Advances in Logic-Based Entity Resolution: Enhancing ASPEN with Local Merges and Optimality Criteria

Zhiliang Xiang[1] , **Meghyn Bienvenu**[2] , **Gianluca Cima**[3] ,
Víctor Gutiérrez-Basulto[1] , **Yazmín Ibáñez-García**[1]
[1]Cardiff University, UK
[2]Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France
[3]Sapienza University of Rome, Italy
{xiangz6,gutierrezbasultov,ibanezgarciay}@cardiff.ac.uk, meghyn.bienvenu@labri.fr,
cima@diag.uniroma1.it

Abstract

We present ASPEN$^+$, which extends an existing ASP-based system, ASPEN, for collective entity resolution with two important functionalities: support for local merges and new optimality criteria for preferred solutions. Indeed, ASPEN only supports so-called global merges of entity-referring constants (e.g. author ids), in which all occurrences of matched constants are treated as equivalent and merged accordingly. However, it has been argued that when resolving data values, local merges are often more appropriate, as e.g. some instances of 'J. Lee' may refer to 'Joy Lee', while others should be matched with 'Jake Lee'. In addition to allowing such local merges, ASPEN$^+$ offers new optimality criteria for selecting solutions, such as minimizing rule violations or maximizing the number of rules supporting a merge. Our main contributions are thus (1) the formalization and computational analysis of various notions of optimal solution, and (2) an extensive experimental evaluation on real-world datasets, demonstrating the effect of local merges and the new optimality criteria on both accuracy and runtime.

1 Introduction

Entity Resolution (ER) is a foundational task in computer science, concerned with identifying and merging references (constants) that refer to the same real-world entity (Singla and Domingos 2006). A variety of ER approaches have been explored, differing in their assumptions, the characteristics of the data they handle, and the techniques they employ (Christophides et al. 2021). One general and expressive variant, known as *collective ER*, involves the joint resolution of entity references across multiple interrelated tables (Bhattacharya and Getoor 2007). Declarative approaches have proven particularly well-suited for such complex, multi-relational settings, as they naturally leverage underlying relational dependencies (Bienvenu, Cima, and Gutiérrez-Basulto 2022; Deng et al. 2022; Fagin et al. 2023; Xiang et al. 2024). Nonetheless, and somewhat surprisingly, there are still relatively few logic-based systems that fully support collective ER (Deng et al. 2022; Xiang et al. 2024).

One such system is ASPEN (Xiang et al. 2024), recently developed within the knowledge representation and reasoning community using answer set programming (ASP) (Brewka, Eiter, and Truszczynski 2011; Gebser et al. 2012; Lifschitz 2019). ASPEN implements the LACE framework (Bienvenu, Cima, and Gutiérrez-Basulto 2022) for collective ER, a logical approach that supports recursive, global, explainable, and constraint-aware ER. ASPEN builds on the foundational result that LACE solutions can be faithfully represented using ASP stable models. It extends this theoretical insight by tackling key implementation challenges. Most notably, it addresses the efficient computation of similarity facts within rule bodies. As proposed in LACE, it computes not only individual ER solutions but also an entire space of set-maximal solutions, enabling the derivation of both possible and certain merges. To support scalable reasoning, ASPEN approximates hard-to-compute merge sets (e.g., possible and certain merges) using lower- and upper-bound variants that retain formal guarantees. ASPEN supports explanatory reasoning using xclingo (Cabalar and Muñiz 2023), offering justifications for each merge via proof trees, making it a justifiable ER system. Experimentally, ASPEN demonstrates strong performance, particularly in multi-relational settings, outperforming existing rule-based systems like Magellan (Konda et al. 2016) and JedAI (Papadakis et al. 2020) in F1-score across several datasets. It effectively leverages recursion to uncover deeper merges, and its optimized similarity computation approach reduces memory usage by up to 99.6 %, when compared to a naive method. Overall, despite higher computation times, ASPEN proves competitive and scalable for complex, real-world ER settings.

The introduction of ASPEN, along with the public release of its code and associated data, opens up new opportunities to further explore and evaluate ASP techniques in the context of ER. In this paper, we introduce **ASPEN$^+$**, which extends ASPEN with two important functionalities: support for local merges and new optimality criteria for preferred solutions. In the original ASPEN, constant identification is global: all occurrences of matched constants are merged, not just those involved in the derivation of the match. This global semantics is particularly well suited for merging constants that serve as entity references, such as author names or paper IDs, and has been adopted in several logic-based frameworks (Arasu, Ré, and Suciu 2009; Burdick et al. 2016; Deng et al. 2022; Fagin et al. 2023;

Bienvenu, Cima, and Gutiérrez-Basulto 2022). However, it has been argued that when resolving data values, local merges are often more appropriate (Bertossi, Kolahi, and Lakshmanan 2013; Fan 2008; Fan et al. 2009; Bienvenu et al. 2023; Fagin et al. 2023). Indeed, a local semantics allows some instances of 'J. Lee' may refer to 'Joy Lee', while others should be matched with 'Jake Lee'. ASPEN$^+$ also introduces additional optimality criteria for defining preferred solutions, extending beyond the approach used in ASPEN, which prioritizes solutions with the greatest number of merges (w.r.t. set inclusion). It supports seven optimality criteria, such as minimizing rule violations or maximizing the number of rules supporting a merge. Our main contributions are thus (1) the formalization of various notions of optimal solution and computational analysis of the problem of recognizing optimal solutions w.r.t. the chosen optimality criteria, and (2) an extensive experimental evaluation on real-world datasets, demonstrating the effect of local merges and the new optimality criteria on both accuracy and runtime.

Related Work Several declarative systems for ER have been proposed, including Magellan (Konda et al. 2016) and JedAI (Papadakis et al. 2020). However, most of these systems are designed for pairwise ER. Apart from ASPEN, the most closely related logic-based ER approaches supporting multi-relational settings and recursion are Dedupalog (Arasu, Ré, and Suciu 2009) and MRL (Deng et al. 2022). Nevertheless, these implementations are not publicly available and differ from ASPEN$^+$ in key aspects.: i) ASPEN$^+$ explores a space of preferred solutions under multiple optimality criteria, rather than producing a single solution; ii) it supports both local and global merges, unlike these systems which are limited to global merges only. Note that, like ASPEN, ASPEN$^+$ is an implementation of the LACE$^+$ framework (Bienvenu et al. 2023), which extends LACE to support local merges. In addition, ASPEN$^+$ goes beyond LACE$^+$ by exploring a range of optimality criteria. We also note in passing that the framework proposed by Fagin et al. 2023, like LACE$^+$, supports both global and local merges, although no system currently implements it.

Programs, code and data of ASPEN$^+$ and an extended version with appendix are available at (Xiang 2025) .

2 Preliminaries

Databases We assume an infinite set of *constants* $\mathbf{C}$ partitioned into three disjoint sets: a set of *objects*, $\mathbf{O}$, serving as references to real-world entities, a set $\mathbf{V}$ of *values*, and a set $\mathbf{TID}$ of tuple identifiers (tids), used to annotate database facts. A *(database) schema* $\mathcal{S}$ consists of a finite set of relation symbols $\mathbf{R}$, each having an associated arity $k \in \mathbb{N}$ and type vector $\{\mathbf{O}, \mathbf{V}\}^k$. We write $R/k \in \mathcal{S}$ to indicate that R has arity k and denote by $\mathbf{type}(R, i)$ the ith element of R's type vector. If $\mathbf{type}(R, i) = \mathbf{O}$ (resp. $\mathbf{V}$), we call i an *object (resp. value) position* of R.

A $\mathbf{TID}$*-annotated database over a schema* $\mathcal{S}$ is a finite set of facts D of the form $R(c_0, c_1, \ldots, c_k)$, where $R/k \in \mathcal{S}$, $c_0 \in \mathbf{TID}$ and $c_i \in \mathbf{O} \cup \mathbf{V}$ for $1 \leq i \leq k$. We require that each $t \in \mathbf{TID}$ occurs at most once in D. For simplicity, we refer to such a set of facts as a *database*. We write $\mathsf{Obj}(D)$ to denote the set of objects occurring in D and $\mathsf{Cells}(D)$ to denote the set of values cells of D: $\{\langle t, i\rangle \mid R(t, c_1, \ldots, c_k) \in D, \mathbf{type}(R, i) = \mathbf{V}\}$.

Queries A *conjunctive query (CQ)* takes the form $q(\vec{x}) = \exists \vec{y}.\varphi(\vec{x}, \vec{y})$, where $\vec{x}$ and $\vec{y}$ are disjoint tuples of variables, and $\varphi(\vec{x}, \vec{y})$ is a conjunction of relational atoms of the form $R(t_0, t_1, \ldots, t_k)$ where R/k is a relation symbol and $t_i \in \mathbf{C} \cup \vec{x} \cup \vec{y}$ for $0 \leq i \leq k$, as well as similarity atoms.

Constraints We will consider *denial constraints* (DC) of the form $\forall \vec{x}.\neg\varphi(\vec{x})$, where $\varphi(\vec{x})$ is a conjunction of atoms which are either relational atoms or inequality atoms $t_1 \neq t_2$ with variables from $\vec{x}$. A database D *satisfies a DC* if and only if $\exists \vec{x}.\varphi(\vec{x})$ is not satisfied in D. Notably, DCs generalize the well-known class of functional dependencies (FDs).

3 LACE$^+$ Entity Resolution Framework

In this section, we briefly recall LACE$^+$ and introduce new optimality criteria for selecting preferred ER solutions. Due to space constraints, the presentation focuses solely on the material relevant to ASPEN$^+$. For a complete account, we refer the interested reader to (Bienvenu et al. 2023).

3.1 ER Specifications

Entity resolution is concerned with identifying pairs of syntactically distinct database constants that refer to the same thing (such pairs will be called *merges*). In LACE$^+$, a distinction is made between objects and values, and thus the attributes of database relations are typed as object or value attributes. Objects are references to real-world entities, e.g. paper and author identifiers, whereas values specify the properties of such entities, e.g. name and phone number of authors. Crucially, all occurrences of an object (possibly across different database tables) are assumed to refer to the same entity. This means that if objects o and o' are deemed to be the same, then we merge *all* occurrences of o and o', while the meaning of a value can depend on its context, so e.g. one occurrence of a value *J. Lee* might be merged with *Joy Lee*, while another could merge with *Jack Lee* (or remain unmerged). For this reason, objects and values need to be handled differently during the ER process.

Rules for Objects We use *hard* and *soft* rules for objects to identify pairs of objects that must or might refer to the same real-world entity. Such rules have the following forms:

$$q(x, y) \Rightarrow \mathsf{EqO}(x, y) \qquad q(x, y) \dashrightarrow \mathsf{EqO}(x, y)$$

where $q(x, y)$ is a CQ whose free variables x and y occur only in object positions. Note that EqO is a special relation symbol used to store the merged pairs of objects. While hard and soft rules have essentially the same syntactic form (except for the type of arrow), they differ in meaning: hard rules indicate pairs of objects that *must* be merged, whereas soft rules indicate likely merges.

Rules for Values We use *hard and soft rules for values* to handle local identifications of values which are non-identical representations of the same information. Such rules take the following forms (using $\Rightarrow$ for hard rules, $\dashrightarrow$ for soft rules):

$$q(x_t, y_t) \rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle) \quad \rightarrow \in \{\Rightarrow, \dashrightarrow\}$$

where $q(x_t, y_t)$ is a CQ whose variables x_t and y_t each occur once in q in position 0 of atoms with relations R_x and R_y respectively, and i and j are value positions of R_x and R_y. Here the special relation symbol EqV is used to store pairs of *value cells* (i.e. tuple-position pairs) which have been merged. Intuitively, the above hard (resp. soft) rule states that a pair of tids (t_1, t_2) being an answer to q provides sufficient (resp. reasonable) evidence for concluding that the values in cells $\langle x_t, i\rangle$ and $\langle y_t, i\rangle$ are non-identical representations of the same information.

Specifications In LACE$^+$, ER specifications consist of hard and soft rules for objects and values, together with a set of denial constraints that define what counts as a consistent database.

Definition 1. *A* LACE$^+$ entity resolution (ER) specification Σ *takes the form* $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$*, where* $\Gamma_O = \Gamma_h^o \cup \Gamma_s^o$ *is a finite set of hard and soft rules for objects,* $\Gamma_V = \Gamma_h^v \cup \Gamma_s^v$ *is a finite set of hard and soft rules for values, and* Δ *is a finite set of denial constraints.*

Example 1. *Figure 1b shows a specification* Σ*. The soft rule* σ_o^1 *suggests that author records with similar names and matching birth date and place (*dob*,* pob*) likely refer to the same author. The hard rule* ρ_v^1 *enforces merging similar names for identical author IDs, preserving the functional dependency from constraint* δ_1*.*

3.2 ER Solutions

In LACE$^+$, a solution to an ER specification Σ and database D takes the form of a pair $\langle E, V\rangle$, where E is an equivalence relation over $\mathsf{Obj}(D)$ and V is an equivalence relation over $\mathsf{Cells}(D)$. Solutions thus specify which pairs of objects and which pairs of value cells are deemed to be the same. Naturally, the rules and constraints of the specification impose requirements on E and V. Every solution must be obtained by repeatedly applying the hard and soft rules for objects and values, choosing to stop when no further hard rule can be applied and the denial constraints are satisfied. Importantly, the database and equivalence relations are updated to reflect the already applied rules, which can lead to new rules being applicable and/or constraints becoming (un)satisfied.

The formal definition of solution relies upon the notion of an *extended database* $D_{E,V}$ *induced by* $\langle E, V\rangle$. Basically, every occurrence of an object o in the original database D is replaced by the set of objects $\{o' \mid (o, o') \in E\}$ and the value in cell $\langle t, i\rangle \in \mathsf{Cells}(D)$ is replaced by the set of values occurring in the merged cells $\{\langle t', i'\rangle \mid (\langle t, i\rangle, \langle t', i'\rangle) \in V\}$ (the tuple identifiers in position 0 are left untouched). The semantics of queries and constraints are then suitably adapted to handle such extended databases, whose cells now contain sets of constants rather than single constants (we refer readers to (Bienvenu et al. 2023) for the details). We say that a pair of objects (o, o') is *active* in $D_{E,V}$ w.r.t. a (hard or soft) rule for objects $q(x, y) \rightarrow \mathsf{EqO}(x, y)$ just in the case that (o, o') is an answer to $q(x, y)$ evaluated over $D_{E,V}$. We also say that $D_{E,V}$ *satisfies* a (hard or soft) rule for objects $r = q(x, y) \rightarrow \mathsf{EqO}(x, y)$ when all the pairs *active* in $D_{E,V}$ w.r.t. r occur in E. We can define in a similar fashion what it means for a pair $(\langle t, i\rangle, \langle t', i'\rangle)$ to be active in $D_{E,V}$ w.r.t. a rule for values as well as when $D_{E,V}$ *satisfies* a (hard or soft) rule for values. We write $D_{E,V} \models \Lambda$ when $D_{E,V}$ satisfies each $\lambda \in \Lambda$. Finally, we will use the notation $\mathsf{EqRel}(P, S)$ for the smallest equivalence relation on S that extends $P \subseteq S \times S$. We are now ready to recall the formal definition of solutions in LACE$^+$.

Definition 2. *Given an ER specification* $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$ *and a database* D*, we call* $\langle E, V\rangle$ *a* candidate solution for (D, Σ) *if it satisfies one of the following three conditions:*

- $E = \mathsf{EqRel}(\emptyset, \mathsf{Obj}(D))$ *and* $V = \mathsf{EqRel}(\emptyset, \mathsf{Cells}(D))$*;*
- $E = \mathsf{EqRel}(E' \cup \{(o, o')\}, \mathsf{Obj}(D))$*, where* $\langle E', V\rangle$ *is a candidate solution for* (D, Σ) *and* (o, o') *is active in* $D_{E,V}$ *w.r.t. some rule* $r \in \Gamma_O$*;*
- $V = \mathsf{EqRel}(V' \cup \{(\langle t, i\rangle, \langle t', i'\rangle)\}, \mathsf{Cells}(D))$*, where* $\langle E, V'\rangle$ *is a candidate solution for* (D, Σ) *and* $(\langle t, i\rangle, \langle t', i'\rangle)$ *is active in* $D_{E,V}$ *w.r.t. some rule* $r \in \Gamma_V$*.*

A solution *for* (D, Σ) *is a candidate solution* $\langle E, V\rangle$ *such that (a)* $D_{E,V} \models \Gamma_h^o$*, (b)* $D_{E,V} \models \Gamma_h^v$*, and (c)* $D_{E,V} \models \Delta$*. We use* $\mathsf{Sol}(D, \Sigma)$ *for the set of solutions for* (D, Σ)*.*

Example 2. *Consider the database* D *and specification* Σ *in Figure 1. A trivial candidate solution is* $\langle E_0, V_0\rangle$*, with* $E_0 = \mathsf{EqRel}(\emptyset, \mathsf{Obj}(D))$ *and* $V_0 = \mathsf{EqRel}(\emptyset, \mathsf{Cells}(D))$*. Note that* (a_1, a_2) *is active in* D_{E_0,V_0} *w.r.t.* σ_o^1*. Let,* $E_1 = \mathsf{EqRel}(E_0 \cup \{(a_1, a_2)\}, \mathsf{Obj}(D))$*. However, this causes a violation of* δ^1 *due to differing names for the same* aid *in* t_1 *and* t_2*. Applying* ρ_v^1 *merges cells* $\langle t_1, 2\rangle$ *and* $\langle t_2, 2\rangle$*, yielding* $V_1 = \mathsf{EqRel}(V_0 \cup \{(\langle t_1, 2\rangle, \langle t_2, 2\rangle)\}, \mathsf{Cells}(D))$*. The resulting induced database satisfies* δ^1*, but* $(\langle t_4, 2\rangle, \langle t_5, 2\rangle)$ *is still active in* D_{E_1,V_1} *w.r.t.* σ_v^2*, allowing the possibility to extend* V_1 *to* $V_2 = \mathsf{EqRel}(V_1 \cup (\langle t_4, 2\rangle, \langle t_5, 2\rangle), \mathsf{Cells}(D))$*. Note that,* $\langle E_0, V_0\rangle$*,* $\langle E_1, V_1\rangle$ *and* $\langle E_1, V_2\rangle$ *are all solutions for* (D, Σ)*, as their corresponding induced databases satisfy all denial constraints and hard rules in the specification* Σ*, whereas* $\langle E_1, V_0\rangle$ *is not, as it induced database violated* δ^1*.*

3.3 Optimality Criteria for ER Solutions

In general, a LACE$^+$ specification may give rise to zero, one, or many solutions. The absence of solutions results from constraint violations that cannot be resolved through allowed merges, while multiple solutions arise from the choice of which soft rule applications to perform (as the constraints may block some combinations of merges). In the original work on LACE$^+$, a notion of maximal solution was introduced to focus on solutions which contain the most merges (w.r.t. set inclusion). While this notion is quite reasonable, other natural optimality criteria can be used to select preferred solutions. Indeed, we may want to give more importance to a merge that is supported by multiple rules, or compare solutions based upon soft rule violations.

To formalize these alternative criteria, we introduce the notation $\mathbf{actP}(D, E, V, \Gamma)$ for the set of all (p, r) such that pair p is active in $D_{E,V}$ w.r.t. rule $r \in \Gamma$. Our proposed optimality criteria are obtained by associating each solution

Author(tid, aid, name, dob, pob)				
tid	aid	name	dob	pob
t_1	a_1	A. Turing	23/07/1912	London
t_2	a_2	Alan Turing	23/07/1912	London
t_3	a_3	Clerk Maxwell	13/06/1831	Edinburgh

Awarded(tid, aid, awrd)		
tid	aid	awrd
t_4	a_1	Smith's Prize(1936)
t_5	a_2	Smith's Prize
t_6	a_3	Smith's Prize

The similarity predicate $\approx$, is such that A. Turing $\approx$ Alan Turing and Smith's Prize $\approx$ Smith's Prize(1936)

(a) Database D over the schema $\mathcal{S} = \{\text{Author}/4, \text{Awarded}/2\}$

$$\delta^1 = \forall x_t, y_t, y, n_1, n_2, d_1, d_2, p_1, p_2.\neg(\text{Author}(x_t, y, n_1, d_1, p_1) \wedge \text{Author}(y_t, y, n_2, d_2, p_2) \wedge n_1 \neq n_2)$$
$$\sigma_o^1 = \exists x_t, y_t, n_1, n_2, d, p.\text{Author}(x_t, y_1, n_1, d, p) \wedge \text{Author}(y_t, y_2, n_2, d, p) \wedge n_1 \approx n_2 \dashrightarrow \mathsf{EqO}(y_1, y_2)$$
$$\rho_v^1 = \exists y, n_1, n_2, d_1, d_2, p_1, p_2.\text{Author}(x_t, y, n_1, d_1, p_1) \wedge \text{Author}(y_t, y, n_2, d_2, p_2) \wedge n_1 \approx n_2 \Rightarrow \mathsf{EqV}(\langle x_t, 2\rangle, \langle y_t, 2\rangle)$$
$$\sigma_v^2 = \exists y, z, w.\text{Awarded}(x_t, y, z) \wedge \text{Awarded}(y_t, y, w) \wedge z \approx w \dashrightarrow \mathsf{EqV}(\langle x_t, 2\rangle, \langle y_t, 2\rangle)$$

(b) LACE$^+$ ER specification $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$, with $\Gamma_O = \{\sigma_0^1\}$, $\Gamma_V = \{\rho_v^1, \sigma_v^2\}$ and $\Delta = \{\delta^1\}$

Author(tid, aid, name, dob, pob)				
tid	aid	name	dob	pob
t_1	$\hat{o}_1$	$\hat{v}_1$	{23/07/1912}	{London}
t_2	$\hat{o}_1$	$\hat{v}_1$	{23/07/1912}	{London}
t_3	$\{a_3\}$	{Clerk Maxwell }	{13/06/1831}	{Edinburgh}

Awarded(tid, aid, awrd)		
tid	aid	awrd
t_4	$\hat{o}_1$	$\hat{v}_2$
t_5	$\hat{o}_1$	$\hat{v}_2$
t_6	$\{a_3\}$	{Smith's Prize}

where $\hat{o}_1 = \{a_1, a_2\}$, $\hat{v}_1 = \{\text{A.Turing, Alan Turing}\}$ and $\hat{v}_2 = \{\text{Smith's Prize, Smith's Prize(1936)}\}$

(c) Extended database $D_{E,V}$ induced by the merges $(a_1, a_2) \in E$, and $(\langle t_1, 2\rangle, \langle t_2, 2\rangle)$, $(\langle t_3, 2\rangle, \langle t_4, 2\rangle) \in V$

Figure 1: A solution to an ER specification in LACE$^+$

$\langle E, V\rangle$ with one of the following sets (using Γ for $\Gamma_O \cup \Gamma_V$):

$$\text{EQ}(E, V) = E \cup V$$
$$\text{SUPP}(E, V) = \{(p, r) \in \mathbf{actP}(D, E, V, \Gamma) \mid p \in E \cup V\}$$
$$\text{ABS}(E, V) = \{p \mid (p, r) \in \mathbf{actP}(D, E, V, \Gamma), p \notin E \cup V\}$$
$$\text{VIOL}(E, V) = \{(p, r) \in \mathbf{actP}(D, E, V, \Gamma) \mid p \notin E \cup V\}$$

then apply either set-inclusion or cardinality maximization or minimization. Observe that $\text{SUPP}(E, V)$ refines $\text{EQ}(E, V)$ by indicating the supporting rules for merges. Likewise, $\text{ABS}(E, V)$ gives only the active but absent pairs, while $\text{VIOL}(E, V)$ records which soft rules the absent pair violates. The resulting optimality criteria are as follows:

- maxES/maxEC: maximize $\text{EQ}(E, V)$
- maxSS/maxSC: maximize $\text{SUPP}(E, V)$
- minAS/minAC: minimize $\text{ABS}(E, V)$
- minVS/minVC: minimize $\text{VIOL}(E, V)$

where the final S (resp. C) indicates comparison using set-inclusion (resp. set cardinality). For example, a solution $\langle E, V\rangle$ is minVC-optimal if there is no other solution $\langle E', V'\rangle$ such that $\text{VIOL}(E', V') < \text{VIOL}(E, V)$. Note that maxES-optimal solutions correspond to the maximal solutions of (Bienvenu et al. 2023). See Appendix A for further intuitions on the criteria. Interestingly, the optimality criteria give rise to different sets of optimal solutions, except for maxES and maxSS, which actually coincide. As a result, there are seven distinct optimality criteria overall.

Proposition 1. *All pairs of defined criteria produce different sets of optimal solutions, except for* maxES *and* maxSS.

3.4 Recognizing Optimal Solutions

A key challenge is determining if a solution is optimal under a given criterion. This problem is coNP-complete in data complexity for maxES (Bienvenu et al. 2023), and we show the same for the other criteria.

Theorem 1. *For all seven optimality criteria, recognition of optimal solutions is coNP-complete in data complexity.*

Proof sketch. The upper bound is based on a guess-and-check procedure, analogous to the one in (Bienvenu et al. 2023) for maxES. We now give a reduction from the complement of 3SAT. We use the following fixed schema:

$$\mathcal{S} = \{R_{fff}/3,\ R_{fft}/3,\ R_{ftf}/3,\ R_{ftt}/3,\ R_{tff}/3,\ R_{tft}/3,\ R_{ttf}/3,\ R_{ttt}/3,\ V/1,\ F/1,\ T/1,\ B/1\}$$

with only object attributes. Given an input $\varphi = c_1 \wedge \ldots \wedge c_m$ to 3SAT over variables $x_1, \ldots, x_n$, where $c_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$, we construct an $\mathcal{S}$-database D^φ that contains:

- the fact $V(x_i)$ for each $i \in [1, n]$,
- the facts $T(1)$, $F(0)$, $B(0)$, and $B(1)$;
- for each clause $c_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ in φ, the fact $R_{b_1 b_2 b_3}(x_{i,1}, x_{i,2}, x_{i,3})$ with $\ell_{i,j} = (\neg)x_{i,j}$ and $b_{i,j} = t$ (resp. $b_{i,j} = f$) if $\ell_{i,j}$ is a positive (resp. negative) literal

We consider the fixed ER specification $\Sigma_{\text{3SAT}} = \langle \Gamma_O, \emptyset, \Delta\rangle$, where Γ_O contains a single soft rule for objects:

- $\sigma = V(x) \wedge B(y) \dashrightarrow \mathsf{EqO}(x, y)$

and Δ contains the following denial constraints (all the variables are implicitly universally quantified):

- $\delta_0 = \neg(F(y) \wedge T(y))$
- $\delta_1 = \neg(R_{fff}(y_1, y_2, y_3) \wedge T(y_1) \wedge T(y_2) \wedge T(y_3))$
- $\delta_2, \ldots, \delta_8$ defined analogously to δ_1, but for relations $R_{fft}, R_{ftf}, R_{ftt}, R_{tff}, R_{tft}, R_{ttf}, R_{ttt}$
- $\delta_9 = \neg(V(v) \wedge B(v) \wedge V(x) \wedge T(y) \wedge F(z) \wedge x \neq y \wedge x \neq z)$

The soft rule σ allows a variable to merge with 0 or 1 (but not both due to δ_0). Constraints δ_1 to δ_8 ensure that the chosen truth values do not violate any clause. Finally, constraint δ_9 is used to ensure that if even one variable is merged with a truth value, using the sole soft rule σ, then *every* variable must be merged with a truth constant. Together, the constraints ensure that the soft rule can only be applied if the formula is satisfiable. It can thus be shown that φ is unsatisfiable if and only if $\langle E_{\mathsf{triv}}, \emptyset \rangle$ is an X-optimal solution for $(D^\varphi, \Sigma_{3\mathrm{SAT}})$, where $E_{\mathsf{triv}} = \mathsf{EqRel}(\emptyset, \mathsf{Obj}(D^\varphi))$ and X denotes any of the seven optimality criteria. □

Interestingly, (Bienvenu et al. 2023) also studied the above problem in a *restricted setting*, which forbids the presence of inequality atoms in denial constraints. It was shown that, under this restriction, the problem becomes tractable for maxES, and in fact PTIME-complete in data complexity. We now show that the same holds for all optimality criteria based on set inclusion, but not for those based on cardinality.

Theorem 2. *In the restricted setting, recognition of optimal solutions becomes PTIME-complete in data complexity for the optimality criteria* maxES*,* minAS*, and* minVS*, while it remains coNP-complete in data complexity for the optimality criteria* maxEC*,* maxSC*,* minAC*, and* minVC*.*

Proof sketch. **PTIME cases.** We provide only a sketch of the upper bound for minAS. Given (D, Σ) and $\langle E, V \rangle$, we check whether $\langle E, V \rangle \in \mathsf{Sol}(D, \Sigma)$ (if not, then we return `false` and we are done). For each $p \in \textsc{abs}(E, V)$, we then check whether $E \cup V \cup \{p\}$ can lead to a solution for (D, Σ) having a strictly smaller set of active pairs than $\textsc{abs}(E, V)$.

To perform this latter check, we iteratively include all the new pairs that become active due to some hard rule (which may already occur in the first iteration due to the addition of p to $E \cup V$) as well as all the new pairs that become active due to some soft rule and were not originally in $\textsc{abs}(E, V)$. Once a fixpoint is reached, we check whether the resulting $\langle E', V' \rangle$ is such that $\langle E', V' \rangle \in \mathsf{Sol}(D, \Sigma)$, implying that $\langle E', V' \rangle$ is a solution such that $\textsc{abs}(E', V') \subsetneq \textsc{abs}(E, V)$. If for some $p \in \textsc{abs}(E, V)$ this is the case, then we return `true`; otherwise, we return `false`. The correctness of the above procedure is an immediate consequence of the following property which holds in the restricted setting: if $D_{E',V'} \not\models \Delta$ and $E' \cup V' \subseteq E'' \cup V''$, then $D_{E'',V''} \not\models \Delta$ as well. In particular, this means that if a pair $\langle E', V' \rangle$ obtained as above is such that $D_{E',V'} \not\models \Delta$, then it is not possible to obtain a solution for (D, Σ) by adding further merges.

coNP cases. We give a reduction from the complement of 3SAT for the optimality criteria based on cardinality. We use the fixed schema $\mathcal{S}' = \mathcal{S} \cup \{H/2\}$ with only object attributes, where $\mathcal{S}$ is as in the proof of Thm 1. Given an input φ to 3SAT over variables $x_1, \ldots, x_n$, we construct an $\mathcal{S}'$-database $\mathcal{D}^\varphi = D^\varphi \cup \{H(c_1, c_2)\}$, where D^φ is as in the proof of Theorem 1 while c_1 and c_2 are two fresh constants.

We consider the fixed ER specification $\Sigma'_{3\mathrm{SAT}} = \langle \Gamma, \Delta \rangle$, where Γ contains the soft rule σ illustrated in the proof of Theorem 1 as well as the following additional soft rule:

- $\sigma' = H(x, y) \dashrightarrow \mathsf{EqO}(x, y)$

and Δ contains the following denial constraints (all the variables are implicitly universally quantified):

- $\delta_0 = \neg(F(y) \wedge T(y))$
- $\delta_1 = \neg(R_{fff}(y_1, y_2, y_3) \wedge T(y_1) \wedge T(y_2) \wedge T(y_3) \wedge H(z, z))$
- $\delta_2, \ldots, \delta_8$ defined analogously to δ_1, but for relations $R_{fft}, R_{ftf}, R_{ftt}, R_{tff}, R_{tft}, R_{ttf}, R_{ttt}$

Essentially, the denial constraints are similar to those in the proof of Theorem 1, except that $\delta_1, \ldots, \delta_8$ also contain the conjunct $H(z, z)$. This latter conjunct holds if and only if c_1 merges with c_2, which is possible due to σ'.

Now, let $E^0 = \{(x_i, 0) \mid 1 \leq i \leq n\}$ and $E = \mathsf{EqRel}(E^0, \mathsf{Obj}(\mathcal{D}^\varphi))$. Note that $\langle E, \emptyset \rangle \in \mathsf{Sol}(\mathcal{D}^\varphi, \Sigma'_{3\mathrm{SAT}})$ (because $(c_1, c_2) \notin E$) and $\textsc{abs}(E, \emptyset) = \{(x_i, 1) \mid 1 \leq i \leq n\} \cup \{(c_1, c_2)\}$ (so, $|\textsc{abs}(E, \emptyset)| = n + 1$). The only way to get a solution $\langle E', \emptyset \rangle$ such that $|\textsc{abs}(E', \emptyset)| < n + 1$ is to merge c_1 and c_2 as well as all the variables with 0 or 1, which is possible if and only if φ is satisfiable. It can thus be shown that φ is unsatisfiable if and only if $\langle E, \emptyset \rangle$ is a minAC-optimal solution for $(\mathcal{D}^\varphi, \Sigma'_{3\mathrm{SAT}})$. Similar considerations apply for minVC, while for maxSC and maxEC we need to introduce a copy of each variable into a new unary predicate, an additional soft rule allowing to merge such copies with 0 or 1, and an additional denial constraints specifying that no variable and its copy are assigned to the same truth value. This guarantees that each maxEC-optimal (resp. maxSC-optimal) solution has the same cardinality. □

4 ASP Encoding and Implementation

We assume basic familiarity with ASP, see (Gebser et al. 2012; Lifschitz 2019) for a detailed introduction. For our purposes, it suffices to focus on two constructs: *normal rules*, which have a single atom in the head, and *constraints*, which are rules with an empty head. We denote an ASP program (a set of such rules) by Π. Given a program Π, its *grounding* is the set of all ground instances of rules in Π, using the constants that appear in Π, and $SM(\Pi)$ denotes the set of all its stable models. In addition to solving the decision problem of checking whether $SM(\Pi)$ is non-empty, ASP-solvers also support advanced reasoning features such as enumerating the first n stable models of Π, projecting models onto a specified set of atoms, and enumerating the first n projected models (Gebser et al. 2018). ASP solvers also support optimization, allowing the computation (or enumeration) of n elements of $SM(\Pi)$ that optimize a specified objective. For example, the directive $\#\texttt{minimize}\{w, \mathbf{t} : \mathbf{L}\}$ instructs the solver to minimize the weighted occurrence of tuples $\mathbf{t}$ with weight w, subject to a list of literals $\mathbf{L}$.

4.1 ASP Encoding of Solutions

Given an ER specification Σ and a database D, we define an ASP program $\Pi_{(D,\Sigma)}$ containing all the facts in D, and an ASP rule for each (hard or soft) rule in Σ. Consider, for example, the specification Σ' in Figure 1. Rules are translated as follows:

$\sigma_o^1 : \{\texttt{eqo}(X,Y); \texttt{neqo}(X,Y)\}= 1 \leftarrow \texttt{author}(T,X,N,D,P), \texttt{author}(T',Y,N',D,P), \texttt{val}(T,2,N_1), \texttt{val}(T',2,N_2), \texttt{sim}(N_1,N_2,S), S \geq 95.$

$\rho_v^1 : \texttt{eqv}(T,2,T',2) \leftarrow \texttt{eqo}(X,Y), \texttt{sim}(N_1,N_2,S), S \geq 95, \texttt{author}(T,X,N,D,P), \texttt{author}(T',Y,N',D',P'), \texttt{val}(T,2,N_1), \texttt{val}(T',2,N_2).$

$\sigma_v^2 : \{\texttt{eqv}(T,2,T',2); \texttt{neqv}(T,2,T',2)\}= 1 \leftarrow \texttt{eqo}(X,Y), \texttt{awarded}(T,X,A), \texttt{awarded}(T',Y,A'), \texttt{val}(T,2,A_1), \texttt{val}(T',2,A_2), \texttt{sim}(A_1,A_2,S), S \geq 95.$

$\delta^1 : \bot \leftarrow \texttt{author}(T,X,N,D,P), \texttt{author}(T',Y,N',D',P'), \texttt{eqo}(X,Y), \texttt{not i}(T,2,T',2).$

In brief, each relational atom in the body of a rule is translated into an atom in the body of an ASP rule. The relations eqo and eqv represent mandatory object and value merges, respectively, thereby encoding solutions to (D,Σ). Atoms of the form $\texttt{eqo}(X,Y)$ in ASP rule bodies are used to encode that instantiations of X and Y are determined to denote the same object. To capture local semantics, the program includes facts of the form $\texttt{proj}(T,i,V)$, indicating that the i-th position of tuple T has value V. Each proj fact is then assigned to a value set, identified by a cell, through local merges. This is formalized by the rule: $\texttt{val}(T,I,V) \leftarrow \texttt{proj}(T,I,V), \texttt{eqv}(T',J,T,I)$. Importantly, similarity measures over value positions are evaluated not on the original variables in the relational atoms, but on pairs of fresh variables associated with val-atoms. The program also includes rules to ensure that both eqo and eqv form equivalence relations. Soft constraints are implemented using choice rules, allowing the inclusion of a pair (X,Y) (or $(T,2,T',2)$, respectively) in either eqo (or eqv) or their respective negations neqo and neqv. Note that, in addition to the encoding format shown above, database atoms (facts) can also be represented in a reified format (see Appendix C.1).

Practical Changes Following ASPEN, we represent similarity between constants using a ternary relation $\texttt{sim}_\texttt{i}(X,Y,S)$, where X and Y are the constants being compared, and S is the similarity score. Null values are represented using the atom $\texttt{empty}(nan)$, in which, also as in ASPEN, the special constant nan is prevented from participating in any eqo (resp. neqo) and sim atoms. Cells with null values are allowed to participate in eqv. However, when equality between value positions is checked in a rule body, e.g., using $\texttt{val}(T,i,V), \texttt{val}(T,j,V)$, we include the condition $\texttt{not empty}(V)$ to exclude nulls from the comparison. This encoding blocks merges that would arise from treating two nulls as equivalent in rule bodies. Moreover, to capture the local semantics of inequalities in DCs, the encoding must ensure that the value sets of the two cells are disjoint. This requirement was overlooked in (Bienvenu et al. 2023). To this end, for each pair of variables v and v' associated with cells $\langle t,i\rangle$ and $\langle t',j\rangle$, we add the following rule to capture value overlap:

$\texttt{i}(T,i,T',j) \leftarrow \texttt{val}(T,i,V), \texttt{val}(T',j,V), \texttt{not empty}(V).$

This rule asserts that the predicate $\texttt{i}(T,i,T',j)$ holds if the two cells share a non-null value—that is, their value sets intersect. Consequently, inequality between the two positions can be expressed by the absence of such an intersection: $\texttt{not i}(T,i,T',j)$ like in δ^1.

4.2 Computing Optimal Solutions

To compute solutions under the maxSC criterion, we extend the predicates eqo and eqv (or neqo and neqv) by adding an extra argument to represent the rule label. Each hard or soft rule in $\Pi_{(D,\Sigma)}$ is assigned a unique label constant, which is included in the head atom to distinguish merges derived from different rules. In rule bodies, any occurrence of eqo or eqv uses an anonymous variable in the label position to avoid hard-coding the label. Finally, for each rule labeled r that applies to objects (respectively, to values), we add a corresponding rule to $\Pi_{(D,\Sigma)}$: $\texttt{w}(X,Y,l,w) \leftarrow \texttt{eqo}(X,Y,r)$. (resp. $\texttt{w}(T,i,T',j,r,w) \leftarrow \texttt{eqv}(T,i,T',j,r)$.). This assigns a weight w to merges derived from rule r.

The transformation for encoding rule violations under the minV criterion follows a similar approach. Specifically, the head atoms of hard rules are annotated with a default label constant r', while those of soft rules are assigned distinct labels. For example, in Section 4.1, the head of the soft rule σ_o^1 is replaced by $\{\texttt{eqo}(X,Y,\sigma_o^1); \texttt{neqo}(X,Y,\sigma_o^1)\}= 1$, similarly for σ_v^2. In contrast, the head of the hard rule ρ_v^1 is expanded with the default label r'. Moreover, rules that aggregate weights (e.g. those required for computing maxSC-optimal solutions) will not be included.

Set-Based Optimisation Set-based optimal solutions correspond to the stable models of $\Pi(D,\Sigma)$ that contain an optimal set of target facts, depending on the chosen criterion: (1) a maximal set of eqo/eqv facts for maxE; (2) a minimal set of (labeled) neqo/neqv facts for minV and minA; (3) a maximal set of weighted facts for maxS. Following (Xiang et al. 2024), we utilise the ASP-based preference framework asprin (Brewka et al. 2023) to encode preferences over the stable models of a program. For our uses, we prefer a model M' over M if its projection on the target facts is a strict superset or subset of that of M, under the selected criterion.

Alternatively, domain-specific heuristics can be used to simulate set preferences by prioritizing the truth assignments of target facts (Rosa, Giunchiglia, and Maratea 2010; Gebser et al. 2013). We encode such heuristics using the statement $\#\texttt{heuristic t.}\ [1,\ \texttt{p}]$, where t is a non-ground atom with the same predicate as the target facts for the chosen criterion, and $\texttt{p} \in \{\texttt{True}, \texttt{False}\}$ enables maximization or minimization, respectively.

Cardinality-Based Optimisation To compute cardinality-optimal solutions, we use the standard ASP optimization statement (Gebser et al. 2012), which minimizes (or maximizes) over the full tuple of arguments in the target atoms, using a default weight and priority of 1. For instance, to compute minVC, we include the following statements: $\#\texttt{minimize}\{1@1, \texttt{X}, \texttt{Y}, \texttt{R} : \texttt{neqo}(\texttt{X}, \texttt{Y}, \texttt{R})\}$

and $\#\texttt{minimize}\{1@1,\texttt{T},\texttt{I},\texttt{T}',\texttt{J},\texttt{R}:\texttt{neqv}(\texttt{T},\texttt{I},\texttt{T}',\texttt{J},\texttt{R})\}$. Alternatively, preferences over the cardinality of target facts can be expressed using asprin, allowing the selection of models with either maximal or minimal cardinality depending on the chosen criterion. It is worth noting that while heuristic-driven solving and multi-threaded solving (Gebser, Kaufmann, and Schaub 2012) can both improve efficiency for optimization tasks, they are not compatible and cannot be used simultaneously.

4.3 Implementation

The workflow of ASPEN$^+$ begins with an ASP encoding of an ER specification, a `CSV`/`TSV` dataset containing duplicate and potentially corrupted values (e.g., nulls), and a set of command-line options. A preprocessing component computes similarity scores between relevant constant pairs based on attributes referenced in similarity atoms. These scores, along with the dataset, are passed to a program transformer, which rewrites the encoding into the ASPEN$^+$ format and generates ASP facts for both database tuples and similarities. Finally, the ER controller, built on `clingo`, computes optimal solutions according to the specified input options. See Appendix C for a workflow illustration.

We implemented ASPEN$^+$ by extending the open-source system ASPEN. Specifically, we developed a new program transformer, based on the translation procedure from (Bienvenu et al. 2023), which enables users to write ASP rules in the style of ER specifications without dealing with the underlying complexities of ASP encoding. Additionally, we enhanced the ER controller to support computing solutions under various optimality criteria, leveraging ASP features such as heuristic-driven solving.

5 Experiments

Our experiments focus on the following questions: (1) How does ASPEN$^+$ perform compared to SOTA systems that rely exclusively on global semantics? (2) What are the most efficient configurations for each optimality criterion? How do the resulting solutions vary in terms of performance? Can these solutions be enumerated efficiently? We also present qualitative studies that highlight the value of integrating both global and local semantics to address complex ER scenarios.

Datasets and Metrics We evaluate ASPEN$^+$ on six multi-relational datasets: (i) *Imdb*, based on the IMDB database, which includes entities such as artists and movies, and contains real-world duplicates; (ii) *Mu*, which features synthetic duplicates derived from the MusicBrainz database; (iii) *MuC*, a noisier version of *Mu* that retains the same duplicates but includes more missing values and syntactic variations; (iv) *Poke*, a larger-scale dataset from the Pokémon database, characterized by a greater number of tuples and more complex inter-table relationships. We also consider two synthetic datasets, *ImdbC* and *MuCC*. These are derived from *Imdb* and *MuC*, respectively, and contain a higher proportion of syntactic variants. Both datasets follow the variant generation protocol described in (Xiang et al. 2024). We use the standard accuracy metrics (Köpcke, Thor, and Rahm 2010) of *Precision* (**P**), *Recall* (**R**), and *F1-Score* ($\mathbf{F_1}$). Sources and statistics for all datasets, as well as details of the experimental environment and similarity measures, are provided in Appendix D.1.

5.1 Combining Global and Local Semantics

Setup We evaluate the performance of ASPEN$^+$ against three baselines: ASPEN and two pairwise rule-based ER systems, Magellan (Konda et al. 2016) and JedAI (Papadakis et al. 2020). Both ASPEN$^+$ and ASPEN use the same global rules from ASPEN, with ASPEN$^+$ additionally incorporating local rules. Functional dependencies (FD) from the dataset schema are included in both ASP-based systems, provided they yield satisfiable ER programs. For Magellan and JedAI, we apply the same preconditions and similarity measures as in the ASP-based systems to ensure comparability. Like in ASPEN, we consider a single maxES-solution as default output of ASPEN$^+$. We report accuracy metrics and total running time $\mathbf{t_o}$ for all systems. For the ASP-based systems, we also report grounding time ($\mathbf{t_g}$), solving time ($\mathbf{t_s}$), and the number of DCs (**#DC**) added without violating constraints. Results are shown in Table 1.

Accuracy ASPEN$^+$ consistently achieved the highest F1 scores across all datasets. Compared to the pairwise baselines, it outperformed Magellan and JedAI by wide margins, with F1 gains ranging from 7.86% to 82% over Magellan, and 1.64% to 77% over JedAI. Similar trends were observed for ASPEN, highlighting the advantage of logic-based approaches that support collective ER and enforce constraints.

When compared to ASPEN, ASPEN$^+$ matched its performance on *Imdb* and *Mu*, but achieved F1 improvements of 2.14%, 0.15%, 17.6%, and 3.2% on *ImdbC*, *MuC*, *MuCC*, and *Poke*, respectively. Except for *Imdb* and *ImdbC* (where inherent constraint violations exist) ASPEN$^+$ successfully incorporated all FDs from the schema. In contrast, ASPEN's reliance on global semantics alone often led to unsatisfiable programs when all FDs were included. Thus, ASPEN$^+$ produced solutions with better precision and recall, particularly on noisy datasets like *MuCC*. Interestingly, even in *Poke*, where both systems could include all FDs, ASPEN$^+$ achieved 4.72% higher recall than ASPEN, demonstrating that without handling alternative value representations, valid merges can still be blocked by DCs.

Running Time The overall time $\mathbf{t_o}$ consists of preprocessing (similarity computation) and ER time. For ASP-based systems, the ER time is further composed of grounding and solving time. ASPEN uses an optimized similarity algorithm for preprocessing, making similarity computation faster on more complex datasets. In contrast, ASPEN$^+$ uses a naive cross-product similarity computation, as ASPEN's optimized algorithm is incompatible with local semantics. Overall, both Magellan and JedAI significantly outperformed the ASP-based systems on $\mathbf{t_o}$. Compared to ASPEN$^+$, Magellan was faster by factors of 22, 30, 10, 11, 15, and 45 on the *Imdb*, *ImdbC*, *Mu*, *ImdbC*, *MuCC*, and *Poke*, respectively. Similarly, JedAI achieved speed advantages of 4.5 to 506 times over the same datasets. Due to differences in the similarity computation, ASPEN$^+$ was 7 and 8 times faster than ASPEN on simpler datasets like *Imdb* and

Method	Data	F_1	(P / R)	t_o	t_g	t_s	#DC	Data	F_1	(P / R)	t_o	t_g	t_s	#DC
Magellan	*Imdb*	88.09	**99.80** 78.83	**3.89**	N/A	N/A	0/5	*ImdbC*	83.05	**99.77** 71.12	**3.11**	N/A	N/A	0/5
JedAI		97.49	99.40 95.67	18.78	N/A	N/A	0/5		97.49	99.40 95.67	18.67	N/A	N/A	0/5
ASPEN		**99.27**	99.39 **99.14**	610.49	**11.51**	**0.082**	1/5		96.99	99.36 94.73	757.17	**11.75**	**0.088**	1/5
ASPEN$^+$		**99.27**	99.39 **99.14**	86.24	13.02	0.096	3/5		**99.13**	99.39 **98.87**	94.85	18.71	0.69	3/5
Magellan	*Mu*	89.78	98.63 82.38	**64.83**	N/A	N/A	0/23	*MuC*	55.54	**97.51** 38.83	**66.87**	N/A	N/A	0/23
JedAI		70.67	87.46 59.30	105.06	N/A	N/A	0/23		32.75	73.95 21.02	101.02	N/A	N/A	0/23
ASPEN		**97.64**	**99.45** **95.89**	666.01	**1.78**	**0.20**	21/23		90.31	91.86 **88.81**	695.76	**20.15**	**3.57**	17/23
ASPEN$^+$		**97.64**	**99.45** **95.89**	665.65	1.82	0.21	23/23		**90.46**	92.17 **88.81**	770.44	72.81	12.16	23/23
Magellan	*MuCC*	55.48	**97.62** 38.75	**64.81**	N/A	N/A	0/23	*Poke*	7.01	3.97 29.74	260.96	N/A	N/A	0/10
JedAI		31.04	72.47 19.75	101.47	N/A	N/A	0/23		2.1	1.08 46.56	**23.46**	N/A	N/A	0/10
ASPEN		71.18	77.83 65.58	718.38	**21.01**	**10.47**	16/23		81.78	99.71 69.31	4,454	**311.37**	**0.91**	10/10
ASPEN$^+$		**88.85**	89.5 **88.21**	993.31	97.51	23.02	23/23		**84.98**	**99.73** **74.03**	11,800	496.14	48.52	10/10

Table 1: Results on Complex Multi-relational Datasets

ImdbC . However, on more complex datasets, the optimised similarity algorithm used by ASPEN gave speed advantages of 1.1, 1.4, and 2.6 times compared to ASPEN$^+$ on *MuC* , *MuCC* , and *Poke* .

In terms of $\mathbf{t_g}$ and $\mathbf{t_s}$, ASPEN consistently outperformed ASPEN$^+$. On *MuC* , *MuCC* , and *Poke* , ASPEN 's grounding was 3.6, 4.6, and 1.5 times faster, respectively. The differences in solving time were even more pronounced: ASPEN$^+$ was 2.1, 3.4, 7.8, and 53 times slower than ASPEN on *MuCC* , *MuC* , *ImdbC* , and *Poke* . These results show that while ASPEN$^+$ delivers higher ER quality, it often does so at the expense of computational efficiency.

5.2 Comparison on Optimality Criteria

Setup For each dataset, we fixed the corresponding ER program and used ASPEN$^+$ to compute one and up to 50 optimal solutions based on the set of optimality criteria defined in Section 3.3. For set-based criteria, heuristic-driven solving was enabled. For cardinality-based criteria, we used weighted constraints and enabled parallel solving with 36 threads. We recorded the average accuracy metrics over the number of enumerated solutions #e as $\overline{F}_1/\overline{P}/\overline{R}$. For efficiency, we measured the solving time to compute the first optimal model (t_s^1) and the average solving time per model t_s^n. For simplicity, we use the prefixes *S-* and *C-* to denote, in general, set-based and cardinality-based optimality criteria, respectively. Results are summarized in Table 2 and Table 3. Appendix D.2 presents results for various configurations under each optimality criterion (e.g., using aspirin vs. using heuristic). The results reported in this section correspond to the most efficient configuration.

Accuracy For the simpler datasets, *Imdb* , *ImdbC* , and *Mu* , we observed that all optimality criteria produced a single optimal solution identical to maxES, resulting in the same $\overline{F}_1$ (see Appendix D.3). This is consistent with findings from (Xiang et al. 2024), indicating that in the presence of data with less corrupted values and nulls, value-based

Data	Method	$\overline{F}_1$	($\overline{P}$ / $\overline{R}$)		t_s^1	#e	t_s^n
MuC	maxES/SS	90.50	92.20	**88.86**	**12.16**	50	1.86
	minAS	**91.88**	**95.11**	**88.86**	12.8	50	**1.7**
	minVS	91.7	94.73	88.85	12.44	50	1.78
MuCC	maxES/SS	88.85	89.5	**88.21**	23.02	50	1.67
	minAS	89.62	91.11	88.18	23.01	50	**1.54**
	minVS	**90.13**	**92.2**	88.15	**21.61**	50	2.48
Poke	maxES/SS	**84.98**	**99.73**	**74.03**	**48.52**	1	N/A
	minAS	83.83	**99.73**	72.3	51.58	50	0.06
	minVS	84.62	**99.73**	73.48	56.87	50	**0.01**

Table 2: Result on optimal solutions under different set-optimisation criteria over the datasets

Data	Method	$\overline{F}_1$	($\overline{P}$ / $\overline{R}$)		t_s^1	#e	t_s^n
MuC	maxEC	90.51	92.20	**88.9**	35.09	50	**2.53**
	maxSC	90.52	92.21	**88.9**	**30.1**	16	12.09
	minAC	91.4	94.05	**88.9**	84.69	50	2.61
	minVC	**92.01**	**95.35**	88.89	48.97	50	5.66
MuCC	maxEC	88.83	89.45	**88.21**	66.71	1	N/A
	maxSC	88.85	89.50	**88.21**	**52.8**	2	629.56
	minAC	89.51	90.93	88.14	92.97	50	**2.12**
	minVC	**89.74**	**91.36**	88.18	67.82	50	8.42
Poke	maxEC	**84.98**	**99.73**	**74.03**	**48.52**	1	N/A
	maxSC	**84.98**	**99.73**	**74.03**	49.1	1	N/A
	minAC	84.80	**99.73**	73.75	49.23	50	**0.037**
	minVC	84.91	**99.73**	73.91	49.11	2	1.15

Table 3: Result on optimal solutions under different cardinality-optimisation criteria over the datasets

comparisons (e.g., similarity or equality) are sufficient for merging, leaving little room for variation. In contrast, for more complex datasets, minVS and minVC achieved the highest accuracy on *MuC* and *MuCC* , outperforming the default maxES by 1.5% and 1.28%, respectively. However, on *Poke* , the opposite trend was observed: maxES slightly outperformed minVS and minVC by 0.36% and 0.07%, resp.
We also observed that solutions based on different optimality criteria differ in the following:
minA & minV vs maxS & maxE. Except for a tie on *Poke* , minA and minV consistently outperformed maxS and maxE in terms of precision. The largest improvements were observed on *MuC* and *MuCC* , with leads of 2.93% and 2.7% for the S-criteria, and 1.5% and 1.9% for the C-optimal solutions, respectively. In contrast, maxS/maxE obtained better recall, with the largest leads of 1.73% seen when comparing maxES/SS to minAS on *Poke* . This highlights the different characteristics between minA/minV and maxS/maxE, while the former prioritizes precision, the latter is more inclusive and has better coverage. As a result, on datasets with more corrupted values like *MuC* and *MuCC* , minA/minV obtained higher $\overline{F}_1$.
maxE vs maxS & minA vs minV. $\overline{F}_1$ scores for maxE and maxS are very close, with only marginal differences on *MuC* and *MuCC* , and identical results on *Poke* . Although maxS typically achieves higher precision, maxE consistently yields the best recall. This suggests that although both criteria are similar in nature, prioritizing rule applications may lead to more cautious merging decisions while maintaining good coverage. Similar trends were observed between minA and minV, where minV achieved higher precision and better $\overline{F}_1$ scores on all datasets except *MuC* .
Set vs Cardinality. Accuracy metrics between S-optimal and C-optimal solutions differ only slightly, with the former generally achieving higher precision and the latter showing better recall. Although S optimization often produces many solutions, C-optimized solutions may be preferable when a single representative model is desired.

Efficiency For S-optimization, first-model solving time $\mathbf{t_s^1}$ is generally at the same level across criteria, with the largest difference being 1.1 times between minVS and maxES on *Poke* . Enumeration speed $\mathbf{t_s^n}$ is also similar, except on *Poke* , where minVS was 6 times faster than minAS. In contrast, C-optimization methods show greater variability. minAC was consistently the slowest for first-model computation, taking up to 2.8, 1.76, and 1.01 times longer than the fastest criteria (maxEC and maxSC) on *MuC* , *MuCC* , and *Poke* , resp.
Comparing the solving times of each S-optimal criterion with its C-optimal counterpart, we observed substantial increases in both $\mathbf{t_s^1}$ and $\mathbf{t_s^n}$—despite executing C-optimal computations on 36 threads. Without multi-threaded solving, both minAC and minV hit the 24-hour timeout, even with heuristic-driven solving enabled (see Appendix D). These results suggest that in resource-constrained environments, S-optimal solutions are more practical, as the marginal quality gains from C-optimal solutions often come at a significant computational cost.

5.3 Qualitative Studies

We present qualitative examples from the *MuCC* dataset, illustrating the efficacy and necessity of local semantics.
In the *MuCC* dataset, the schema includes:

- *Place*(*pid*,*name*,*type*,*address*,*area*,*coordinate*): stores information of building or outdoor area used for performing or producing music, where the *area* attribute is a reference to *id* of the *Area* table.
- *Release*(*rid*, *artist*, *name*, *barcode*, *language*): records information of release of music albums.

The ER program for *MuCC* contains: **i)** a hard rule hr_1: the *rid* of two releases are the same if they are from the same *artist* and in the same *language* and similar *name*s. **ii)** a soft rule sr_1: the *pid* of two places are possibly the same if they have the same *type* and similar *name*s. **iii)** two denial constraints d_1 and d_2: FDs requiring that *pid*/*rid* must associate to the same *coordinate*/*barcode*, respectively.

Blocked Merges The two *Place* tuples

- *Place*(t_1, p_1, *Kunsthalle* , *Kindikty*, a_1, *48.20 81.63*)
- *Place*(t_2, p_2, *Kunstha1le* , *Kindikty dist.*, a_2, *48.20-81.63*)

satisfy sr_1, but will be blocked by d_1 if the two *value constants 48.20 81.63* and *48.20-81.63* are not merged. Thus, the pair is absent in the stable models of ASPEN , but might be present in those of ASPEN$^+$.

Violation of DCs The two *Release* tuples

- *Release*(t_3, r_1, at_1, *chante les poètes*, *48.20 81.63*, *FR*)
- *Release*(t_4, r_2, at_2, *chanteLes poètes*, *48.20-81.63*, *Fr*)

satisfy the hard rule hr_1, requiring that $\texttt{eqo}(\texttt{r1}, \texttt{r2})$ be included in any stable model. However, without merging the value constants *48.20 81.63* and *48.20-81.63*, the program violates the DC d_2, resulting in no solution for *MuCC* . As a result, d_2 must be excluded from ASPEN 's ER program. In contrast, ASPEN$^+$ can safely incorporate d_2 by handling alternative representations of values.

Local Merges are Necessary To satisfy d_2, the barcodes in the two *Release* tuples must be merged. However, these same values also appear as *coordinates* in the *Place* table, where they refer to distinct locations—one in Germany, the other in Kazakhstan. As a result, the merge must be handled locally using ASPEN$^+$; a global merge would incorrectly unify distinct places and propagate errors to related tables such as *Area*, due to referential dependencies.

6 Conclusions

We have introduced ASPEN$^+$, a logic-based system for collective ER that supports both global and local merges, along with a suite of seven optimality criteria for preferred solutions. Our formalization and complexity analysis of these criteria offer new insights into optimal ER solution selection. Through extensive experiments, we demonstrate the practical benefits of local semantics and flexible optimization, achieving superior accuracy on complex, real-world datasets. ASPEN$^+$ thus marks a significant step toward principled ER in noisy and multi-relational environments.

Acknowledgements

The authors were partially supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014) and by MUR under the PNRR project FAIR (PE0000013). The authors thank the Potassco community for their support in resolving the issues encountered while using `clingo` .

References

Arasu, A.; Ré, C.; and Suciu, D. 2009. Large-scale deduplication with constraints using dedupalog. In Ioannidis, Y. E.; Lee, D. L.; and Ng, R. T., eds., *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, 952–963. IEEE Computer Society.

Bertossi, L. E.; Kolahi, S.; and Lakshmanan, L. V. S. 2013. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems* 52(3):441–482.

Bhattacharya, I., and Getoor, L. 2007. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data* 1(1):5.

Bienvenu, M.; Cima, G.; Gutiérrez-Basulto, V.; and Ibáñez-García, Y. 2023. Combining global and local merges in logic-based entity resolution. In Marquis, P.; Son, T. C.; and Kern-Isberner, G., eds., *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 742–746.

Bienvenu, M.; Cima, G.; and Gutiérrez-Basulto, V. 2022. LACE: A logical approach to collective entity resolution. In Libkin, L., and Barceló, P., eds., *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, 379–391. ACM.

Brewka, G.; Delgrande, J. P.; Romero, J.; and Schaub, T. 2023. A general framework for preferences in answer set programming. *Artif. Intell.* 325:104023.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Commun. ACM* 54(12):92–103.

Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2016. A declarative framework for linking entities. *ACM Trans. Database Syst.* 41(3):17:1–17:38.

Cabalar, P., and Muñiz, B. 2023. Explanation graphs for stable models of labelled logic programs. In *ICLP Workshops*, volume 3437 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Christophides, V.; Efthymiou, V.; Palpanas, T.; Papadakis, G.; and Stefanidis, K. 2021. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.* 53(6):127:1–127:42.

Deng, T.; Fan, W.; Lu, P.; Luo, X.; Zhu, X.; and An, W. 2022. Deep and collective entity resolution in parallel. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, 2060–2072. IEEE.

Fagin, R.; Kolaitis, P. G.; Lembo, D.; Popa, L.; and Scafoglieri, F. 2023. A framework for combining entity resolution and query answering in knowledge bases. In *KR*, 229–239.

Fan, W.; Jia, X.; Li, J.; and Ma, S. 2009. Reasoning about record matching rules. *Proceedings of the VLDB Endowment* 2(1):407–418.

Fan, W. 2008. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2008)*, 159–170. ACM.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gebser, M.; Kaufmann, B.; Romero, J.; Otero, R.; Schaub, T.; and Wanko, P. 2013. Domain-specific heuristics in answer set programming. In desJardins, M., and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, 350–356. AAAI Press.

Gebser, M.; Leone, N.; Maratea, M.; Perri, S.; Ricca, F.; and Schaub, T. 2018. Evaluation techniques and systems for answer set programming: a survey. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 5450–5456. ijcai.org.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Multi-threaded ASP solving with clasp. *Theory Pract. Log. Program.* 12(4-5):525–545.

Konda, P.; Das, S.; C., P. S. G.; Doan, A.; Ardalan, A.; Ballard, J. R.; Li, H.; Panahi, F.; Zhang, H.; Naughton, J. F.; Prasad, S.; Krishnan, G.; Deep, R.; and Raghavendra, V. 2016. Magellan: Toward building entity matching management systems. *Proc. VLDB Endow.* 9(12):1197–1208.

Köpcke, H.; Thor, A.; and Rahm, E. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.* 3(1):484–493.

Lifschitz, V. 2019. *Answer Set Programming*. Springer.

Papadakis, G.; Mandilaras, G. M.; Gagliardelli, L.; Simonini, G.; Thanos, E.; Giannakopoulos, G.; Bergamaschi, S.; Palpanas, T.; and Koubarakis, M. 2020. Three-dimensional entity resolution with jedai. *Inf. Syst.* 93:101565.

Rosa, E. D.; Giunchiglia, E.; and Maratea, M. 2010. Solving satisfiability problems with preferences. *Constraints An Int. J.* 15(4):485–515.

Singla, P., and Domingos, P. M. 2006. Entity resolution with markov logic. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, 572–582. IEEE Computer Society.

Xiang, Z.; Bienvenu, M.; Cima, G.; Gutiérrez-Basulto, V.; and Ibáñez-García, Y. 2024. ASPEN: asp-based system for collective entity resolution. In Marquis, P.; Ortiz, M.; and Pagnucco, M., eds., *Proceedings of the 21st International*

Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024.

Xiang, Z. 2025. ASPEN$^+$ Git Repository. https://github.com/zl-xiang/ASPEnP.