Using Linear and Semi–Definite Programming for Computing Models of Disjunctive Deductive Databases

F. Jarre, R. Lepenis, D. Seipel

University of Würzburg
Am Hubland, D - 97074 Würzburg
seipel@informatik.uni-wuerzburg.de

Abstract

Efficient algorithms for computing the minimal and stable models are needed to attract practitioners attention to *disjunctive deductive databases*. In 1994 Bell et al. [4] have examined the problem of computing these models based on methods of mathematical optimization, more precisely, using the *simplex method* and *cutting–plane methods*.

We have followed up on the idea of using techniques from mathematical optimization and developed algorithms based on more recently proposed optimization methods – so called *interior point methods* – and on suitable heuristics.

Our algorithms can be used for computing *cost–minimal* models in addition to the common *minimal* models *w.r.t. set inclusion*, and also for solving programs with more complex constraints than linear constraints. E.g. we used quadratic constraints for modeling 0-1-conditions, which leads to a better approximation of the underlying problem.

1 Introduction

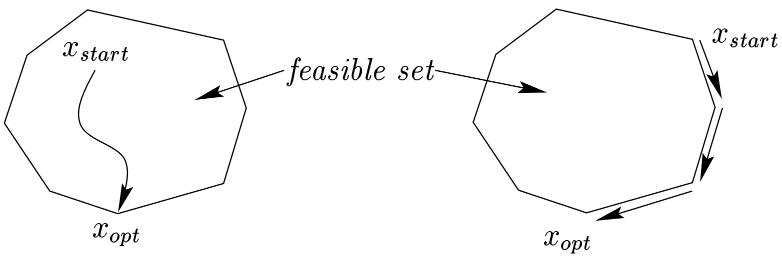
As a marriage of disjunctive logic programming and database technologies, disjunctive deductive databases can form the foundation for integrated high–end reasoning systems. Advanced applications in which incomplete information must be extensively analyzed are likely to benefit from this technology.

It is possible to compute the minimal models of a disjunctive deductive database based on solving *integer programs*, cf. [4]. However, solving these programs in general is very expensive. Linear programs without the integer constraints can be solved more efficiently. Therefore, a classical way of solving integer programs is to *approximate* the integer program by a sequence of *linear programs* over real numbers. More precisely, the 0-1-conditions $x_i \in \{0,1\}$ are replaced by the interval conditions $x_i \in [0,1]$. The resulting linear program is solved, and from the solution a new constraint is constructed that restricts the set of solutions of the linear program towards the set of solutions of the 0-1-program. Then, the new linear program is solved in the same manner. This process is repeated until an integer solution is found.

We have used different linear and *semi–definite* approximations for 0-1-program, and we have applied appropriate iterative heuristics to obtain 0-1 solutions from the solutions of the semi–definite programs. The advantage in using semi–definite programs is that – compared to linear programs – they lead to better relaxations of 0-1-programs and therefore to a smaller number of steps to find a 0-1-solution. The disadvantage is that they are more expensive.

The linear and semi–definite programs are solved using *interior point methods* while Bell et al. used the well–known *simplex method*. As the simplex method uses the combinatorial nature of linear problems – it follows the edges of the polyhedron containing the *feasible points* of the linear program – it has an *exponential worst–case time complexity* for solving a linear program. Interior point methods follow a sequence of *interior points*, i.e. points fulfilling strictly the inequalities of the optimization problem, towards an optimal solution of the problem. These methods have a *polynomial time complexity* for a number of optimization problems including linear and semi–definite problems.[1] In the recent past, serious efforts have been undertaken to develop a well founded theory for this area and to construct efficient methods. For an introduction into interior point methods especially for linear and semi–definite programming we refer to [2], [7], [13] and [14].[2]

The following figure gives an impression of the *computational behaviour* of the interior point (to the left) and the simplex method (to the right), respectively:



The organization of the paper is as follows: Section 2 contains some fundamental definitions from disjuncive deductive databases. In Section 3 we recapitulate the approach for transforming a disjunctive deductive database into a set of linear restrictions given by Bell et al. [4]. In Section 4 the mathematical methods we have developed for computing minimal models are described. In Section 5 we give a conclusion.

2 Basic Definitions and Notations

Disjunctive Deductive Databases

A *disjunctive rule* is given by a formula

$$\alpha \leftarrow \beta \wedge not \cdot \gamma,$$

where $\alpha = A_1 \vee ... \vee A_k$, $\beta = B_1 \wedge ... \wedge B_m$ and $not \cdot \gamma = not\ C_1 \wedge ... \wedge not\ C_n$ with atoms $A_1, \ldots, A_k, B_1, \ldots, B_m$ and $C_1, \ldots, C_n$, $k \geq 1$ and $n, m \geq 0$. A disjunctive rule with $n = m = 0$ is called a disjunctive fact. A *disjunctive deductive database* P[3] is a finite set of disjunctive rules. The *ground version* of P, denoted by *gnd* (P), is the set of all ground instances of the rules of P.

For example, let G be the graph in the figure below. The meaning of the dotted lines is that there exists an edge from a to b or there exists an edge from a to c, i.e. we have

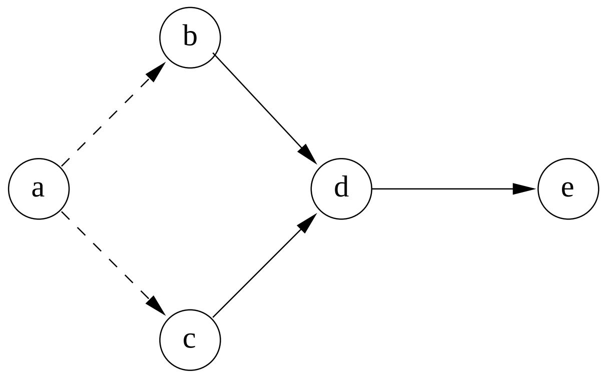
[1]Notice, however, that as we are solving a sequence of subproblems this does of course not mean that the overall complexity of computing the minimal models of a disjunctive deductive database is polynomial.

[2]Some further literature: [9], [10] and [12].

[3]also called *disjunctive logic program*

only an uncertain knowledge about the world. The graph G can be described using the set of disjunctive facts $S = \{ \ arc(a,b) \lor arc(a,c), \ arc(b,d), \ arc(c,d), \ arc(d,e) \ \}$ and the set of disjunctive rules

$$P = \{ \ path(\mathrm{X},\mathrm{Z}) \leftarrow arc(\mathrm{X},\mathrm{Z}), \ path(\mathrm{X},\mathrm{Z}) \leftarrow arc(\mathrm{X},\mathrm{Y}) \land path(\mathrm{Y},\mathrm{Z}) \ \}.$$



Minimal and Stable Models

Let P be a disjunctive deductive database. An Herbrand interpretation I is given by a subset $I \subseteq HB_P$ of the Herbrand base HB_P of P. I is an *Herbrand model* of P if for all ground instances $r = (\alpha \leftarrow \beta \land not \cdot \gamma) \in gnd\,(P)$ of rules in P we have: $I \models \beta \land not \cdot \gamma \ \Rightarrow \ I \models \alpha$.

An Herbrand model I of P is *minimal* (w.r.t. set inclusion), if there exists no other Herbrand model I' of P with $I' \subsetneq I$. The set of all minimal models of P is denoted by $\mathcal{MM}_P$. An Herbrand model I of P is called *cardinality–minimal* if the cardinality of every model of P is greater than or equal to the cardinality of M. Obviously a cardinality–minimal model is a minimal model.

Given an Herbrand interpretation $I \subseteq HB_P$ of P, the *Gelfond–Lifschitz transformation* P^I of P with respect to I contains all rules $\alpha \ \leftarrow \ \beta$ such that there exists a ground rule $r = (\alpha \ \leftarrow \ \beta \land not \cdot \gamma) \in gnd\,(P)$ such that $I \models not \cdot \gamma$. An Herbrand model I of P is called a *stable model* if I is a minimal model of P^I. It can easily be shown that all stable models of a disjunctive deductive database P are also minimal models of P.

For example, the database $P' = S \cup P$ has got two minimal models

$$I_1 = I \cup \{ \, arc(a,b), \, path(a,b) \, \} \ \text{ and } \ I_2 = I \cup \{ \, arc(a,c), \, path(a,c) \, \},$$

with the common part $I = \{ \ arc(b,d), \ arc(c,d), \ arc(d,e), \ path(a,d), \ path(a,e), \ path(b,d), \ path(b,e), \ path(c,d), \ path(c,e), \ path(d,e) \ \}$.

3 Transformation into Linear Restrictions

In the following we recapitulate the approach for transforming a disjunctive deductive database P into a set of linear restrictions given by Bell et al. [4], and the problem of computing the minimal models of P into the problem of solving a number of linear 0-1-programs. We will use linear programs in the form

$$\begin{array}{lrl} (L) & \min & c^T \cdot x \\ & \text{subject to} & a_i^T \cdot x \geq b_i, \ \text{for } i = 1, \ldots, m \\ & & x \geq 0 \end{array}$$

with vectors $c, x, a_1, \ldots, a_m \in I\!R^n$ and reals $b_1, \ldots, b_m \in I\!R$. The superscript "^{T}" denotes the transposititon of a matrix (or vector). A (linear) program is called a (linear) integer program, if, in addition, all variables x_i are restricted to be integers. A (linear) integer program is called a (linear) 0-1-program, if $x_i \in \{0,1\}$; in this case the variables x_i are called binary variables.

To transform a ground disjunctive deductive database P into a linear 0-1-program, *binary variable representations* are used. For all atoms $A \in HB_P$, let x_A be a corresponding binary variable to A. The set $\{ x_A \mid A \in HB_P \}$ is called a binary variable representation of HB_P. We assign *linear constraints*

$$res(r) \; = \; (\sum_{i=1}^{k} x_{A_i} \; \geq \; 1 - \sum_{j=1}^{m} (1 - x_{B_j}) - \sum_{l=1}^{n} x_{C_l} \;)$$

to disjunctive ground instances

$$r = (A_1 \vee ... \vee A_k \leftarrow B_1 \wedge ... \wedge B_m \wedge not \; C_1 \wedge ... \wedge not \; C_n) \in gnd\,(P)$$

and sets of inequalities $res(P) = \{ res(r) \mid r \in grd(P) \}$ to the disjunctive deductive database P. To assign values to the variables of a binary variable representation of HB_P, a binary variable assignment is used, i.e. a mapping $S : \{x_A | A \in HB_P\} \rightarrow \{0,1\}$. For an interpretation I, the binary variable assignment S_I of I is given by $S_I(x_A) = 1$, if $A \in I$, and 0, otherwise.

The following theorem states the connection between the computation of the models of a disjunctive deductive database P and the corresponding set of inequalities $res(P)$:

Theorem 1 *Let P be a disjunctive database, I an interpretation and S_I the corresponding variable assignment. S_I is a solution of $res(P)$ iff I is an Herbrand model of P.*

For example, for the database

$$P = \{ \; a \vee b, \; c \leftarrow b \; \}$$

with the Herbrand base $HB_P = \{ a, b, c \}$ the minimal models of P are $M_1 = \{ a \}$ and $M_2 = \{ b, c \}$. The other models $M_3 = \{ a, c \}$ and $M_4 = \{ a, b, c \}$ are non–minimal. The corresponding set of inequalities is

$$\begin{array}{lll} x_a + x_b & & \geq 1 \\ \quad\; - \; x_b + x_c & & \geq 0. \end{array}$$

According to the form (L) we obtain $c^T = (\; 1 \; 1 \; 1 \;)$,

$$A = \begin{pmatrix} a_1^T \\ a_2^T \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad x = \begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix}.$$

Given 0-1-values of the variables x_a, x_b, x_c that solve both inequalities, we have a model according to Theorem 1. The model consists of those elements of the Herbrand base, where the corresponding variables x_* are 1. In our case $x_1 = (1,0,0)^T$, $x_2 = (0,1,1)^T$, $x_3 = (1,0,1)^T$ and $x_4 = (1,1,1)^T$ are the variable assignments that solve the inequalities. Obviously they correspond to the models M_1, M_2, M_3 and M_4.

Basic Idea for Computing Minimal Models

To compute a *cardinality–minimal model* of a disjunctive deductive database P, we can solve the linear 0-1-program

$$\begin{array}{ll} \min & \sum_{A \in HB_P} x_A \\ s.t. & res(P) \\ & x_A \in \{0,1\}. \end{array}$$

To compute *all minimal (w.r.t. set inclusion) models* of P, we work with an iterative algorithm: We solve the 0-1-program and obtain a cardinality–minimal model M. Then we enlarge the set of linear constraints of the 0-1-program by adding the inequality $\sum_{A \in M} x_A \leq k-1$, where k is the cardinality of the computed minimal model M. This inequality excludes the found minimal model and all models containing this model from the set of solutions of the original 0-1-program.[4] We solve the new 0-1-program, get another minimal model, exclude it from the set of solutions of the 0-1-program by adding a new inequality to the set of constraints, solve the next 0-1-program. By repeating these steps till the set of solutions is empty we obviously find all minimal models of a disjunctive deductive database P.

To compute *cost–minimal models* we can replace the cost–function $\sum_{A \in HB_P} x_A$ by an arbitrary (semi–definite) cost–function, e.g. the linear one $\sum_{A \in HB_P} c_A \cdot x_A$.

4 Methods from Mathematical Optimization

We get the following *framework for computing the minimal models* of a disjunctive deductive database P:

Algorithm 1

1. Let $\mathcal{MM}_P = \emptyset$ and $AR = \emptyset$, the set of added restrictions.
2. Use Algorithm 2 to minimize $\sum_{A \in HB_P} x_A$ subject to $res(P) \cup AR$.
3. If no optimal solution can be found, STOP.
   $\mathcal{MM}_P$ is the set of all minimal models of P.
4. Otherwise, let M be the model corresponding to the optimal solution found in step 2.
   Add M to $\mathcal{MM}_P$, i.e. let $\mathcal{MM}_P = \mathcal{MM}_P \cup M$.
5. Add the constraint $\sum_{A \in M} x_A \leq (k-1)$ to AR, where k is the cardinality of M.
   Go to step 2.

Algorithm 2

1. Approximate the 0-1-program IP using a linear or semi–definite program,
   and solve the latter using an interior point method.
2. If the solution from step 1 is a 0-1-solution, IP is solved.
3. Otherwise, according to an appropriate strategy add a new constraint to the problem, and go to step 1.

[4]Note that in the sum $\sum_{A \in M} x_A$ only the elements of the found minimal model are taken into account.

Semi–Definite Programs

Now let us have a look at semi–definite programs. Semi–definite programs are *convex* optimization problems. Their structure is similar to the structure of linear programs; semi–definite programming means linear programming using positive semi–definite variables. A semi–definite program is usually written in the form

$$\begin{array}{lll} (SDP) & \min & C \bullet X \\ & s.t. & A^{(i)} \bullet X = b_i, \text{ for } i = 1, \ldots, k \\ & & X \text{ positive semi–definite,} \end{array}$$

where C and the $A^{(i)}$ for $i = 1, \ldots, k$ are symmetric $(n \times n)$–matrices and $b_i \in I\!R$ for $i = 1, \ldots, k$. $X = (X_{i,j})$ is a positive semi–definite matrix, i.e. it is a symmetric matrix over real numbers, that meets the condition $v^T \cdot X \cdot v \geq 0$ for all $v \in I\!R^n$. E.g. the matrix $X = \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix}$ is positive semi–definite, since for all vectors $v = (\, v_1 \, v_2 \,)^T$ we have $v^T \cdot X \cdot v = 2 \cdot v_1^2 + 3 \cdot v_2^2 + (v_1 + v_2)^2 \geq 0$. The operator "$\bullet$" denotes the *inner product* of two $(n \times n)$–matrices $A = (A_{i,j})$ and $B = (B_{i,j})$: $A \bullet B = trace(A^T \cdot B) = \sum_{i,j=1}^n A_{i,j} \cdot B_{i,j}$.

It can be seen easily that a linear program is a specialization of a semi–definite program; if for a linear program (L') $\min\{\, c^T \cdot x \mid A \cdot x = b,\ x \geq 0 \,\}$ we set $C = Diag(c)$, $X = Diag(x)$ and $A^{(i)} = Diag(a_i)$ and require X to be positive semi–definite, we immediately obtain a semi–definite program (SDP) from the linear program (L). Here $Diag(y)$ denotes the diagonal–matrix induced by a vector y, i.e. the matrix having the elements of the vector y on its main diagonal and zeros everywhere else. So in the case of directly transforming a linear program into a semi-definite program, X has only variables along the diagonal. Conversely $diag(Y)$ (with lower case "d") denotes the vector y derived from a diagonal matrix Y. So $diag(Diag(y)) = y$ and $Diag(diag(Y)) = Y$.

Semi–Definite Relaxation of 0-1-Programs

There are a number of possibilities to approximate 0-1-programs – especially those derived from disjunctive deductive databases – with semi–definite programs. We give a *naive* relaxation as an example.[5] First we have to transform the 0-1-program into the equivalent program:

$$\begin{array}{lll} (L'') & \min & c^T \cdot x \\ & s.t. & x \in \mathcal{IP} \end{array}$$

where $\mathcal{IP} = \{\, x \in \{0,1\}^n \mid a_j^T \cdot x \geq 0 \text{ for } j = 1, \ldots, m,\ x_1 = 1 \,\}$. To do this, the constraints $a_j^T \cdot x \geq b_j$ are *homogenized* by introducing an extra dummy–variable x_1 of value 1. More concret, this means that we set new values for a_j, for $j = 1, \ldots, m$, and for x and c:

$$a_j = \begin{pmatrix} -b_j \\ a_j \end{pmatrix}, \quad x = \begin{pmatrix} 1 \\ x \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ c \end{pmatrix}.$$

Notice, that the new a_j is based on the old value of b_j. Next we put $b_j = 0$, for $j = 1, \ldots, m$, and $n = n + 1$. Then we get a minimization problem of the desired structure.

Secondly, to get the naive semi–definite approximation for (L'') we put $X = x \cdot x^T$, i.e. a special matrix of rank 1, and require X to be positive semi–definite. Thus, for $x \geq 0$ it holds

[5] For a more detailed description see [7].

$x = (diag(X))^{\frac{1}{2}}$, where the notation "$^{1/2}$" for a vector means that for every component of the vector the (positive) square root is taken. If we have $x \in \{0,1\}^n$ we can equivalently use $x = diag(X)$. The constraint $x_1 = 1$ can be transformed in a linear subspace of the space of the symmetric matrices via $X_{1,1} = 1$. Because of $x_1 = 1$ and $X = x \cdot x^T$ we get $X_{1,.} = (\ x_1\ x_2\ \dots\ x_n\ )$ as the first row of X. As $diag(X) = (\ x_1^2\ \dots\ x_n^2\ )^T$ and as it is possible to express the 0-1-constraint $x_i \in \{0,1\}$ via the quadratic equation $x_i = x_i^2$, we get: $x_i \in \{0,1\} \iff X_{1,i} - X_{i,i} = 0$, for $i = 1,\dots,n$. Using $Diag(a_j) \bullet X \geq 0$, for $j = 1,\dots,n$, instead of the inequalities $a_j^T \cdot x \geq 0$ we can write $\mathcal{IP}$ equivalently as follows:

$$\mathcal{IP} = \{\ diag(X) \mid X \in \tilde{S},\ X = x \cdot x^T\ \}$$

with $\tilde{S} = \{\ X \mid X$ positiv semi-definite, $X_{1,i} - X_{i,i} = 0$, for $i = 1,\dots,n$, $X_{1,1} = 1$, $Diag(a_j) \bullet X \geq 0$, for $j = 1,\dots,m\ \}$. Because the constraint $X = x \cdot x^T$ makes it very difficult to solve the program efficiently it is omitted and therefore the set of feasible points $\mathcal{IP}$ is enlarged. With the diagonal matrix $C = Diag(c)$ we get the program

$$\begin{array}{rl} \min & C \bullet X \\ s.t. & X_{1,1} = 1, \quad X_{1,i} - X_{i,i} = 0, \text{ for } i = 2,\dots,n \\ & Diag(a_j) \bullet X \geq 0, \text{ for } j = 1,\dots,m \\ & X \text{ positive semi–definite.} \end{array}$$

To write the equalities $X_{1,1} = 1$ and $X_{1,i} - X_{i,i} = 0$, for $i = 2,\dots,n$, in the semi–definite form $A^{(1)} \bullet X = 1$ respectively $A^{(i)} \bullet X = 0$, for $i = 2,\dots,n$ we define symmetric matrices $A^{(i)}$, for $i = 1,\dots,n$, as follows: For $X_{1,1} - 1 = 0$ we define the matrix $A^{(1)}$ by

$$A^{(1)}_{k,l} = \begin{cases} 1, & \text{for } k = l = 1 \\ 0, & \text{otherwise} \end{cases}$$

We replace the equality $X_{1,i} - X_{i,i} = 0$ by $2 \cdot X_{i,i} - X_{1,i} - X_{i,1} = 0$ to get matrices $A^{(i)}$, for $i = 2,\dots,n$, with

$$A^{(i)}_{k,l} = \begin{cases} -1, & \text{for } (k = i \text{ and } l = 1) \text{ and for } (k = 1 \text{ and } l = i) \\ 2, & \text{for } k = l = i \\ 0, & \text{otherwise} \end{cases}$$

With the diagonal matrix $B^{(j)} = -Diag(a_j)$ we finally get

$$\begin{array}{lrl} (SL) & \min & C \bullet X \\ & s.t. & A^{(1)} \bullet X = 1, \quad A^{(i)} \bullet X = 0, \text{ for } i = 2,\dots,n \\ & & B^{(j)} \bullet X \leq 0, \text{ for } j = 1,\dots,m \\ & & X \text{ positive semi–definite} \end{array}$$

We point out that this semi–definite program is equivalent to the linear program obtained by replacing the constraint $x_i \in \{0,1\}$ in the 0-1-program with $x_i \in [0,1]$. With a more refined approach, better semi–definite approximations can be obtained; due to a lack of space we omit the derivation of this approach here.[6]

Experimental results will be given below.

[6] For detailed descriptions see [2] and [7].

An example: The naive semi–definite approximation of the disjunctive deductive database $P = \{\ a \vee b,\ c \leftarrow b\ \}$ of Section 3 leads to the following matrices:

$$A^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\quad A^{(2)} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A^{(3)} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\quad A^{(1)} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

$$B^{(1)} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\quad B^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\quad C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

Interior Point Algorithms

To approximate the 0-1-programs we have implemented various linear and semi–definite relaxations. To solve the resulting linear and semi–definite programs we used different *interior point algorithms*:

1. An algorithm (cf. [6]) based on the *logarithmic barrier function* and working on the *dual program*[7]; this algorithm uses a phase 1 to compute a starting point.
2. A *primal–dual method* based on a modification of an algorithm introduced in 1994 in [5]; to obtain a starting point we use an adequate enlargement of our problem
3. An *infeasible primal–dual algorithm*: This type of algorithms uses points that need to fulfill only the inequality constraints while the algorithm is running. Only in the final iterations of the program the equality constraints have to be fulfilled. In other words, these methods iterate to optimality and feasibility at the same time.

Essential for efficiently solving semi–definite programs derived from database problems using interior point methods is to exploit the sparse structure of the semi–definite matrices and to reduce the problem of finding a search direction to an efficiently solvable system of linear equations. In our implementations we construct a positive definite system[8], such that we can use the Cholesky–decomposition to solve it.[9] We get similar statements for solving linear programs with interior point methods.

[7]It is possible to construct dual programs to our so called *primal* linear and semi–definite programs mentioned above. Under certain conditions the dual programs have the same set of optimal solutions and it can be advantageous to solve the dual program instead of the primal program. Often it makes sense to solve both problems at the same time which leads to so called *primal-dual algorithms*.

[8]i.e. the matrix M describing the linear equality system is symmetric and fulfills the condition $v^T \cdot M \cdot v > 0$ for all vectors $v \neq 0$

[9]The Cholesky–decomposition of a positiv definite matrix A is defined by $A = L \cdot L^T$, where L is an upper triangular matrix. To the efficient solution of positive definite systems via the Cholesky–decomposition see for instance [13].

Experimental Results

Using semi–definite relaxations and adequate heuristics it is possible to solve 0-1-programs derived from disjunctive deductive databases by solving 1.6 semi–definite programs on the average. The following table shows the quality of two different heuristics H_1 and H_2, which have been developed in [7], and of some linear (L) and semi–definite (S_0, S_1, S_2, S_0', S_1', S_2') relaxations. S_0 uses the naive semi–definite relaxation (SL), whereas S_1 and S_2 use the mentioned more refined semi–definite relaxations (see [7]). S_0', S_1' and S_2' use the same relaxations, but another interior point method (see [7]). The numbers say how many linear or semi–definite programs had to be solved on the average to construct a 0-1-solution:

	L	S_0	S_1	S_2	S_0'	S_1'	S_2'
H_1	3.1	3.2	2.7	1.6	3.1	2.6	1.7
H_2	3.2	3.4	2.4	1.6	3.1	2.5	1.6

The experiments show that as the size of the database increases, the numbers in the table increase very slowly. The time required to compute the minimal models does not increase exponentially, if the number of models does not increase significantly. Our current implementation is done in MATLAB, and it will be replaced by a faster implementation in C.
It can be shown theoretically that the approaches S_1 and S_2 in general provide better relaxations than the linear approach L (see [7] and [8]).

5 Conclusions

Semi–definite relaxations combined with interior point methods can be used for computing *minimal models* of disjunctive deductive databases. In particular cost–minimal models and minimal models w.r.t. set–inclusion can be computed.

Semi–definite optimzation can handle much more general types of constraints than linear programming. When applied in *constraint logic programming CLP*, this will increase the expressive power of the constraints that can be processed.

Due to the *polynomial* time complexity of interior point methods – for linear, semi–definite and even a large class of other convex problems – the proposed combination is especially useful for dealing with large problems having complex constraints, where the simplex method cannot be applied.

Currently, semi–definite programming is investigated for solving various types of problems from *combinatorial optimization*; e.g. it is the most effective method for solving *max–cut* problems.

References

[1] T. Frühwirth, S. Abdennadher: *Constraint–Programmierung*, Springer, 1997

[2] F. Alizadeh: *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization, 5(1), pp. 13-51, 1995

[3] P. Barth, A. Bockmayer: *Global Consistency in CLP($\mathcal{PB}$)*, Proc. Tenth Workshop on Logic Programming 1995 (WLP'95)

[4] C. Bell, A. Nerode, R.T. Ng, V.S. Subrahmanian: *Mixed Integer Programming Methods for Computing Nonmonotonic Deductive Databases*, Journal of the ACM, vol. 41(6), pp. 1178-1215, 1994

[5] C. Helmberg, F. Rendl, H. Wolkowicz, R.J. Vanderbei: *An Interior–Point Method for Semidefinite Programming*, Report 264, Institut für Mathematik, Christian–Doppler–Laboratorium "Diskrete Optimierung", Technische Universität Graz, 1994

[6] F. Jarre, M.A. Saunders: *A Practical Interior–Point Method for Convex Programming*, SIAM Journal on Optimization, vol. 5(1), pp. 149-171, 1995

[7] R. Lepenis: *Modellberechnung bei disjunktiven deduktiven Datenbanken mittels semidefiniter Programmierung*, Diplomarbeit, Bayerische Julius–Maximilians–Universität Würzburg, 1998

[8] L. Lovasz, A. Schrijver, *Cones of Matrices and Setfunktions, and 0-1 Optimization*, SIAM Journal on Optimization, 1(2), 1991

[9] Y. Nesterov, A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, Philadelphia, 1993

[10] C. Roos, T. Terlaky, J. P. Vial, *Theory and Algorithms for linear Optimization. An Interior Point Approach*, Wiley, 1997

[11] D. Seipel, J. Minker, C. Ruiz: *Model Generation and State Generation for Disjunctive Logic Programs*, Journal of Logic Programming, vol. 32(1), 1997, pp. 48-69.

[12] L. Vandenberghe, S. Boyd, *Semidefinite Programming*, SIAM Review, Vol. 38, No. 1, pp. 49-95, 1996

[13] S.J. Wright: *Primal–Dual Interior–Point Methods*, SIAM, 1997

[14] Y. Ye: *Interior Point Algorithms, Theory and Analysis*, Wiley, 1997