Improved Encodings of Acyclicity for Translating Answer Set Programming into Integer Programming

Masood Feyzbakhsh Rankooh and **Tomi Janhunen**
Tampere University, Finland
{masood.feyzbakhshrankooh, tomi.janhunen}@tuni.fi

Abstract

In this work, we introduce novel translations of Answer Set Programming (ASP) into Integer Programming (IP). While building upon a previously introduced IP translation, we revisit the translation of acyclicity constraints essential for capturing answer sets precisely. By leveraging vertex elimination graphs, we demonstrate that a new translation of acyclicity can yield integer programs with a more restrictive linear relaxation compared to previous methods. This enhancement enables IP solvers to prune the search space more efficiently. Furthermore, we show how acyclicity can be expressed more concisely in IP given any feedback vertex set of the underlying dependency graph. Experimental results underscore the improved efficiency of our methods over the previously implemented translation. The new vertex elimination based translation with Gurobi as the back-end solver turns out competitive against Clingo, a state-of-the-art native ASP solver, in a number of non-tight Answer Set Optimization (ASO) benchmarks.

1 Introduction

Answer set programming (ASP) features highly expressive, rule-based languages for knowledge representation and reasoning (see, e.g., [Brewka *et al.*, 2011] for an overview). Typically, the solutions of a search problem are captured as *answer sets* of the respective logic program encoding the problem. Besides native answer set *solvers*, the search for answer sets can be implemented (i) by *translating* rules into other kinds of constraints, e.g., propositional clauses [Janhunen, 2006] or linear (in)equalities [Liu *et al.*, 2012], (ii) by calling alternative solvers capable of satisfying such constraints, and (iii) by extracting answer sets from satisfying assignments. To capture answer sets and extended rules [Simons *et al.*, 2002] properly, such translations must address aggregated conditions, recursive definitions, and default negation. E.g., the Liu et al. [2012] translation from ASP to *integer programming* (IP) conforms to this structure. In particular, the translation deploys *level numbers* to rule out self-supporting (circular) recursive definitions and, thus, essentially puts them subject to an *acyclicity constraint*, cf. [Bomanson *et al.*, 2016].

One important aspect of translating complex problems into IP is coming up with programs whose *linear programming* (LP) relaxations are reasonably restrictive. Restrictive LP relaxations are important, especially when coupled with advanced optimization techniques such as the *branch-and-cut* method [Mitchell, 2002]. By relaxing integer constraints in IP, i.e., by allowing variables to take fractional values, a corresponding linear programming relaxation is formulated. The incorporation of restrictive LP relaxations, involving additional constraints or tightened versions of existing ones, becomes synergistic with the branch-and-cut method. This sophisticated optimization approach systematically explores the solution space by iteratively branching on variables and utilizing *cutting planes* derived from the LP relaxations. The interplay of restrictive relaxations and the branch-and-cut method improves the overall efficiency of optimization, enabling the resolution of more complex problems.

In this work, we introduce two new translations from ASP to IP, building on the prior IP translation [Liu *et al.*, 2012]. In our first translation, we refine the representation of acyclicity constraints using *vertex elimination graphs* in analogy to [Rankooh and Janhunen, 2022]. The resulting integer programs have more restrictive linear relaxations, enabling IP solvers to prune the search space more efficiently. In our second translation, we improve the representation of the acyclicity constraint in IP based on any *feedback vertex set* of the underlying dependency graph. The latter translation is particularly effective in settings where vertex elimination becomes computationally costly. The experimental evaluation reveals the superior performance of the new translations in contrast with the Liu et al. [2012] translation. The new vertex elimination based translation with CPLEX is competitive against Clingo, a leading native ASP solver, especially in non-tight Answer Set Optimization (ASO) benchmarks.

While the focus of this paper is on improving answer set solving, it is essential to note that the improved representations of the acyclicity constraint are not exclusive to ASP. Consequently, these can be effectively employed to enhance any IP problems that incorporate acyclicity constraints. This versatility underscores the potential impact and usefulness of the proposed techniques beyond the specific realm of ASP.

We continue with preliminaries in Section 2 and recall the basics of translations in Sections 3 and 4. Our new translations are presented in Sections 5 and 6. Experimental evalua-

tion in Section 7 is followed by conclusions in Section 8.

2 Preliminaries

In the sequel, *weight constraint programs* (WCPs) [Simons *et al.*, 2002] consist of *rules* the following three forms:

$$a \leftarrow b_1, \ldots, b_n, \texttt{not } c_1, \ldots, \texttt{not } c_m. \quad (1)$$

$$\{a\} \leftarrow b_1, \ldots, b_n, \texttt{not } c_1, \ldots, \texttt{not } c_m. \quad (2)$$

$$a \leftarrow k \leq [\, b_1 = w_1, \ldots, b_n = w_n, \texttt{not } c_1 = w_{n+1}, \ldots, \texttt{not } c_m = w_{n+m}\,]. \quad (3)$$

The symbols a, $b_1, \ldots, b_n$ where $n \geq 0$, and $c_1, \ldots, c_m$ where $m \geq 0$ occurring in the rules signify *(propositional) atoms*, and the symbol "not" is employed to denote *negation by default*. The *bound* k and the *weights* $w_1, \ldots, w_{n+m}$ in (3) are non-negative integers. Rules of the forms (1)–(3) are classified as *normal*, *choice*, and *weight rules*, respectively, as detailed in [Simons *et al.*, 2002]. Essentially, each rule r provides a basis for deriving its *head* $\mathrm{head}(r) = a$ if the conditions in its *body* $\mathrm{body}(r)$ are fulfilled. In other words, atoms involved can be either derived by other rules or not. For a choice rule r of the form (2), the derivation of $\mathrm{head}(r)$ is optional. In the case of a weight rule r of the form (3), $\mathrm{head}(r)$ is to be derived when the sum of weights associated with *satisfied* body conditions must be at least k.

We use $\mathrm{body}^+(r)$ and $\mathrm{body}^-(r)$ to denote the sets of atoms $b_1, \ldots, b_n$ (resp. $c_1, \ldots, c_m$) occurring positively (resp. negatively) in $\mathrm{body}(r)$. The rules of a *positive* logic program satisfy $m = 0$, i.e., are negation free. Given a WCP P, the *definition* of an atom a in P is the set of rules $\mathrm{def}(a) = \{r \in P \mid \mathrm{head}(r) = a\}$. The *signature* of a WCP P is defined as the set of atoms $\mathrm{At}(P) = \bigcup_{r \in P}(\{\mathrm{head}(r)\} \cup \mathrm{body}^+(r) \cup \mathrm{body}^-(r))$. This set comprises atoms that occur in P. The *positive dependency graph* of P is represented as $\mathrm{DG}^+(P) = \langle \mathrm{At}(P), \succeq \rangle$, where the relation $a \succeq b$ holds for $a, b \in \mathrm{At}(P)$ if $\mathrm{head}(r) = a$ and $b \in \mathrm{body}^+(r)$ for some rule $r \in P$. Within this graph, a *strongly connected component* (SCC) is a maximal subset $S \subseteq \mathrm{At}(P)$ wherein all distinct atoms $a, b \in S$ are mutually dependent (transitively) on each other through a directed path in $\mathrm{DG}^+(P)$.

An *interpretation* $I \subseteq \mathrm{At}(P)$ specifies the atoms $a \in \mathrm{At}(P)$ that are considered *true* ($a \in I$) and those that are considered *false* ($a \notin I$). An *interpretation* $I \subseteq \mathrm{At}(P)$ is deemed to satisfy a rule $r \in P$ in the forms (1) and (3), denoted as $I \models r$, if the fulfillment of the body, expressed as $I \models \mathrm{body}(r)$, implies that $\mathrm{head}(r) \in I$, i.e., $I \models \mathrm{head}(r)$. In the case of a choice rule r of form (2), $I \models r$ unconditionally. Furthermore, the interpretation I is considered a (classical) *model* of P if $I \models r$ holds for every $r \in P$. For every positive program P, there exists a unique *least model* denoted as $\mathrm{LM}(P)$, obtained as the intersection $\bigcap\{I \subseteq \mathrm{At}(P) \mid I \models P\}$, the least fix-point of T_P operator.

Given an interpretation I, the *reduct* r^I of r concerning I is computed by partially evaluating the negative conditions of r. In the case of a normal rule (1), $r^I = \emptyset$ if $c_i \in I$ for some $1 \leq i \leq m$ and $r^I = \{a \leftarrow b_1, \ldots, b_n\}$ otherwise. For a choice rule (2), the latter case additionally necessitates that $a \in I$. Regarding a weight rule (3), $r^I = \{a \leftarrow l \leq [b_1 = w_1, \ldots, b_n = w_n]\}$. Here, the updated bound l is determined from k by subtracting w_{n+i} for each $1 \leq i \leq m$ such that $c_i \notin I$. Finally, for an entire WCP P, the reduct $P^I = \bigcup\{r^I \mid r \in P\}$ and I is considered a *stable model* of P if and only if $I = \mathrm{LM}(P^I)$. It is also beneficial to distinguish the *supporting rules* of P with respect to I, denoted as $\mathrm{SR}_P(I) = \{r \in P \mid \mathrm{head}(r) \in I, I \models \mathrm{body}(r)\}$. Then, a model $I \models P$ is *supported* (by P) when $I = \{\mathrm{head}(r) \mid r \in \mathrm{SR}_P(I)\}$. Every stable model of P is supported by P, but supported models are not necessarily stable, as exemplified by $I = \{a\}$ for the program $P = \{a \leftarrow a.\}$.

Example 1. *Consider a WCP P consisting of the rules:*

$$a_1 \leftarrow a_2, a_3.$$
$$\{a_2\}.$$
$$a_3 \leftarrow 3 \leq [a_1 = 1, a_2 = 2, \texttt{not } a_2 = 3].$$

The signature $\mathrm{At}(P) = \{a_1, a_2, a_3\}$ *and* $\mathrm{DG}^+(P)$ *has SCCs* $S_1 = \{a_1, a_3\}$ *and* $S_2 = \{a_2\}$. *There are two stable models* $M_1 = \{a_2\}$ *and* $M_2 = \{a_3\}$ *justified by reducts* $P^{M_1} = \{a_1 \leftarrow a_2,\ a_3.\ a_2.\ a_3 \leftarrow 3 \leq [a_1 = 1, a_2 = 2].\}$ *and* $P^{M_2} = \{a_1 \leftarrow a_2,\ a_3.\ a_3 \leftarrow 0 \leq [a_1 = 1, a_2 = 2]\}$. *But the model* $M_2 = \{a_1, a_2, a_3\}$ *is only supported, not stable.* ■

3 Instrumentation with Acyclicity Constraint

Our next objective is to revisit the *acyclicity translation* $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ of a WCP P [Bomanson *et al.*, 2016], which employs special *dependency atoms* $\texttt{dep}(a, b)$ to represent the activation of the corresponding arc $\langle a, b \rangle \in \mathrm{DG}^+(P)$ in the acyclicity constraint. This transformation is applied on an atom-by-atom basis and is necessary only for atoms $a \in \mathrm{At}(P)$ involved in a non-trivial SCC S of P with $|S| > 1$. For such an S and an atom $a \in S$, the approach is to *instrument* P with additional rules that capture *well-support* for a (cf. [Bomanson *et al.*, 2016]). For every arc $\langle a, b \rangle \in \mathrm{DG}^+(P)$ that is specific to S, the potential dependency of a on b is articulated through a choice rule $\{\texttt{dep}(a, b)\} \leftarrow b$. In addition to the dependency atoms, special atoms are introduced to enforce well-support for the atom a with respect to its defining rules $\{r_1, \ldots, r_k\} = \mathrm{def}(a)$. These special atoms are denoted as $\texttt{ws}(r_1), \ldots, \texttt{ws}(r_k)$, and they are utilized in a constraint that ensures well-support for a: $f \leftarrow a, \texttt{not ws}(r_1), \ldots, \texttt{not ws}(r_k), \texttt{not } f$ Here, f is introduced as a new atom. Given the SCC S of a in P, it is assumed that each $\mathrm{body}^+(r)$ in (1)–(3) is ordered in such a way that for some $0 \leq l \leq n$, $b_1 \in S, \ldots, b_l \in S$ while $b_{l+1} \notin S, \ldots, b_n \notin S$. Then, if a *defining* rule $r \in \mathrm{def}(a)$ is of the form (1) or (2), the rule (4) below captures well-support mediated by r. If the defining rule r is of the form (3), then the corresponding rule for well-support is (5).

$$\texttt{ws}(r) \leftarrow \texttt{dep}(a, b_1), \ldots, \texttt{dep}(a, b_l), b_{l+1}, \ldots, b_n, \texttt{not } c_1, \ldots, \texttt{not } c_m. \quad (4)$$

$$\texttt{ws}(r) \leftarrow k \leq [\, \texttt{dep}(a, b_1) = w_1, \ldots, \texttt{dep}(a, b_l) = w_l, \texttt{not } c_1 = w_{n+1}, \ldots, \texttt{not } c_m = w_{n+m}\,]. \quad (5)$$

For the program $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ constructed in this manner, the distinction between stable and supported models disappears when we specifically consider *acyclic models* I. An

acyclic model I is one in which the digraph induced by the set of arcs $\{\langle a, b\rangle \mid \texttt{dep}(a, b) \in I\}$ is acyclic.

Proposition 1 ([Bomanson *et al.*, 2016, Theorem 3.11])**.** *Let P be a WCP.*

1. *If M is a stable model of P, then $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ has an acyclic supported model N such that $M = N \cap \mathrm{At}(P)$.*
2. *If N is an acyclic supported model of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$, then $M = N \cap \mathrm{At}(P)$ is a stable model of P and well-supported by $R = \{r \in P \mid \texttt{ws}(r) \in N, \mathrm{head}(r) \in N\}$.*

Example 2. *Consider the program in Example 1. Remember that $S = \{a_1, a_3\}$ is the only non-trivial SCC. Let r_1 be the defining rule for a_1, and r_2 the one for a_3. By adding rules*

$\{\texttt{dep}(a_1, a_3)\} \leftarrow a_3.$
$\{\texttt{dep}(a_3, a_1)\} \leftarrow a_1.$
$\texttt{ws}(r_1) \leftarrow \texttt{dep}(a_1, a_3),\ a_2.$
$\texttt{ws}(r_2) \leftarrow 3 \leq [\texttt{dep}(a_3, a_1) = 1, a_2 = 2, \texttt{not } a_2 = 3].$
$f \leftarrow a_1,\ \texttt{not ws}(r_1),\ \texttt{not } f.$
$f \leftarrow a_3,\ \texttt{not ws}(r_2),\ \texttt{not } f.$

we can ensure that all acyclic supported models of the program are stable. In particular, we note that the model $N = \{a_1, a_2, a_3, \texttt{dep}(a_1, a_3), \texttt{dep}(a_3, a_1), \texttt{ws}(r_1), \texttt{ws}(r_2)\}$ is supported, but not acyclic and, therefore, not stable. ■

4 Translation into Integer Programming

Once the acyclicity translation $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ of a WCP P is obtained, translating P to an integer program is possible. This translation has previously been done in [Liu *et al.*, 2012], which we review here. We proceed by breaking down the translation into two parts: (i) translation of rules in $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ and (ii) translation of acyclicity. For the rest of this paper, we assume that $\mathrm{DG}^+(P)$ only includes arcs with both ends in the same SCC. This assumption does not harm the generality of our arguments in the sequel, as such arcs are the only ones of value as long as acyclicity is concerned.

4.1 Translation of Rules

Given a WCP P, for the sake of simplicity, we assume that members of $\mathrm{At}(P)$ are indexed in the form of $a_1, ..., a_n$. We define a mapping $\mathcal{V}$ that maps every atom $\texttt{a} \in \mathrm{At}(P)$ to a binary variable a, and every atom $\texttt{dep}(a_i, a_j) \in \mathrm{At}(\mathrm{Tr}_{\mathrm{ACYC}}(P))$ to a binary variable $x_{\langle i,j\rangle}$ to represent the respective atoms. Also, for each rule in $\mathrm{Tr}_{\mathrm{ACYC}}(P)$, we consider a binary variable bd^r to represent the notion of r causing the truth of its head. We construct an integer program IP^r_P by adding inequalities (6) and (7) from below. For every normal rule r of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ in the form (1), we include

$$\sum_{b \in \mathrm{body}^+(r)} \mathcal{V}(b) + \sum_{c \in \mathrm{body}^-(r)} (1 - \mathcal{V}(c)) - |\mathrm{body}(r)| \cdot bd^r \geq 0, \tag{6}$$

and

$$\sum_{b \in \mathrm{body}^+(r)} \mathcal{V}(b) + \sum_{c \in \mathrm{body}^-(r)} (1 - \mathcal{V}(c)) - bd^r \geq |\mathrm{body}(r)| - 1. \tag{7}$$

Inequality (6) guarantees that if the body of r is satisfied, bd^r must be set to one. Inequality (7) is for the completion and prevents bd^r to be one unless the body holds. For every choice rule r of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ in the form (2), only inequality (7) is needed. That is because even if the body of r is satisfied, r may not cause its head to be in the answer set. For every weight rule r of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ in the form (3), we include

$$\sum_{b_i \in \mathrm{body}^+(r)} w_i \cdot \mathcal{V}(b_i) + \sum_{c_i \in \mathrm{body}^-(r)} w_{n+i} \cdot (1 - \mathcal{V}(c)) - k \cdot bd^r \geq 0, \tag{8}$$

and

$$\sum_{b_i \in \mathrm{body}^+(r)} w_i \cdot \mathcal{V}(b_i) + \sum_{c_i \in \mathrm{body}^-(r)} w_{n+i} \cdot (1 - \mathcal{V}(c)) - bd^r \geq k - 1. \tag{9}$$

Similarly to the case of normal rules, (8) guarantees that if the body of r is satisfied, bd^r must be set to one, and the inequality (9) prevents bd^r from being one unless the body holds. For every atom $\texttt{a} \in \mathrm{At}(\mathrm{Tr}_{\mathrm{ACYC}}(P))$ we include

$$\sum_{r \in \mathrm{def}(a)} bd^r - |\mathrm{def}(a)| \cdot \mathcal{V}(a) \leq 0, \tag{10}$$

and

$$\sum_{r \in \mathrm{def}(a)} bd^r - \mathcal{V}(a) \geq 0. \tag{11}$$

Inequality (10) guarantees that if at least one rule causes $\texttt{a} \in \mathrm{At}(\mathrm{Tr}_{\mathrm{ACYC}}(P))$ to be in the answer set, then $\mathcal{V}(a)$ must be set to one. Inequality (11) causes $\mathcal{V}(a)$ to be zero in the case that no rule is causing a to be included in the answer set.

4.2 Translation of Acyclicity

For $a \in \mathrm{At}(P)$, let $SCC(a)$ denote the strongly connected component of $\mathrm{DG}^+(P) = \langle \mathrm{At}(P), \succeq \rangle$ including a. We construct integer program IP^o_P by adding for every $\langle a_i, a_j\rangle \in \succeq$,

$$x_i - x_j \geq (x_{\langle i,j\rangle} - 1)|SCC(a_i)| + 1. \tag{12}$$

Inequality (12) is a translation of the Bellman-Ford equation [Bellman, 1958] into integer programming, which causes every model M to set x_i to the length of the longest path reaching a_i in the graph induced by $\{\langle a_i, a_j\rangle \mid M(x_{\langle i,j\rangle}) = 1\}$. As a result every assignment causing a cycle in its induced graph cannot be consistent. It should be clear that for every variable x_i, the upper bound on the value assigned by any model to x_i is $|SCC(a_i)|$.

It has been shown that the union of IP^r_P and IP^o_P is a correct IP translation of P.

Proposition 2 ([Liu *et al.*, 2012])**.** *Let P be a WCP.*

1. *If M is a stable model of P, then $IP^r_P \cup IP^o_P$ has a model N such that if $x \in M$ then $N(\mathcal{V}(x)) = 1$.*
2. *If N is a model of $IP^r_P \cup IP^o_P$, then $M = \{x | N(\mathcal{V}(x)) = 1\} \cap \mathrm{At}(P)$ is a stable model of P.*

5 Acyclicity via Vertex Elimination

The concept of vertex elimination graphs, originally introduced by Rose and Tarjan [1975], has been recently shown effective for guaranteeing acyclicity in constraint programs with underlying graphs [Rankooh and Rintanen, 2022; Rankooh and Janhunen, 2022; Zhou *et al.*, 2023]. Here, we show that using vertex elimination graphs, one can produce IP translations of answer set programs with more restrictive LP relaxations compared to that of $\text{IP}^r_P \cup \text{IP}^o_P$. Furthermore, we prove that a feedback vertex set (a subset of vertices that contains at least one vertex from every cycle in the graph) of a graph can be obtained as a byproduct of the vertex elimination process. We later use the obtained feedback vertex set to improve the IP encoding (12) of the Bellman-Ford equation.

5.1 Vertex Elimination Graphs

Given a digraph $\mathcal{G} = \langle V, E\rangle$, an ordering of V is a bijection $\alpha : \{1, \ldots, n\} \to V$. The *fill-in* $F(v)$ for a vertex v in a directed graph is essentially the set of arcs that, if added, would form a complete neighborhood around the vertex v. In other words, it is the set of arcs from the in-neighbors of v to the out-neighbors of v, formally defined as

$$F(v) = \{\langle x, y\rangle \mid \langle x, v\rangle \in E, \langle v, y\rangle \in E, x \neq y\}. \quad (13)$$

The *v-elimination* graph of $\mathcal{G}$ is obtained by removing the vertex v from $\mathcal{G}$, and adding the fill-in of v to the resulting graph. Formally, it is represented as $\mathcal{G}(v) = \langle V \setminus \{v\}, E(v) \cup F(v)\rangle$, where $E(v) = \{\langle x, y\rangle \mid \langle x, y\rangle \in E, x \neq v, y \neq v\}$.

Given a digraph $\mathcal{G}$ and an ordering α of its vertices, the *elimination process* of $\mathcal{G}$ according to α is the a sequence denoted as $\mathcal{G} = \mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{n-1}$, where $\mathcal{G}_i$ is the $\alpha(i)$-elimination graph of $\mathcal{G}_{i-1}$ for $i = 1, \ldots, n-1$. This elimination process is performed iteratively for each vertex in the specified ordering α. At each step, a vertex is eliminated, and the associated fill-in arcs are added to the graph, resulting in a sequence of graphs.

The fill-in of the digraph $\mathcal{G}$ according to the ordering α, denoted by $F_\alpha(\mathcal{G})$, is the set of all arcs added to $\mathcal{G}$ in the vertex elimination process. Formally, $F_\alpha(\mathcal{G})$ is defined by (14) where $f(v)$ is the fill-in of $v = \alpha(i)$ in the graph $\mathcal{G}_{i-1}$:

$$F_\alpha(\mathcal{G}) = \bigcup_{v \in V} f(v). \quad (14)$$

The vertex elimination graph of $\mathcal{G}$ according to α, denoted by $\mathcal{G}^*_\alpha$, is defined as the union of all graphs produced in the elimination process of $\mathcal{G}$ according to α:

$$\mathcal{G}^*_\alpha = \langle V, E \cup F_\alpha(\mathcal{G})\rangle. \quad (15)$$

The *elimination width* of α [Hunter and Kreutzer, 2008] is defined by the maximum number of outgoing arcs of vertex $\alpha(i)$ in $\mathcal{G}_{i-1}$, for $i = 1, ..., |V|$. The asymptotic number of arcs in $\mathcal{G}^*_\alpha$ is then $\mathcal{O}(|E| + \delta|V|)$, where δ is the elimination width of α. The problem of finding the optimal ordering function, one with the smallest elimination width, has been shown to be NP-complete [Rose and Tarjan, 1975]. Nevertheless, there are effective heuristics for finding empirically useful orderings. Examples are the *minimum fill-in* and *minimum degree* that accordingly choose a vertex for removal at each step during the elimination process. An important property of the vertex elimination process is that if the original graph $\mathcal{G}$ has a directed cycle, then $\mathcal{G}^*_\alpha$ will have a cycle of length 2. This property is independent of the specific ordering α and is inherent to the nature of the elimination process in the presence of directed cycles.

Theorem 1. *Let $\mathcal{G} = \langle V, E\rangle$ be a digraph, α an ordering of its vertices, and $\mathcal{G}^*_\alpha$ the vertex elimination graph of $\mathcal{G}$ according to α. For every cycle $v_1, ..., v_k, v_1$ in $\mathcal{G}$, there exists some $v_i, v_j \in \{v_1, ..., v_k\}$ such that arcs $\langle v_i, v_j\rangle$ and $\langle v_j, v_i\rangle$ are both present in $\mathcal{G}^*_\alpha$.*

Proof. We give the proof by induction on k. For the case of $k = 2$, we have the cycle v_1, v_2, v_1. The conclusion holds because $\mathcal{G}$ is a subgraph of $\mathcal{G}^*_\alpha$. Now assume that the conclusion holds for every cycle of length k. Let $v_1, ..., v_{k+1}, v_1$ be a cycle of length $k+1$ in $\mathcal{G}$. Let v_l be the first member of $\{v_1, ..., v_{k+1}\}$ that is eliminated according to α. Assume without loss of generality that $1 < l < k+1$ (otherwise we can always reorder indices of the cycle to make this assumption hold). Now $v_1, ..., v_{l-1}, v_{l+1}, ..., v_{k+1}, v_1$, produced after eliminating v_l, is a cycle of length k in $\mathcal{G}^*_\alpha$. By the induction hypothesis, there exist some $v_i, v_j \in \{v_1, ..., v_{l-1}, v_{l+1}, ..., v_k\}$ such that arcs $\langle v_i, v_j\rangle$ and $\langle v_j, v_i\rangle$ are both present in $\mathcal{G}^*_\alpha$. □

Theorem 1 shows how every cycle of a directed graph can be represented by a pair of vertices in it. As a result, by choosing one vertex out of each of such pairs, one can obtain a feedback vertex set.

Corollary 1. *Let $\mathcal{G} = \langle V, E\rangle$ be a digraph, α an ordering of its vertices, and $\mathcal{G}^*_\alpha$ the vertex elimination graph of $\mathcal{G}$ according to α. Let F be the set of all vertices $v \in V$ such that for some $u \in V$ ordered by α after v, arcs $\langle v, u\rangle$ and $\langle u, v\rangle$ are both present in $\mathcal{G}^*_\alpha$. Then F is a feedback vertex set of $\mathcal{G}$.*

5.2 Vertex Elimination Based Translation

Let α be an ordering of vertices of $\text{DG}^+(P)$, and $\mathcal{G}_0, \ldots, \mathcal{G}_{n-1}$ the elimination process of $\text{DG}^+(P)$ according to α. We produce the IP^{ve}_P by adding the following. For every arc $\langle a_i, a_j\rangle$ in $\text{DG}^+(P)$, we add

$$x'_{\langle i,j\rangle} \geq x_{\langle i,j\rangle}. \quad (16)$$

For every $\langle a_i, a_j\rangle \in f(a_k)$, we add

$$x'_{\langle i,j\rangle} \geq x'_{\langle i,k\rangle} + x'_{\langle k,i\rangle} - 1, \quad (17)$$

and for every i and j with $\langle a_i, a_j\rangle \in \mathcal{G}^*_\alpha$ and $\langle a_j, a_i\rangle \in \mathcal{G}^*_\alpha$,

$$x'_{\langle i,j\rangle} + x'_{\langle j,i\rangle} \leq 1. \quad (18)$$

Let $\text{Var}(X)$ denote the set of variables in an integer/linear program X. Considering Proposition 1, Theorem 2 shows that there is a correspondance between the set of stable models of P and the set of integer models of $\text{IP}^r_P \cup \text{IP}^{ve}_P$.

Theorem 2. *If $IP^r_P \cup IP^{ve}_P$ has a model M, then the restriction of M on $Var(IP^r_P)$ is an acyclic model of IP^r_P. If N is an acyclic model for IP^r_P, then there is an extension of N that is a model for $IP^r_P \cup IP^{ve}_P$.*

Proof. Let M be a model for $\text{IP}^r_P \cup \text{IP}^{ve}_P$. Then M is a model of IP^r_P. Assume M is not acyclic. Without loss of generality (regarding the indices of vertices) assume that the graph induced by $\{\langle a_i, a_j\rangle \mid M(x'_{\langle i,j\rangle}) = 1\}$, denoted by $\mathcal{H}$, has a cycle $a_1, ..., a_k, a_1$. According to Theorem 1, there are vertices a_i and a_j such that $\langle a_i, a_j\rangle$ and $\langle a_j, a_i\rangle$ are both present in $\mathcal{H}^*_\alpha$. However, because $\mathcal{H}$ is a subgraph of $\mathcal{G} = \text{DG}^+(P)$, $\mathcal{H}^*_\alpha$ is clearly a subgraph of $\mathcal{G}^*_\alpha$, and by (16) and (17), we have: $M(x'_{\langle i,j\rangle}) = 1$ and $M(x'_{\langle j,i\rangle}) = 1$. Therefore, $M(x'_{\langle i,j\rangle}) + M(x'_{\langle j,i\rangle}) = 2$ that contradicts (18).

Let N be an acyclic model for IP^r_P. We construct a model N' for $\text{IP}^r_P \cup \text{IP}^{ve}_P$ as an extension of N by letting $N'(x) = N(x)$ for $x \in \text{Var}(\text{IP}^r_P)$. The graph induced by $\{\langle a_i, a_j\rangle \mid N(x_{\langle i,j\rangle}) = 1\}$, denoted by $\mathcal{D}$ is acyclic, and there is a topological total ordering β of vertices of $\text{DG}^+(P)$ according to $\mathcal{D}$. We set $N'(x'_{\langle i,j\rangle})$ to 1, for arcs $\langle a_i, a_j\rangle$ in $\mathcal{D}^*_\alpha$ if and only if a_i is ordered before a_j according to β. We show that N' satisfies (16) to (18), by these settings.

- If $N(x_{\langle i,j\rangle}) = 1$, then there is an arc from a_i to a_j in $\mathcal{D}$, and therefore a_j is ordered after a_i according to β. Thus, $N'(x'_{\langle i,j\rangle}) = 1$ and inequality (16) is satisfied.
- For every $\langle a_i, a_j\rangle \in f(a_k)$, if either $N'(x'_{\langle i,k\rangle}) = 0$ or $N'(x'_{\langle k,i\rangle}) = 0$, then inequality (17) is trivially satisfied. However, if $N'(x'_{\langle i,k\rangle}) = 1$ and $N'(x'_{\langle k,i\rangle}) = 1$, then a_i is ordered before a_k, and a_k is ordered before a_j according to β. Therefore, a_i is ordered before a_j according to β. Thus, $N'(x'_{\langle i,j\rangle}) = 1$ and inequality (17) is satisfied.
- For every i and j such that $\langle a_i, a_j\rangle \in \mathcal{G}^*_\alpha$, if $N'(x'_{\langle i,j\rangle}) = 1$, then a_i is ordered before a_j according to β, and $N'(x'_{\langle j,i\rangle}) = 0$. We conclude that (18) is satisfied. □

We now show that the linear relaxation of our vertex elimination based translation is at least as restrictive as the linear relaxation of the translation introduced by Liu et al. [2012].

Lemma 1. *Let LP^{ve}_P be the LP relaxation of $\text{IP}^r_P \cup \text{IP}^{ve}_P$, and $v_1, ..., v_k, v_{k+1} = v_1$ be a cycle in $\mathcal{G}^*_\alpha$. If M is a model for LP^{ve}_P, then $\sum_{i=1,...,k} M(x'_{\langle i,i+1\rangle}) \leq k - 1$.*

Proof. We give the proof by induction on k. For $k = 2$, the conclusion holds because M satisfies constraint (18). Assume that the conclusion holds for any cycle of length k. Let $v_1, ..., v_{k+1}, v_{k+2} = v_1$ be a cycle in $\mathcal{G}^*_\alpha$ of length $k + 1$. Assume the v_j is the first vertex in this cycle eliminated according to the ordering α. Then $v_1, ..., v_{j-1}, v_{j+1}, ..., v_{k+2} = v_1$ must be a cycle of length k in $\mathcal{G}^*_\alpha$. Therefore $\sum_{i=1,...,j-2,j+1,...,k+1} M(x'_{\langle i,i+1\rangle}) + M(x'_{\langle j-1,j+1\rangle}) \leq k - 1$. However, since according to (17), $M(x'_{\langle j-1,j\rangle}) + M(x'_{\langle j,j+1\rangle}) - 1 \leq M(x'_{\langle j-1,j+1\rangle})$, we have $\sum_{i=1,...,j-2,j+1,...,k+1} M(x'_{\langle i,i+1\rangle}) + M(x'_{\langle j-1,j\rangle}) + M(x'_{\langle j,j+1\rangle}) - 1 \leq k - 1$, and therefore, $\sum_{i=1,...,k+1} M(x'_{\langle i,i+1\rangle}) \leq k$ □

Let LP^{ve}_P and LP^o_P be the LP relaxation of $\text{IP}^r_P \cup \text{IP}^{ve}_P$ and $\text{IP}^r_P \cup \text{IP}^o_P$, respectively. Assume that LP^{ve}_P has a feasible model M. We produce $\text{LP}^{o,M}_P$, by replacing (12) in LP^o_P with

$$x_i - x_j \geq (M(x'_{\langle i,j\rangle}) - 1)(|SCC(a_i)|) + 1. \tag{19}$$

Lemma 2. *For every model M of LP^{ve}_P, $\text{LP}^{o,M}_P$ has a model N such that $N(a) = M(a)$ for every $a \in$ Var(IP^r_P).*

Proof. From M, we construct a model N for $\text{LP}^{o,M}_P$. For every variable $a \in \text{Var}(\text{IP}^r_P)$, let $N(a) = M(a)$. Let G_M be a weighted graph with the same vertices as in $\text{DG}^+(P)$. For every arc $\langle a_i, a_j\rangle$ in $\text{DG}^+(P)$, we add $\langle a_i, a_j\rangle$ to G_M with weight $(-M(x'_{\langle i,j\rangle}) + 1)|SCC(a_i)| - 1$, denoted by $w_{\langle i,j\rangle}$. We show that there is no simple directed cycle with negative total weight in G_M. Assuming the necessary renaming of variables, let $a_1, ..., a_k, a_{k+1} = a_1$ be a simple directed cycle in G_M. Then $a_1, ..., a_k, a_{k+1} = a_1$ is also a simple directed cycle in $\mathcal{G}^*_\alpha$, the vertex elimination graph of $\text{DG}^+(P)$. Therefore, according to Lemma 1 we have $\sum_{i=1,...,k} M(x'_{\langle i,i+1\rangle}) \leq k - 1$. On the other hand, we have: $\sum_{i=1,...,k} w_{\langle i,i+1\rangle} = -k + k|SCC(a_i)| - |SCC(a_i)| \sum_{i=1,...,k} M(x'_{\langle i,i+1\rangle}) \geq -k + k|SCC(a_i)| - (k-1)|SCC(a_i)| = |SCC(a_i)| - k \geq 0$. Now, let $N(a_i)$ be the minimum total weight over all simple paths leading to a_i in G_M. Since there is no simple directed cycle with negative total weight in G_M, N is well-defined. Moreover, for every $\langle i, j\rangle$ in $\text{DG}^+(P)$, since there is an arc with weight $(-M(x'_{\langle i,j\rangle}) + 1)|SCC(a_i)| - 1$ from a_i to a_j, we have $N(x_i) - N(x_j) \geq (M(x'_{\langle i,j\rangle}) - 1)(|SCC(a_i)|) + 1$. □

Theorem 3. *If LP^{ve}_P is feasible, so is LP^o_P.*

Proof. We construct $\text{LP}^{o,M}_P$, where M is a model of LP^{ve}_P. According to Lemma 2, $\text{LP}^{o,M}_P$ has some model N such that $N(a) = M(a)$ for every $a \in \text{Var}(\text{IP}^r_P)$. Therefore, (6) to (11) are satisfied by N. According to (16), for all arcs $\langle i, j\rangle$ of $\text{DG}^+(P)$, we have $M(x'_{\langle i,j\rangle}) \geq M(x_{\langle i,j\rangle}) = N(x_{\langle i,j\rangle})$. Therefore, for every arc $\langle a_i, a_j\rangle$ in $\text{DG}^+(P)$, we have $N(x_i) - N(x_j) \geq (M(x'_{\langle i,j\rangle}) - 1)(|SCC(a_i)|) + 1 \geq (N(x_{\langle i,j\rangle}) - 1)(|SCC(a_i)|) + 1$. Thus N satisfies (12). □

Theorem 3 shows that LP^{ve}_P is at least as restrictive as LP^o_P. Moreover, Lemma 2 shows that if LP^{ve}_P is feasible, then for every single model M of LP^{ve}_P, the linear program $\text{LP}^{o,M}_P$, which is at least as restrictive as LP^o_P, is feasible. This suggests that using $\text{IP}^r_P \cup \text{IP}^{ve}_P$ could often result in a more efficient pruning of the search space compared to $\text{IP}^r_P \cup \text{IP}^o_P$. By an example we demostrate how LP^{ve}_P could easily be strictly more restrictive than LP^o_P.

Example 3. *Consider the program presented in Example 2. Since the positive dependency graph of P has only two arcs $\langle a_1, a_3\rangle$ and $\langle a_3, a_1\rangle$, the vertex elimination process does not add any other arcs. Let $M(x_{\langle 1,3\rangle}) = 0.5$ and $M(x_{\langle 3,1\rangle}) = 0.6$. Because of inequalities (16) and (18), M cannot be a model for LP^{ve}_P. Since there is no other node in the same SCC, then M can neither be a model for LP^o_P. However, if there was one other node a_4 in the same*

SCC, then, regardless of the values assigned by M to variables relevant to x_4, inequalities in the form of (12) become $x_1 - x_3 \geq -0.5$ and $x_3 - x_1 \geq -0.2$, which can be satisfied by letting $M(x_1) = M(x_3) = 0$. From this example it should also be clear how larger SCCs weaken the linear relaxation of inequality (12) even more. ■

6 Acyclicity via Feedback Vertex Sets

As mentioned in Section 5, the asymptotic number of arcs added by vertex elimination is $\mathcal{O}(\delta|V|)$, where δ and V represent the elimination width and the number of vertices, respectively. In cases where both δ and $|V|$ are high, our vertex elimination based method may generate integer programs of considerable size, potentially negating the benefits of its more informative linear relaxation.

To address this issue, we introduce an alternative method in this section. This approach utilizes the same set of variables as in LP^o_P but imposes (implicit) constraints on the domains of variables involved in cycle detection. We achieve this by taking advantage of feedback vertex sets (FVSs). FVSs have previously been used in SAT-based ASP to reduce the number of variables while guaranteeing that number of models remains fixed [Hecher and Kiesel, 2023]. Here, however, we use FVSs to restrict the possible values of integer variables, while maintaining the satisfiability (or optimal value of the objective function) of our translation.

Let F be any feedback vertex set of $\text{DG}^+(P)$. We construct integer program IP^f_P by adding for every $a_i \in F$ and $\langle a_i, a_j \rangle \in \text{DG}^+(P)$, the inequality

$$x_i - x_j \geq (x_{\langle i,j \rangle} - 1)(|F \cap SCC(a_i)|) + 1, \qquad (20)$$

and for every $a_i \notin F$ and $\langle a_i, a_j \rangle \in \text{DG}^+(P)$, the inequality

$$x_i - x_j \geq (x_{\langle i,j \rangle} - 1)(|F \cap SCC(a_i)|). \qquad (21)$$

Theorem 4. *If $\text{IP}^r_P \cup \text{IP}^f_P$ has a model M then M is an acyclic model of IP^r_P. If N is an acyclic model for IP^r_P then there is an extension of N that is a model for $\text{IP}^r_P \cup \text{IP}^f_P$.*

Proof. Let M be a model for $\text{IP}^r_P \cup \text{IP}^f_P$. Then M is a model of IP^r_P. Assume that M is not acyclic. Then, assuming the necessary renaming of variables, the graph induced by $\{\langle a_i, a_j \rangle \mid M(x_{\langle i,j \rangle}) = 1\}$ has a cycle $a_1, ..., a_k, a_1$, and thus, $M(x_{\langle 1,2 \rangle}) = ... = M(x_{\langle k,1 \rangle}) = 1$. Since F is a feedback vertex set for $\text{DG}^+(P)$, at least one of $a_1, ..., a_k$, say a_j, must be in F. We can assume that $j \neq k$, otherwise we reorder the indices to make this assumption hold. According to (20), $M(x_j) > M(x_{j+1})$ and according to (20) and (21), $M(x_1) \geq M(x_2) \geq ... \geq M(x_k) \geq M(x_1)$. We conclude that $M(x_1) > M(x_1)$, a contradiction.

Let N be an acyclic model for IP^r_P. We construct a model N' for $\text{IP}^r_P \cup \text{IP}^f_P$ as an extension of N by letting $N'(a) = N(a)$ for $a \in \text{Var}(\text{IP}^r_P)$. Let G_N be a weighted graph with the same vertices as in $\text{DG}^+(P)$. For every arc $\langle a_i, a_j \rangle$ in $\text{DG}^+(P)$, we add $\langle a_i, a_j \rangle$ to G_N iff $N(x_{\langle i,j \rangle}) = 1$. Every arc $\langle a_i, a_j \rangle$ of G_N is assigned a weight of one if $a_j \in F$, and a weight of zero otherwise. Since N is acyclic, G_N is a weighted DAG, and therefore, the set of all directed paths leading to any given vertex is finite. Now, for every vertex x_i we set $N'(x_i)$ to the maximum total weight over all directed paths leading to a_i in G_N. We show that N' satisfies inequality (20) and (21). If $N(x_{\langle i,j \rangle}) = 0$, (20) and (21) are satisfied because, (i) if $N'(a_i) = 0$, a_i has no incoming arc in G_N, and no path to a_j can have a total weight greater than $|F \cap SCC(a_i)| - 1$; and (ii) by the definition of N', for every vertex $a_i \in F$ we have: $0 \leq N'(x_i) \leq |F \cap SCC(a_i)|$. On the other hand, if $N(x_{\langle i,j \rangle}) = 1$, there is an arc in G_N from a_i to a_j. Then N' satisfies inequality (20) because if $a_j \in F$, the weight of $\langle a_i, a_j \rangle$ is one. Similarly, inequality (21) holds because if $a_j \notin F$, the weight of $\langle a_i, a_j \rangle$ is zero. □

From the proof of Theorem 4, it should be clear that for every model M of $\text{IP}^r_P \cup \text{IP}^f_P$, $M(x_i)$ is at most $|F \cap SCC(a_i)|$, which is a lower bound of $|SCC(a_i)|$, the bound necessary in $\text{IP}^r_P \cup \text{IP}^o_P$ for x_i.

7 Empirical Evaluation

We have taken the Acyc2solver tool from the ASPTOOLS collection[1], and incorporated our encoding methods into it. With the exception of acyclicity encoding, all other features of the Acyc2solver tool remain unaltered in our implemented translators. Consequently, the translation results for all ASP rules in both our translators' outputs and the original Acyc2solver tool's output are identical. We have added the resulting translators to the ASPTOOLS collection.

All experiments were conducted on a Linux cluster featuring Intel Xeon 2.40 GHz CPUs, employing a timeout of 600 seconds per problem and a memory limit of 8 GB. For our vertex elimination based method, the order of elimination was determined using the minimum degree heuristic. This involves eliminating a vertex with the minimal total number of incoming and outgoing arcs in the graph produced after eliminating previously processed vertices. In our feedback vertex set based method, we utilized the feedback vertex set generated by an implementation of Corollary 1.

We assess our two translations: the vertex elimination based translation *VE* and the feedback vertex set based translation *FVS*. Our opponents include the original implementation of the Acyc2solver tool (*A2S*) and Clingo 5.4.0 [Gebser *et al.*, 2019], a prominent native ASP solver, using both Branch and Bound (BB) and Unsatisfiable Core (USC) guided optimization strategies of Clingo for optimization problems. Additionally, we employ Gurobi Optimizer version 11.0.0[2] and IBM ILOG CPLEX Optimization Studio 20.1[3] as IP solvers, known for their efficiency. The number of available threads has been set to one for all solvers. Although both Gurobi and CPLEX offer a variety of parameters to control the search, we only use the default ones. Therefore, the solvers' parameters have not been tuned to produce the best performance for our new methods.

The benchmark set comprises non-tight problem sets from previous ASP competitions. A problem is considered non-

[1] https://github.com/asptools/software

[2] https://www.gurobi.com/solutions/gurobi-optimizer

[3] https://www.ibm.com/products/ilog-cplex-optimization-studio

Domain	Gurobi			CPLEX			Clingo	
	VE	FVS	A2S	VE	FVS	A2S	BB	USC
Bayes(60)	60	44	44	44	38	38	41	32
Connect(120)	19	21	19	10	1	0	13	81
Markov(60)	52	10	9	17	6	5	32	0
TSP(30)	29	29	29	25	29	29	0	0
Valve(318)	24	23	23	22	22	22	56	25
Total(588)	184	127	124	118	96	94	142	138

Table 1: Numbers of problems solved in the optimization category

Domain	Gurobi			CPLEX			Clingo
	VE	FVS	A2S	VE	FVS	A2S	
Comb(99)	46	39	21	46	18	15	65
Ham(300)	170	151	151	182	153	153	199
Knight(300)	26	279	279	76	278	278	37
Lab(246)	14	12	1	120	77	10	209
Maz(50)	35	27	11	39	19	5	50
Rand(14)	4	4	4	4	3	3	14
Total(1009)	295	512	447	467	548	464	574

Table 2: Numbers of problems solved in the decision category

tight if its positive dependency graph has at least one non-trivial strongly connected component. It is important to note that for tight problems, our translations are not of particular interest, as they yield results identical to the original Acyc2solver tool. Our evaluation encompasses both decision and optimization problem sets.

The optimization problem sets (with their corresponding competition year in parantheses) consist of *Bayesian Network Learning (2017)*, *Connected Maximum-Density Still Life (2015)*, *Markov Network Learning (2017)*, *Traveling Salesman Problem (2017)*, and *Valves Location Optimization (2015)*, abbreviated as *Bayes*, *Connect*, *Markov*, *TSP*, and *Valve* in our table presentations, respectively. The decision problem sets include *Combined Configuration (2015)*, *Knight Tour With Holes (2015)*, *Labyrinth (2015)*, *Maze Generation (2011)*, and *Random NonTight (2007)*, shortened as *Comb*, *Knight*, *Lab*, *Maze*, and *Rand*. It is important to note that we have not included the *Steiner Tree* problem set from the 2017 competitions in our experiments. This is due to the acyc2solver tool's inability to generate the integer programs within the specified time and memory constraints for this benchmark. Additionally, we incorporate the Hamiltonian cycle encoding from [Niemelä, 1999], denoted as *Ham*, which comprises 30 randomly generated *planar graphs* with $60, 70, \ldots, 150$ nodes, totaling 300 instances. The numbers of problems solved for optimization and decision problem sets are detailed in Tables 1 and 2, respectively.

Several observations can be drawn from our empirical results. A comparison between Gurobi and CPLEX in Tables 1 and 2 reveals that Gurobi performs better for optimization problems, while CPLEX exhibits superior performance for decision problems. When contrasting our translators (*VE* and *FVS*) with *A2S*, a consistent trend emerges: *VE* outperforms *FVS* and *A2S* when coupled with Gurobi in optimization problems, and with CPLEX in decision problems. An exception to this trend is the *Knight Tour With Holes* problem set, where both *FVS* and *A2S* significantly outperform *VE*. This anomaly is attributed to the large elimination widths (ranging in the order of hundreds, the largest among all problem sets) and substantial SCC sizes (ranging in the order of thousands, also the largest among all problem sets) in the positive dependency graphs of this problem set. This results in adding hundreds of thousands of extra variables to the integer program when vertex elimination is used, explaining the low efficiency of the solvers. Additionally, it is noteworthy that *FVS* solves at least as many problems as *A2S* across all problem sets, outperforming it in several of them.

Table 1 illustrates that *VE* with Gurobi as the solver exhibits competitive performance compared to both *BB* and *USC* strategies of Clingo. However, Clingo demonstrates superior efficiency compared to *VE* and other integer programming based methods when solving decision problems, as depicted in Table 2. The primary advantage of IP methods lies in their exploitation of the branch and cut search strategy using linear relaxation. However, this strategy is less advantageous in integer problems without an objective function, where linear relaxation is informative only if it is not feasible. Thus, the gains from using these methods are expected to be less prominent in the decision problem set. This study represents an effort to bridge the efficiency gap between these two paradigms of solving answer set programs, leaving further investigation in this direction for future research.

8 Conclusions

In this paper, we investigate novel translation techniques that could be used to speed up the computation of answer sets when logic programs are translated into integer programs and the respective IP solvers, such as CPLEX and Gurobi, are used for computations. The latest ASP competition results [Gebser *et al.*, 2017; Gebser *et al.*, 2020] suggest that the performance of CPLEX is quite much behind the native ASP solvers on benchmarks in the optimization categories of the competitions. However, based on the experimental results of Section 7—as collected in Tables 1 and 2—the performance gap seems to be closing. In our view, this study serves as a validation for a key motivation behind translation-based Answer Set Programming, emphasizing the utilization of the strengths from both knowledge representation and constraint solving domains.

Besides promising experimental results, this paper has a number of technical contributions that may turn out very useful otherwise. First of all, we present new ways to encode the acyclicity constraint when embedded in integer programming, but the idea is more generic and potentially applicable in other contexts as well. Second, the exploitation of feedback vertex sets in the simplification of the translation opens up new avenues for further research. In particular, it is an interesting generalization of [Chen *et al.*, 2008] where loops with only one externally supporting rule are distinguished.

Acknowledgements

This research is supported by the Research Council of Finland project XAILOG (#345633).

References

[Bellman, 1958] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:78–90, 1958.

[Bomanson *et al.*, 2016] Jori Bomanson, Martin Gebser, Tomi Janhunen, Benjamin Kaufmann, and Torsten Schaub. Answer set programming modulo acyclicity. *Fundam. Informaticae*, 147(1):63–91, 2016.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Chen *et al.*, 2008] Xiaoping Chen, Jianmin Ji, and Fangzhen Lin. Computing loops with at most one external support rule. In Gerhard Brewka and Jérôme Lang, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 401–410. AAAI Press, 2008.

[Gebser *et al.*, 2017] Martin Gebser, Marco Maratea, and Francesco Ricca. The sixth answer set programming competition. *J. Artif. Intell. Res.*, 60:41–95, 2017.

[Gebser *et al.*, 2019] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.*, 19(1):27–82, 2019.

[Gebser *et al.*, 2020] Martin Gebser, Marco Maratea, and Francesco Ricca. The seventh answer set programming competition: Design and results. *Theory Pract. Log. Program.*, 20(2):176–204, 2020.

[Hecher and Kiesel, 2023] Markus Hecher and Rafael Kiesel. The Impact of Structure in Answer Set Counting: Fighting Cycles and its Limits. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, pages 344–354, 8 2023.

[Hunter and Kreutzer, 2008] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.*, 399(3):206–219, 2008.

[Janhunen, 2006] Tomi Janhunen. Some (in)translatability results for normal logic programs and propositional theories. *J. Appl. Non Class. Logics*, 16(1-2):35–86, 2006.

[Liu *et al.*, 2012] Guohua Liu, Tomi Janhunen, and Ilkka Niemelä. Answer set programming via mixed integer programming. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012.

[Mitchell, 2002] John Mitchell. *Branch-and-Cut Algorithms for Combinatorial Optimization Problems*. Oxford University Press, 01 2002.

[Niemelä, 1999] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3-4):241–273, 1999.

[Rankooh and Janhunen, 2022] Masood Feyzbakhsh Rankooh and Tomi Janhunen. Efficient computation of answer sets via SAT modulo acyclicity and vertex elimination. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors, *Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings*, volume 13416 of *Lecture Notes in Computer Science*, pages 203–216. Springer, 2022.

[Rankooh and Rintanen, 2022] Masood Feyzbakhsh Rankooh and Jussi Rintanen. Propositional encodings of acyclicity and reachability by using vertex elimination. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, 2022 Virtual Event, February 22 - March 1, 2022*, pages 5861–5868. AAAI Press, 2022.

[Rose and Tarjan, 1975] Donald J. Rose and Robert Endre Tarjan. Algorithmic aspects of vertex elimination. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing*, pages 245–254, 1975.

[Simons *et al.*, 2002] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.

[Zhou *et al.*, 2023] Neng-Fa Zhou, Ruiwei Wang, and Roland H. C. Yap. A comparison of SAT encodings for acyclicity of directed graphs. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, pages 30:1–30:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.