yielding a parametric representation of the solution set from which it is possible to get immediately the minimal solutions. The main algorithm, which is inspired by the Elliott-MacMahon algorithm, is actually a schema parametrized by a choice criterion.

The efficiency of the particular instances is highly determined by the choice criterion involved. In particular, the ones described herein require solving the constraints one at a time, which is known not to be efficient. It is worth studying criteria that consider several constraints at a time. The algorithm could then be competitive.

Finally, it can be remarked that the algorithms allow the set of constraints to be changed incrementally. Moreover, they can be easily put together with other modules to perform satisfiability tests or efficient solvers for some kind of subproblems.

References

1. Habib Abdulrab and Marianne Maksimenko. General solution of systems of linear diophantine equations and inequations. In Jieh Hsiang, editor, *Proceedings 6th Conference on Rewriting Techniques and Applications, Kaiserslautern (Germany)*, volume 914 of *Lecture Notes in Computer Science*, pages 339–351. Springer-Verlag, April 1995.
2. Farid Ajili and Evelyne Contejean. Complete solving of linear diophantine equations and inequations without adding variables. In Ugo Montanari and Francesca Rossi, editors, *Proceedings 1st International Conference on Principles and Practice of Constraint Programming, Cassis (France)*, volume 976 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, September 1995.
3. A. Boudet, E. Contejean, and H. Devie. A new AC unification algorithm with a new algorithm for solving diophantine equations. In *Proceedings 5th IEEE Symposium on Logic in Computer Science, Philadelphia (Pa., USA)*, pages 289–299, June 1990.
4. M. Clausen and A. Fortenbacher. Efficient solution of linear diophantine equations. *Journal of Symbolic Computation*, 8(1 & 2):201–216, 1989. Special issue on unification. Part two.
5. E. Domenjoud. Solving systems of linear diophantine equations: An algebraic approach. In A. Tarlecki, editor, *Proceedings 16th International Symposium on Mathematical Foundations of Computer Science, Kazimierz Dolny (Poland)*, volume 520 of *Lecture Notes in Computer Science*, pages 141–150. Springer-Verlag, September 1991.
6. E. B. Elliott. On linear homogeneous diophantine equations. *Quartely J. of Pure and Applied Maths*, 136, 1903.
7. M. Filgueiras and A. P. Tomás. Fast methods for solving linear diophantine equations. In M. Filgueiras and L. Damas, editors, *Proceedings of the 6th Portuguese Conference on AI, Porto (Portugal)*, volume 727 of *Lecture Notes in Artificial Intelligence*, pages 297–306. Springer-Verlag, 1993.

8. Miguel Filgueiras and Ana Paula Tomás. A note on the implementation of the MacMahon-Elliott algorithm. Technical report, Centro de Informática da Universidade do Porto, 1992.
9. G. Huet. An algorithm to generate the basis of solutions to homogenous linear diophantine equations. *Information Processing Letters*, 7(3):144–147, 1978.
10. P. A. MacMahon. *Combinatory Analysis*, volume 2, chapter II: A Syzygetic Theory, pages 111–114. Cambridge University Press, 1916. Reprinted by Chelsea, New York, 1960.
11. L. Pottier. Minimal solutions of linear diophantine systems: Bounds and algorithms. In R. V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications, Como (Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173. Springer-Verlag, April 1991.