Exploring Desirable Configurations in Global Logistics with Heuristic Search in Answer Set Programming

Olcay Altay-Kern[1,2] , **Emmanuelle Dietz**[3] , **Isabelle Kuhlmann**[2] , **Matthias Thimm**[2]
[1]Airbus Operations, Hamburg, Germany
[2]Artificial Intelligence Group, University of Hagen, Hagen, Germany
[3]Airbus Central Research & Technology, Hamburg, Germany
{olcay.altay, emmanuelle.dietz}@airbus.com, {isabelle.kuhlmann, matthias.thimm}@fernuni-hagen.de

Abstract

In the design of global logistics problems, the solution spaces are typically extremely large. To demonstrate how these challenges can be addressed in Answer Set Programming (ASP), this work investigates a representative industrial use case of a global logistics problem in the aerospace problem domain. An exploration of specific areas of the search space is done by using heuristic-driven solving for the formulation of domain heuristics that guide the solver to potentially desirable configurations. A quantitative evaluation on the Key Performance Indicators and a qualitative evaluation on the variability of the models by means of a similarity analysis shows promising results.

1 Introduction

In the conceptual-design stage of a commercial aircraft and its manufacturing system, industrial architects must generate and compare multiple production architectures before any landmark investment decisions are taken by the senior management. Multiple scenarios need to be analyzed and optimized with regards to some pre-defined key performance indicators (KPIs), typically recurring production and transportation costs or non-recurring costs of investments (Arista et al. 2023; Zheng et al. 2024). Domain experts' knowledge is crucial in this process as industrial configuration problems tend to have huge search spaces which cannot be explored entirely (Falkner et al. 2018).

At Airbus (2025a), the transformation from document-based to model-based systems engineering and the use of modeling and simulation including multi-objective analysis and optimization is an ongoing process (Helle et al. 2008; Ferrogalini 2019). It aims to provide digital support for system architects and managers in making substantial and well-founded decisions which determine the allocation of huge investments in infrastructure and technology over the coming decades. In this paper, we focus on the development of the next generation Airbus single-aisle aircraft. Driven in part by historical developments and economic considerations, the manufacturing and assembly process of such a high-tech product includes workloads and technologies that require a variety of sites (production locations) equipped with various skills and capabilities distributed throughout the world (Airbus 2025b). Furthermore, in an environment defined by dynamic political and economic shifts (e. g., new trade agreements, increase in prices of essential commodities), the aerospace industry must enhance its resilience by maintaining numerous commitments to partners. These commitments involve dozens of major manufacturing sites worldwide, each responsible for producing or assembling specific components. The foreseen core (top-level) industrial and logistics system comprises over thirty major production and warehouse locations worldwide, along with more than twenty possible transportation means between them (Airbus 2025c; Airbus 2025d).

The objective that we address in this paper is to find solutions to the problem of identifying valid industrial and logistics setups with *Answer Set Programming* (ASP) for the next generation Airbus civil aircraft manufacturing system with (Pareto) optimal KPIs regarding the multi-objective functions. The problem contains 13 production locations, 17 warehouse locations, 34 parts and 30 transport means. ASP is one paradigm chosen in research to generate, analyze, and optimize industrial scenarios for possible implementation as a standard tool for trade-off analysis. A valid solution in our context is an answer set that ensures that all required manufacturing steps are passed in the right order and that transportation is performed between every manufacturing step, finally resulting in the representation of a multi-echelon production and a supply chain network, which describes a manufacturing system that is arranged in stages or multiple layers with materials and products flowing downstream (Section 4.1). The structure and behavior of the industrial system are significantly influenced by general strategies or assumptions about the supply chain and manufacturing setup, referred to as *industrial scenarios*. The topology of the industrial network is changed by these scenarios, which alters its properties and behavior. This leads to additional design constraints and a more complex system, making the associated problems harder to formulate. Sourcing strategies and warehouses are possible constraints (or scenarios) to be considered. The sourcing strategy determines whether a production step must be performed by at least one site or by multiple sites to ensure redundancy, thus potentially increasing system robustness. This strategy can be applied selectively, depending on the production stage. Warehouses similarly contribute to the network's robustness. They can influence the reachability of production sites by enabling changing the transport resources. Moreover, having high variability

among answer sets is required for two reasons: first, to increase confidence that a large amount of the search space is explored and no better local optimum is missed, and second, to provide a variety of solutions that an industrial architect can actively explore to learn from unknown solutions and gain additional knowledge which can support an iterative and incremental product development. This requires an interactive way-of-working with relatively short optimal answer set solving run-times.

The work in this paper takes the logic programming approach to global logistics in a co-design environment presented in (Dietz et al. 2023) as a starting point. The authors detail a co-design strategy using ASP to optimize the global logistics for aircraft construction. To achieve this, the relevant requirements are extracted from a knowledge graph and translated into logic programs. The computed configurations (answer sets) are visualized for easier evaluation. Here, we will analyze the global logistics problem in a broader context and define additional optimization criteria. Additionally, specific areas of the search space will be actively explored by defining *domain heuristics*. This ensures that larger fractions of the search space are discovered and possibly other interesting (local) optima can be found.

The paper is structured as follows. After related work in Section 2, preliminaries on ASP and model similarities are introduced in Section 3. Next, the main contributions of this paper will be presented:

- A general description of the global logistics problem including a formalization in ASP (Section 4),
- the formulation of domain knowledge through facts, integrity constraints and heuristics that guide the solver to potentially desirable configurations (Section 5), and
- an evaluation with respect to key performance indicators and model similarities (Section 6).

Section 7 concludes with a summary and future directions.

2 Related Work

Several approaches of heuristics in ASP within an industrial setting have been proposed. In (Rajaratnam et al. 2023), the authors describe a warehouse delivery problem which consists of a set of robots that need to carry out delivery jobs. The authors develop two encodings, the step-based and path-based encoding. Although the step-based encoding fails to scale to the required level within the industrial setting, the path-based encoding seems efficient enough. Domain heuristics are crucial to reach the required performance. The authors emphasize that significant improvement in the solution quality could be gained relatively easily by using application-specific constraints. A new language to facilitate the non-monotonic specification of heuristics on partial assignments in ASP is presented in (Comploi-Taupe et al. 2023). The authors demonstrate that their approach performs well in two practical domains and offers greater flexibility to define heuristics beyond traditional planning.

Faceted navigation (Alrabbaa, Rudolph, and Schweizer 2018) allows to navigate through the space of answer-sets in a systematic and interactive way. The weighted navigation approach (Fichte, Gaggl, and Rusovac 2022) additionally takes into account the change of facets (with respect to the solution space) to configure the pace of navigation. The authors in (Böhl and Gaggl 2022; Böhl, Ellmauthaler, and Gaggl 2024) propose a multi-shot ASP approach (Gebser et al. 2019b) to identify diverse answer sets, by improving the collection of answer sets. The diversity measure can be specified by the user and the result provides a nice overview on the overall solution space. Similarly (Böhl, Gaggl, and Rusovac 2023) suggest faceted answer set navigation for obtaining 'representative collections' of answer sets. A visual navigation tool for the exploration of solution spaces is proposed in (Dachselt et al. 2022).

The global logistics problem is about the search for interesting product configurations. It would be particularly interesting to see if these methods can be scaled up to our work.

3 Preliminaries

In Section 3.1 we introduce the general notation and terminology and give a brief introduction to ASP (Niemelä 1999; Gebser et al. 2012) and the stable model semantics (Gelfond and Lifschitz 1991). The interested reader is referred to (Brewka, Eiter, and Truszczyński 2011; Janhunen and Niemelä 2016) and the Potassco guide (Gebser et al. 2019a). Section 3.2 provides a method for computing similarities between models applying the Jaccard coefficient (Salton and McGill 1983).

3.1 Answer Set Programming

We consider the countable set of *terms* $\mathcal{T} = \{t_1, \ldots, t_n\}$ that consists only of constants (starting with a lower case letter) and variables (starting with an upper case letter). An *atom* is an expression $p(t_1, \ldots t_m)$ where p is a predicate, $m \geq 0$ and $t_1, \ldots t_m \subseteq \mathcal{T}$. Further, $\mathcal{A}$ is a fixed, finite and non-empty *set of atoms*. A *ground atom* is an atom with only constants. If A is an atom, then $L = A$ is a positive literal and $L = not\ A$ is a negative literal, with *not* being *default negation* (Reiter 1978; Reiter 1980). A (normal) *rule* r is of the form

$$A \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n.$$

where A and A_i, with $1 \leq i \leq n$, are atoms. The *head* of r is denoted as $H(r) = A$. The conjunction to the right of the implication symbol is called *body* of r and is denoted as $B(r) = \{A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n\}$. The set of all positive literals in $B(r)$ is denoted as $B^+(r) = \{A_1, \ldots, A_m\}$ and the set of all negative literals in $B(r)$ is denoted as $B^-(r) = \{not\ A_{m+1}, \ldots, not\ A_n\}$. A rule is *safe* if each variable in r occurs in $B^+(r)$. A rule is *ground* if no variable occurs in r. A *fact* is a ground rule with empty body, i. e. $n = 0$.

A *logic program* $\mathcal{P}$ is a (finite) set of rules. For any program $\mathcal{P}$, let $C_\mathcal{P}$ be the set of all constants appearing in $\mathcal{P}$. In the following, we assume for all $\mathcal{P}$ it holds that $C_\mathcal{P} \neq \emptyset$. The ground program g$\mathcal{P}$ is the set of rules $r\sigma$ obtained by applying to each rule $r \in \mathcal{P}$, all possible substitutions σ from the variables in r to the elements of $C_\mathcal{P}$.

The set of all atoms in g$\mathcal{P}$ is denoted by $\mathcal{A}_\mathcal{P}$. An *interpretation* I of a program $\mathcal{P}$ is a mapping of $\mathcal{A}_\mathcal{P}$ to the set of

truth values $\{\top, \bot\}$, where $\top$ means *true* and $\bot$ means *false*. Given a ground rule r, $I(r) = \top$ denotes that the interpretation I maps r to $\top$ according to the corresponding logic. An interpretation I is a *model* of $\mathcal{P}$ where for each rule r occurring in $g\mathcal{P}$ it holds that $I(r) = \top$. Consider $\mathcal{P}_{\text{ex}}$ a program with three facts and one rule:

$$\text{productionLoc(hh).} \quad \text{productionLoc(tls).} \quad \text{part(tail).}$$
$$\text{produceableAt(P, L)} \leftarrow \text{part(P)}, \text{productionLoc(L).}$$

The facts state that Hamburg (hh) and Toulouse (tls) are production locations and tail is a (production) part. The rule states that every part can be produced at any production location.

The ground program $g\mathcal{P}_{\text{ex}}$ is as follows:

$$\text{productionLoc(hh).} \quad \text{productionLoc(tls).} \quad \text{part(tail).}$$
$$\text{produceableAt(tail, hh)} \leftarrow \text{part(tail)}, \text{productionLoc(hh).}$$
$$\text{produceableAt(tail, tls)} \leftarrow \text{part(tail)}, \text{productionLoc(tls).}$$

An interpretation $I \subseteq \mathcal{A}_{\mathcal{P}}$ *satisfies* a ground rule r iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^- \cap I = \emptyset$. Interpretation I satisfies a ground program $\mathcal{P}$, if each $r \in \mathcal{P}$ is satisfied by I. A non-ground rule r (resp. a program $\mathcal{P}$) is satisfied by an interpretation I if I satisfies all ground instances of r (resp. of $g\mathcal{P}$). I is an *answer set* (also called *stable model*) of $\mathcal{P}$ iff I is the subset-minimal set satisfying the *Gelfond-Lifschitz reduct*: $\mathcal{P}^I = \{H(r) \leftarrow B^+(r) \mid I \cap B^-(r) = \emptyset, r \in g\mathcal{P}\}$. The only answer set of $g\mathcal{P}_{\text{ex}}$ is where produceableAt(tail, hh), produceableAt(tail, tls), productionLoc(hh), productionLoc(tls) and part(tail) are mapped to *true*.

The following two statements are commonly used in ASP.

$$\leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n.$$
$$l\{A : A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n\}u.$$

They are called *integrity constraint* and *cardinality constraint*, respectively. Similar as above, A and A_i with $1 \leq i \leq n$ are atoms. Intuitively, an integrity constraint represents an undesirable situation, i.e. $A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$ should be evaluated to *false*. A cardinality constraint is of the form $l\{c_1; \ldots; c_m\}u$, with $c_1, \ldots, c_m$ being conditional literals, l constituting an optional lower bound, and u an optional upper bound. A *conditional literal* is of the form $L_0 : L_1, \ldots, L_n$, with $L_0, \ldots, L_n$ being literals, and regulates the instantiation of L_0 by means of $L_1, \ldots, L_n$. In other words, we can view a conditional literal as the set $\{L_0 \mid L_1, \ldots, L_n\}$. Further, cardinality constraints can be used in both rule bodies and heads. A rule with a cardinality constraint as the head is referred to as an *(extended) choice rule*. Formally, a choice rule has the form

$$l\{A_1, \ldots, A_m\}u \leftarrow A_{m+1}, \ldots, A_n, \mathit{not}\ A_{n+1}, \ldots, \mathit{not}\ A_o.$$

with $0 \leq m \leq n \leq o$, $A_1, \ldots, A_o$ being atoms, $0 \leq l \leq u$, and l, u being (optional) lower and upper bounds, respectively. An example of an integrity constraint and a choice rule are **(icLocation)** and **(choicePath)** in Table 1, respectively.

In contrast to integrity constraints, *weak constraints* (Buccafurri, Leone, and Rullo 2000) do not necessarily need to hold in the models of the given program:

$$:\sim A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n.[w@p, t_1, \ldots, t_n]$$

with weight w and (optional) priority p, where $w, p \in \mathbb{N}$, and terms $t_1 \ldots t_m$ that occur in $A_1, \ldots A_m$. The optimization statement

$$\#\textbf{minimize}\{w_1@p_1, \mathbf{t}_1 : \mathbf{L}_1, \ldots, w_1@p_n, \mathbf{t}_n : \mathbf{L}_n\}$$

is a short form of the n weak constraints

$$:\sim \mathbf{L}_1. \quad [w_1@p_1, \mathbf{t}_1] \qquad \ldots \qquad :\sim \mathbf{L}_n. \quad [w_n@p_n, \mathbf{t}_n].$$

with $\mathbf{t}_i = t_1, \ldots, t_{m_i}$ being terms and $\mathbf{L}_i = L_1, \ldots, L_{o_i}$ being literals, $1 \leq i \leq n$, and $m_i, o_i \in \mathbb{N}$. The corresponding **#maximize** statement is analogously defined with inverted weights. Examples of optimization statements are **(minPC)**, **(minRC)**, **(minCO2)** in Table 1.

Moreover, we make use of *aggregates* in order to reason about sums over sets of literals. See **(icWorkshare)** in Table 1 for an example.

Heuristic-driven solving (Gebser et al. 2013) allows one to include domain knowledge that guides the search for (possibly) promising models by extending the default heuristic which decides how to assign the truth value of a ground atom when solving. A *domain-specific heuristic* is represented as follows:

$$\#\textbf{heuristic}\ A_0 : A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n.[w@p, m]$$

with weight w, (optional) priority p where $w, p \in \mathbb{N}$, and modifier m, which can represent different types of heuristic information. Six different types of modifiers m are available in Clingo. Here, we only consider m as having a type of either *true* or *false*, which influences the solver's preference when making the decision on the truth value of the atom(s) in consideration.

3.2 Model Similarities

To accommodate the increased variability of solutions, we need a robust way to quantify similarities between answer sets, both within a single optimization run and across runs under different configurations. Each solution comprises multiple transportation paths, determined by the choice of production sites for each part and the transportation means between sites (see Section 4). A possible metric for comparing such sets of paths is the *Jaccard coefficient* (Salton and McGill 1983)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

where J is the Jaccard similarity between two sets A and B.

We will apply this measure to the allocation of parts to production sites and the choice of transportation means between sites. First, the set with respect to the allocation of a part $p \in P$ to sites $s \in S$ for a model is defined as

$$S_{nodes}(p) = \{S_{nodes_1}(p), \ldots, S_{nodes_n}(p)\}$$

with $S_{nodes_i}(p) = \{(p, s_i) \mid p \in P, 1 \leq i \leq n\}$ with n being the total number of sites $s \in S$ producing part

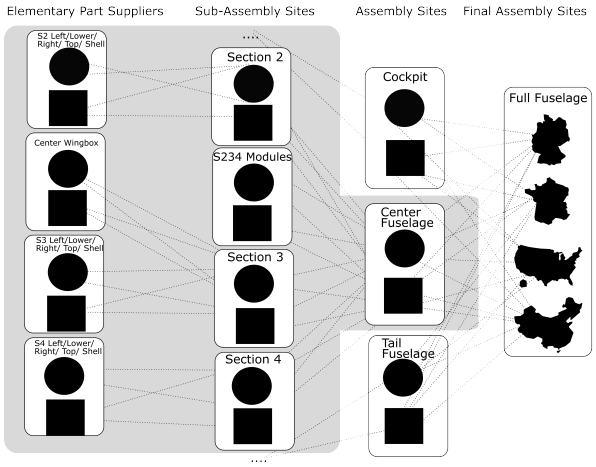

Figure 1: Illustration of a multi-echeleon network applied to the production plan of the A320 aircraft where ● and ■ represent different sites for the produced part. The sub-assembly sites and elementary part suppliers are shown only for the center fuselage (highlighted in gray).

$p \in P$. The corresponding set of all such sets is denoted as $S_{nodes} = \{S_{nodes}(p) \mid p \in P\}$. Likewise, the set of all sets with respect to all choices of transportation means between sites for a model (or answer set) is defined as

$$S_{edges} = \{S_{edges_1}, \ldots, S_{edges_m}\}$$

with m being the number of all performed transportation activities. Accordingly, we compute the Jaccard similarity between two answer sets m_1 and m_2 following Equation 1:

$$J_{production} = \frac{|S_{nodes_{m_1}} \cap S_{nodes_{m_2}}|}{|S_{nodes_{m_1}} \cup S_{nodes_{m_2}}|}$$

$$J_{transport} = \frac{|S_{edges_{m_1}} \cap S_{edges_{m_2}}|}{|S_{edges_{m_1}} \cup S_{edges_{m_2}}|}$$

This measure was identified as the best fit for the global logistics problem to reflect what are *similar* solutions when assigning parts to production sites (production similarity) and transportation means to transport routes (transport similarity), which are the main decisions to be taken. Therefore, we use this measure to evaluate the similarities among models computed with different heuristics in Section 6.

4 The Global Logistics Problem

This section details the specific global logistics problem subject to this study. Section 4.1 shows its correspondence to two classical problems from the literature and, Section 4.2 provides a formalization in ASP.

4.1 Two Problems in One

The global logistics problem consists of two subproblems: First, we need to guarantee that each site produces at least one part (or the required work share by site or country), and second we need to decide on the transportation means among the production sites. In both cases, we are interested in 'good' (i. e., nearly optimal) solutions. For better visualization of the problems, we consider the representation as a multi-echelon production network, which is a production network that is organized in stages that need to be passed downstream. This will be explained in detail later in this section.

The first subproblem, where parts are matched with production locations, can be regarded as a *set multi-cover problem*. The production locations responsible for manufacturing specific parts correspond to the nodes of the multi-echelon industrial and supply chain network. The second subproblem is about selecting transportation means among the production sites, which completes the network with its edges. Optimization inside this network, e. g. minimizing transportation costs, corresponds to the *Shortest path in a Directed Acyclic Graph Problem*. The details are described below.

Optimal Allocation of Part Production Across Sites A set of manufacturing sites across the world is involved in the full manufacturing chain, all with their individual capacities to produce specific components. As sites can produce multiple parts and parts need to be produced by at least two sites, this leads to a many-to-many assignment problem. In this case, we deal with a special case of a *Set Multicover Problem (SMC)*. A *Set Multicover Cover Problem* is to cover all elements of a universe N containing n elements at least a pre-defined number of times i with a minimum number of sets. It is a generalization of the *Set Cover Problem* with the main difference that elements of the universe may be covered multiple times (Hua et al. 2010).

To formalize the above problem, let there be a set $P = \{p_1, p_2, ..., p_n\}$ of parts to be produced and a set $S = \{s_1, s_2, ..., s_m\}$ of (production) sites. Each site $s \in S$ can produce a subset $P_s \subseteq P$ of all parts. Depending on the sourcing strategy, each part $p \in P$ must be produced by i_p sites (e. g. $i = 1$ for single sourcing, $i = 2$ for double sourcing, etc.). Each part p has a fixed workload w_p independent from the site where it is produced. And finally each site $s \in S$ has a dedicated cost factor c_s reflecting the specific manufacturing cost level of the location. Let x_{ps} be the decision variable with

$$x_{ps} = \begin{cases} 1 & \text{if part } p \text{ is produced at site } s \\ 0 & \text{otherwise} \end{cases}$$

The objective function is

$$\min \sum_{s \in S} \sum_{p \in P} c_s \cdot w_p \cdot x_{ps}$$

accordingly and we have the following constraints:

- *Workshare*: each site must produce at least one part

$$\sum_{p \in P_s} x_{ps} \geq 1 \; \forall s \in S$$

- *Sourcing strategy*: each part p must be produced by at least i_p sites

$$\sum_{s \in S : p \in P_s} x_{ps} \geq i_p \; \forall p \in P$$

- *Eligibility*: parts can only be assigned to sites when they are able to produce the part $x_{ps} = 0$ if $p \notin P_s$.

Optimal Routing Across Production Sites The second problem is finding the shortest path in a logistics network which can be regarded as a *Multi-Echelon Supply Chain and Production System*. It represents an industrial setup that is structured in manufacturing stages. Every stage must be passed downstream and for all stages there might be multiple sites to fulfill the task. It can typically be represented as a *Directed Acyclic Graph* (DAG) (Thulasiraman and Swamy 2011). The production plan of the A320 aircraft (see, e. g., (Jirkovský et al. 2025)) is a multi-echelon network with four production stages: elementary part, sub-assembly, assembly, and final assembly, as shown in Figure 1. All need to be passed downstream in this order to produce a product. Common challenges in industrial applications are the optimization of such a network with regard to some KPIs, e. g. production or transportation costs, or analyzing the robustness of the network against disruptions.

Let G be a DAG with $G = (V, E)$ with $u \in V$ and V is partitioned into l layers $V = V_1 \cup V_2 \cup ... \cup V_l$ where each V_i (with $i \in \{1, \ldots, l\}$) is the set of nodes representing the possible production sites at echelon i. Edges $E \subseteq V \times V$ are directed and can only go from one level i to the next level $i + 1$. Each edge $(u_i, v_{i+1}) \in E$ has a cost $c_{tm}(u_i, v_{i+1})$ where tm is any transportation means that is available between the nodes u_i and v_{i+1}.

As there are multiple possibilities to select production sites per echelon, binary decision variables must be introduced for any edge:

$$x_{uv} = \begin{cases} 1 & \text{if } u \text{ and } v \text{ and } edge(u, v) \text{ is selected in the path} \\ 0 & \text{otherwise} \end{cases}$$

The optimization problem with regards to any cost c_{tm} of a transportation means tm at the edge (u_i, v_{i+1}) can then be expressed by:

$$\min \sum_{(u_i, v_{i+1}) \in E} c_{tm}(u_i, v_{i+1})$$

Problem formulations that are expressed as shortest path problems in a DAG with vertices V and edges E can be solved in linear time with $\Theta(V + E)$, for instance with a topological sort method (Cormen et al. 2009).

Regarding the KPIs, the production costs are determined by the assignment of parts to production sites and therefore subject to the *Set Multicover Problem*. Transportation costs and CO_2 emissions on the other hand are subject to the routing problem as they can be represented and computed from the cost of the edges.

In Figure 1, ● and ■ in each rounded rectangle represents a decision to allocate a produced part to a site, while each dotted line between sites represents a route across production sites. For simplicity, we omit the labels on the lines with the respective transport means.

4.2 Formalization in ASP

We will first outline the original approach in (Dietz et al. 2023) and then present the novel work in this paper that improves upon it by including additional domain knowledge.

Original Program The rules in Table 1 show the original program excluding the ~ 30.000 facts about canTransport/8, productionLoc/1, warehouseLoc/1 and produceableAt/2, productionPlan/2. More details on the facts can be found in (Dietz et al. 2023). As discussed in Section 4.1, the global logistics problem consists of two problems: The choice rules (**choiceLvL1**), (**choiceLvL2**), (**choiceLvL3**), (**choiceLvL4**), in addition to the rules described by **(icLocation**) in Table 1 specify the allocation of part production between sites and its sourcing strategy, while (**directRule**), (**ruleVia1**), (**ruleVia2**), (**choicePathFinal**), and (**choicePath**) specify the routing between production sites. Optimization in routing between production sites is formalized in (**minRC**) and (**minCO2**).

Additional Domain Knowledge The search space contains too many 'irrelevant' configurations (models) with the original program and the necessary restrictions (or integrity constraints) still need to be fully understood and defined. Industrial architects often notice that the presented configurations (or answer sets) are not adequate in the sense that they do not account for *seemingly obvious information*. Furthermore, adding such information accelerates the search process in ASP significantly: As an example consider (**ruleVia1**) and (**ruleVia2**) in Table 1: Specifying in the rule that via1 and via2 are warehouses, reduces grounding from minutes to seconds (Dietz et al. 2023).

First, the original program did not include the KPI on optimal part production location allocation. This is now represented by the optimization statement (**minPC**) in Table 1, where costLoc/2 is a fact specifying the production costs at a given location. In order to avoid ambiguity in the implementation of the objective functions, explicit priority levels (@...) have been defined for all minimization statements.

Second, even though the models (or configurations) in the previous program were valid, they do not seem to be implementable in the industrial system. For instance, variability regarding the sourcing strategies at different levels is necessary, which is expressed by the rules (**choiceLvL1**), (**choiceLvL2**), (**choiceLvL3**), and (**choiceLvL4**) in Table 1.

Solvers such as *clasp* (Gebser et al. 2007), which utilize *Conflict-Driven Nogood Learning* (CDNL) techniques, rely on conflicts to prune the search space efficiently. Without sufficient conflicts, the search space remains largely unexplored, preventing the optimizer from converging to an optimum (Gebser et al. 2012). We therefore actively searched for additional domain knowledge that would significantly reduce the search space. For instance, models in which one of the main production sites (or countries) has a very low work share, are strategically not realistic in the established environment. Such requirements might be suboptimal from a purely quantitative perspective with respect to the KPIs, however they are essential. The rules specified by **(icWorkshare**) in Table 1 ensure that both Toulouse and Hamburg (and the respective countries in which they are located in) will have a work share above a certain threshold. This work share is computed with help of the fact valAdded/2 specifying the added value of each product.

Program excluding facts	Description	Label
via1(Part, From, (Via1, To), (TM1, TM2), D, LT, CO2, TC) ← canTransport(From, Via1, Part, TM1, D1, LT1, CO21, TC1), From! = To, productionLoc(From), productionLoc(To), warehouseLoc(Via1), canTransport(Via1, To, Part, TM2, D2, LT2, CO22, TC2), D = D1 + D2, LT = LT1 + LT2, CO2 = CO21 + CO22, TC = TC1 + TC2	transport part via one warehouse or	(**ruleVia1**)
via2(Part, From, ((Via1, Via2), To), (TM1, TM2, TM3), D, LT, CO2, TC) ← canTransport(From, Via1, Part, TM1, D1), From! = To, productionLoc(From), warehouseLoc(Via1), canTransport(Via1, Via2, Part, TM2, D2), Via1! = Via2, productionLoc(To), warehouseLoc(Via2), canTransport(Via2, To, Part, TM3, D3), D = D1 + D2 + D3, LT = LT1 + LT2 + LT3, CO2 = CO21 + CO22 + CO23, TC = TC1 + TC2 + TC3	via two warehouses	(**ruleVia2**)
← productionLoc(P), *not* produced(P)	at least one part per site	(**icLocation**)
produced(P) ← producedAt(_, P)	true if P produces a part	
1{path(Part, From, To, TM, D) : canTransport(From, To, Part, TM, D, LT, CO2, TC); path(From, Part, (Via1, To), (TM1, TM2), D, LT, CO2, TC) : via1(Part, From, (Via1, To), (TM1, TM2), D, LT, CO2, TC), path(Part, From, ((Via1, Via2), To), (TM1, TM2, TM3), D, LT, CO2, TC),) : via2(Part, From, ((Via1, Via2), To), (TM1, TM2, TM3), D, LT, CO2, TC),)}1 ← producedAt(Part, From), productionPlan(Super, Part), producedAt(Super, To)	direct path, or path via one site, or path via two sites	(**choicePath**)
e{producedAt(Part, P) : produceableAt(Part, P)}e ← elementary(Part)	depending on its part	(**choiceLvL1**)
s{producedAt(Part, P) : produceableAt(Part, P)}s ← subassembly(Part)	each part needs to be	(**choiceLvL2**)
a{producedAt(Part, P) : produceableAt(Part, P)}a ← assembly(Part)	produced at e, s, a or f	(**choiceLvL3**)
f{producedAt(Part, P) : produceableAt(Part, P)}f ← finalProduct(Part)	sites, with $e, s, a, f \in \mathbb{N}$	(**choiceLvL4**)
← #**sum**{Value, Part : producedAt(Part, P), valAdded(Part, Value), locatedIn(P, france)} ≤ n	work share for France,	(**icWorkshare**)
← #**sum**{Value, Part : producedAt(Part, P), valAdded(Part, Value), locatedIn(P, germany)} ≤ n	Germany,	
← #**sum**{Value, Part : producedAt(Part, hh), valAdded(Part, Value)} ≤ m	Hamburg and	
← #**sum**{Value, Part : producedAt(Part, tls), valAdded(Part, Value)} ≤ m	Toulouse with $m, n \in \mathbb{N}$	
#**minimize**{(Value ∗ Cost)/e@6, Part, P : producedAt(Part, P), valAdded(Part, Value), costLoc(P, Cost)}	minimize production costs	(**minPC**)
#**minimize**{TC@5, Part, From, To, TM : path(Part, From, To, TM, D, LT, CO2, TC)}	minimize transportation costs	(**minRC**)
#**minimize**{CO2@4, Part, From, To, TM : path(Part, From, To, TM, D, LT, CO2, TC)}	minimize CO_2 emissions	(**minCO2**)

Table 1: Relevant rules, optimization statements and additional domain knowledge excluding the facts.

5 Heuristics

A first analysis of the models generated by the program presented in Section 4.2 shows that the variety among these models is quite low. Industrial architects are not primarily interested in small variations among models (such as a few changes in transport resources) but rather in models that are much different to each other while still fulfilling the specified requirements. For this purpose we will investigate how such a variety can be expressed through heuristic driven solving (Gebser et al. 2013) and preferences. In the following, we present a baseline program (Section 5.1), and subsequently describe multiple extensions of this program by means of different heuristics (Sections 5.2–5.5).

5.1 Baseline Program

The baseline program consists of all the rules in Table 1 including the facts about the global logistics. The optimization statements are specified with the following prioritization: 1. production costs (**minPC**), 2. transportation costs (**minRC**) and 3. CO_2 emissions (**minCO2**).

5.2 Preferred Production Sites (preferred sites)

The optimization of production costs can be facilitated by guiding the search through preferred production sites, i.e., sites where production costs are lowest:

#heuristic producedAt(Part, p) : part(Part).[w, *true*] with $w \in \mathbb{N}$.

where p is a constant representing a production site.

5.3 Preferred Transport Means (preferred TMs)

Optimization of transportation costs can be facilitated by guiding the search through preferred transport means, that is, where transportation costs are the lowest. We define[1]

#heuristic path(..., TM, ...) : truck(TM).[w, *true*] with $w \in \mathbb{N}$.

Transport means that are preferred are trucks with 'standard sized means', which is expressed by truck(trLongDist; trShortDist; trGeneralCargo).

5.4 Hubs Preferred (hubs preferred)

It seems reasonable that parts of the same level are transported to a possibly close site or to the same site for further assembly (cf. Fig. 1), the so-called hubs. We specify that sub-assembly and assembly should be produced at the same location:

#heuristic producedAt(Part, P) : productionPlan(Super, Part), producedAt(Super, P), subassembly(Part).[w, *true*]

with $w \in \mathbb{N}$. If the level of the hub should not be specified, then subassembly(Part) can be omitted.

5.5 Avoid Back-Forth Transport (min back-forth)

Configurations where a part is produced in location A (previous), then sub-assembled in location B (intermediate), and then again assembled in location A (next) should be avoided.

[1] The '...'s abbreviate path in (**choicePath**) in Table 1.

	% of models cost to last (**baseline**)		
Heuristic	Production	Transport	CO_2
last (**baseline**)	100	100	100
first (**baseline**)	102	174	128
(**preferred sites**)	96	66	100
(**preferred TMs**)	98	110	110
(**hubs preferred**)	97	104	102
(**min back-forth**)	110	58	80
(all)	96	63	112

Table 2: The KPI percentages of each heuristic in relation to the last (**baseline**) model. The best results are highlighted in gray.

A straightforward translation is to specify that the intermediate location should not be different to the previous and next one, when they are the same:

#heuristic producedAt(Super, P2) : producedAt(Part, P1), productionPlan(SuperS, Super), productionPlan(Super, Part), producedAt(SuperS, P1), P2! = P1.[w, *false*] with $w \in \mathbb{N}$.

This can be encoded through the minimize statement

#minimize{1, Part, SuperS : backForth(part, SuperS)}.

where

backForth(Part, SuperS) ← productionPlan(SuperS, Super), productionPlan(Super, Part), producedAt(Part, P1), producedAt(Super, P2), producedAt(SuperS, P1), P2! = P1.

Yet the preference can be reformulated by stating that if previous and next locations are the same, then the intermediate location should also be the same. Thus, we use

#heuristic producedAt(Super, P) : producedAt(Part, P), productionPlan(SuperS, Super), productionPlan(Super, Part), producedAt(SuperS, P).[w, *true*] with $w \in \mathbb{N}$.

An initial evaluation shows that the last encoding seems performs the best and thus will be chosen for the evaluation.

6 Evaluation

The heuristics presented in Section 5 will be evaluated with respect to the KPIs of the last model[2] for each heuristic and the variations between them. Experiments were carried out on an AMD EPYC 7443P 24-Core Processor, 64GB RAM type DDR4-3200. A timeout was set to 60 minutes. Note that in Figures 2 to 4, only every 100th model is plotted.

6.1 Dataset

The experiments were performed on a real dataset for a future aircraft program, with 13 production locations, 17 warehouse locations, 34 parts and 30 transport means. Overall there where about ~ 30.000 (logic program) facts.

[2] By 'last model' we refer to the last model that could be computed before the timeout.

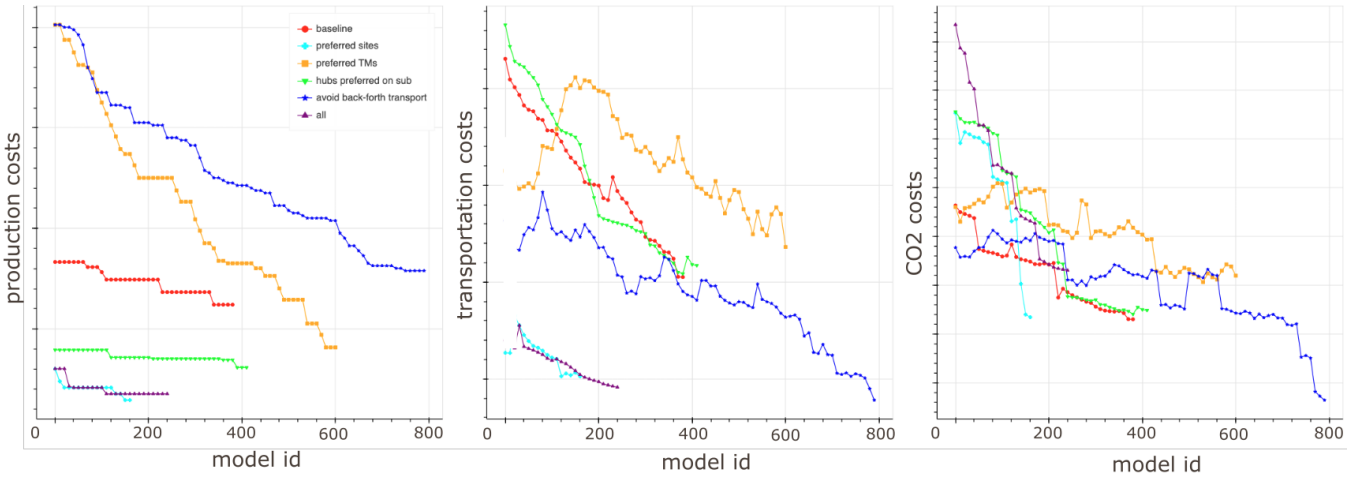

Figure 2: The development of production costs (left), recurring costs (middle) and CO_2 emissions (right) with respect to the heuristics.

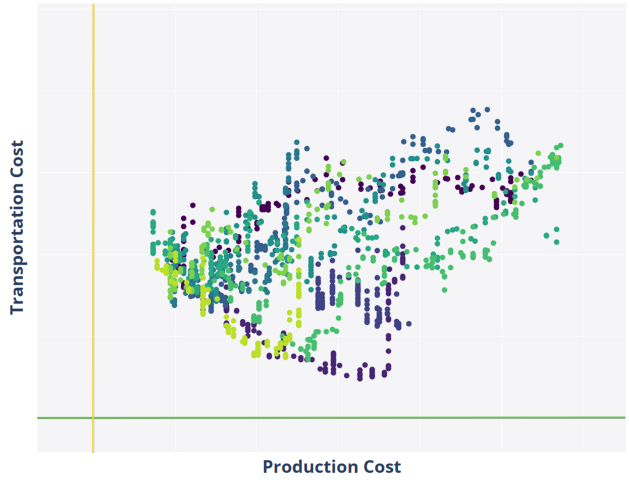

Figure 3: Evolution of production costs and transportation costs.

6.2 Optimization Results

In all cases, grounding took about half a minute and the first model was found after a few seconds. Table 2 shows the percentages of the KPIs (production costs, transportation costs, CO_2 costs) for the last model of each heuristic program with respect to the last model of the baseline program (**baseline**). The second row ('first (**baseline**)') shows the percentage of its first model's KPIs with respect to its last model's KPIs. It shows that the improvement in transportation costs and CO_2 costs are the highest: the last model's transportation costs and CO_2 costs are 57% and 78% of the first model's transportation costs and CO_2 costs, respectively.

Figure 2 shows how the models' KPIs evolved over time. The x-axis refers to the model id (every 100th model) and the y-axis refers to the respective KPI. Interestingly, (**preferred sites**) has the lowest production costs, whereas (**min back-forth**) achieves by far the lowest recurring and CO_2 costs. With the combined use of all heuristics, good solutions are generated from the start for production and transportation costs. Note that ASP forces us to prioritize the KPIs, which restricts the variability regarding the KPIs' lower and upper bound: minimizing production costs has the highest priority whereas minimizing CO_2 costs has lowest priority.

The goal of industrial architects is to find Pareto-optimal models in the industrial system. A *Pareto optimization* aims at finding a best solution between two potentially conflicting objective functions (Martins and Ning 2021).

As Table 2 shows, there is a trade-off between production and transportation costs. This observation is visualized in Figure 3, indicating a Pareto front with respect to production and transportation costs. The figure also shows models of combined heuristics to make the Pareto front better visible. The vertical and horizontal lines show an approximation of lower boundaries for the production costs (yellow line) and transportation costs (green line) defined as follows: For the production cost lower boundary we assumed that all parts are produced at the site with the lowest hourly rate. For the transportation lower boundary all sites were connected with the shortest path regarding the distance and transported by the transportation mean with the lowest cost per distance. It emphasizes that the results are close or already hitting the Pareto front.

6.3 Similarity Evolution

The similarity measure, defined in Section 3.2, is applied in three ways. First, we compare the models of the first and last model of each heuristic run to determine the internal variability within a single run. Second, we aim to see the variability against a reference model to see how much variability can be achieved overall among different runs with different heuristic settings. Finally, the heuristics are compared to each other to check whether they produce diverse answer sets.

Figure 4 shows how the similarity values evolve from the first model to the last model. The x-axis is the index of the

Heuristic	(**preferred sites**)		(**preferred TMs**)		(**hubs preferred**)		(**min back-forth**)		(**all**)	
	p	t	p	t	p	t	p	t	p	t
(**baseline**)	0.22	0.15	0.5	0.23	1	1	0.43	0.21	0.4	0.17
(**preferred sites**)	1	1	0.31	0.16	0.22	0.15	0.22	0.16	0.2	0.15
(**preferred TMs**)			1	1	0.5	0.23	0.42	0.21	0.34	0.17
(**hubs preferred**)					1	1	0.43	0.21	0.4	0.17
(**min back-forth**)							1	1	0.33	0.18
(**all**)									1	1

Table 3: Similarity comparison for all heuristics, with respect to the production sites (p) and transportation means (t).

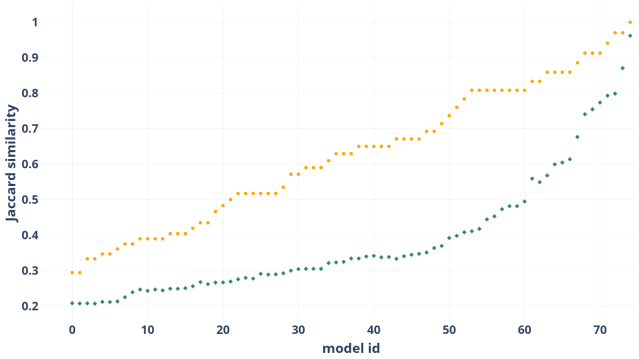

Figure 4: Evolution of the similarity values for production (yellow) and transport (green) within the baseline run.

model and the y-axis is the similarity value, which is split for the production (orange) and transportation (green) similarity. The figure shows the according similarity value of each model compared with the last (best) model.

The first model of the baseline run has similarities with the last model between 0.21 for transportation and 0.29 for production similarity. Similarity values steadily go up until finally reaching 1.0 for both values. This behavior is observed for all heuristic runs. The span of variability within a run is similar for all heuristics, except the 'preferred sites' heuristic, which has a significantly higher similarity for the production starting at 0.91 already from the first model.

Figure 4 indicates that the similarity values for production grow linearly (yellow) while for transportation it almost grows exponentially (green). This is an effect resulting from the prioritization of the optimization statements. With the baseline setting, first the production costs are optimized, then the transportation costs. Therefore, once the production value has improved (one iteration), several iterations of improving the transportation cost follow. Secondly, transportation similarity is production similarity-dependent, meaning that a transportation link can only be similar when both nodes (production sites) are already similar.

The matrix in Table 3 finally shows that even among the heuristics, the variability is high and the different heuristics produce diverse answers sets. Production similarity lies between 0.2 and 0.42, while transportation similarity moves between 0.16 and 0.23 among all heuristics. The variability for transportation is higher in general than for production.

7 Conclusions and Future Work

We addressed the problem of exploring solution spaces for global logistics problems. As a representative industrial use case from the aerospace domain, we considered the development of the next generation Airbus single-aisle aircraft. More precisely, we proposed a characterization of the global logistics problem, alongside a formalization in ASP. In addition, we introduced an encoding of essential domain knowledge via facts, integrity constraints and heuristics to direct the solver toward promising configurations. Further, in an experimental evaluation, we provided an assessment according to the KPIs and an examination of model variability.

This work demonstrates that by means of domain heuristics, diverse answers sets can be produced. Further, we showed that the incorporation of domain knowledge can produce more suitable answer sets in a shorter time, e. g., with (**preferred sites**), even the first models already exhibited significantly decreased production costs compared to other heuristics. The heuristic (**min back-forth**) leads to answer sets with the lowest transportation costs while some heuristics can be relatively neutral or counterproductive with regards to the KPIs. The application of multiple heuristics looks very promising in terms of producing good answer sets quickly, but consumes more time to find models.

A preliminary assessment with an industrial system designer shows that the produced models are valid, useful and often desirable. If solutions were undesirable the cause was tackled with additional knowledge incorporated into the program with additional facts or constraints, leading to better results. This approach is promising for defining requirements ad-hoc, which allows for a 'human in the loop' approach. In particular, it can support industrial architects in the iterative and incremental architecture definition process by investigating different solutions, gaining new insights and feeding them back as new knowledge to better configure the system and receive more favorable solutions.

It would be interesting to exploit existing approaches for identifying diverse answer sets as discussed in Section 2. For the future, we aim at performing a user study and assess the flexibility and expressiveness of the system under working conditions. In parallel, a search for a variety of optimal solutions can be performed with an extended timeout to see if the system can converge to an optimum with the use of domain knowledge.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped improve the quality of this paper. We thank Stefan Richter from Airbus for granting access to the computing server and for his technical assistance throughout the experimental process. The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grant 465447331; project "Explainable Belief Merging, EBM").

References

Airbus. 2025a. Airbus digital twin. https://www.airbus.com/en/newsroom/stories/2025-04-digital-twins-accelerating-aerospace-innovation-from-design-to-operations. 20.05.2025.

Airbus. 2025b. Airbus our worldwide presence. https://www.airbus.com/en/about-us/our-worldwide-presence. 20.05.2025.

Airbus. 2025c. Airbus production. https://www.airbus.com/en/products-services/commercial-aircraft/the-life-cycle-of-an-aircraft/production. 20.05.2025.

Airbus. 2025d. Airbus transportation fleet. https://www.airbus.com/en/newsroom/stories/2023-10-building-a-lower-emission-maritime-transport-fleet. 20.05.2025.

Alrabbaa, C.; Rudolph, S.; and Schweizer, L. 2018. Faceted answer-set navigation. In Benzmüller, C.; Ricca, F.; Parent, X.; and Roman, D., eds., *Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings*, volume 11092 of *Lecture Notes in Computer Science*, 211–225. Springer.

Arista, R.; Zheng, X.; Lu, J.; and Mas, F. 2023. An ontology-based engineering system to support aircraft manufacturing system design. *Journal of Manufacturing Systems* 68:270–288.

Böhl, E., and Gaggl, S. A. 2022. Tunas - fishing for diverse answer sets: A multi-shot trade up strategy. In Gottlob, G.; Inclezan, D.; and Maratea, M., eds., *Proceedings of the 16th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR 2022)*, volume 13416 of *Lecture Notes in Computer Science*, 89–102. Springer International Publishing.

Böhl, E.; Ellmauthaler, S.; and Gaggl, S. A. 2024. Winning snake: Design choices in multi-shot ASP. *Theory and Practice of Logic Programming* 24(4):772–789.

Böhl, E.; Gaggl, S. A.; and Rusovac, D. 2023. Representative answer sets: Collecting something of everything. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Radulescu, R., eds., *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, 271–278. IOS Press.

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

Buccafurri, F.; Leone, N.; and Rullo, P. 2000. Enhancing disjunctive datalog by constraints. *Knowledge and Data Engineering, IEEE Transactions on* 12:845 – 860.

Comploi-Taupe, R.; Friedrich, G.; Schekotihin, K.; and Weinzierl, A. 2023. Domain-specific heuristics in answer set programming: A declarative non-monotonic approach. *Journal of Artificial Intelligence Research* 76.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms*. MIT Press, 3rd edition.

Dachselt, R.; Gaggl, S. A.; Krötzsch, M.; Méndez, J.; Rusovac, D.; and Yang, M. 2022. Nexas: A visual tool for navigating and exploring argumentation solution spaces. In Toni, F., ed., *Proceedings of the 9th International Conference on Computational Models of Argument (COMMA 2022)*, volume 220146 of *FAIA*, 116–127. IOS Press.

Dietz, E.; Philipp, T.; Schramm, G.; and Zindel, A. 2023. A logic programming approach to global logistics in a co-design environment. In Pontelli, E.; Costantini, S.; Dodaro, C.; Gaggl, S. A.; Calegari, R.; d'Avila Garcez, A. S.; Fabiano, F.; Mileo, A.; Russo, A.; and Toni, F., eds., *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*, volume 385 of *EPTCS*, 227–240.

Falkner, A.; Friedrich, G.; Schekotihin, K.; Taupe, R.; and Teppan, E. C. 2018. Industrial applications of answer set programming. *KI - Künstliche Intelligenz* 32(2-3):165–176.

Ferrogalini, M. 2019. Modeling and simulation@ airbus a fundamental digital transformation axis across product, manufacturing and support in service. In *Presentation at MODELS Conference*.

Fichte, J. K.; Gaggl, S. A.; and Rusovac, D. 2022. Rushing and strolling among answer sets - navigation made easy. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*, volume 36 of *5*, 5651–5659.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. clasp: A conflict-driven answer set solver. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 260–265. Springer.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Morgan & Claypool.

Gebser, M.; Kaufmann, B.; Romero, J.; Otero, R.; Schaub, T.; and Wanko, P. 2013. Domain-specific heuristics in answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 350–356.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Lindauer, M.; Ostrowski, M.; Romero, J.; Schaub, T.; Thiele, S.; and Wanko, P. 2019a. Potassco guide version 2.2. 0.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019b. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19(1):27–82.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4):365–386.

Helle, P.; Strobel, C.; Mitschke, A.; Schamai, W.; Riviere, A.; and Vincent, L. 2008. Improving systems specifications a method proposal. In *Proceedings of CSER 2008 Conference*.

Hua, Q.-S.; Wang, Y.; Yu, D.; and Lau, F. C. 2010. Dynamic programming based algorithms for set multicover and mul-

tiset multicover problems. *Theoretical Computer Science* 411(26-28):2467–2474.

Janhunen, T., and Niemelä, I. 2016. The answer set programming paradigm. *AI Magazine* 37(3):13–24.

Jirkovský, V.; Kubalík, J.; Kadera, P.; Schirrmann, A.; Mitschke, A.; and Zindel, A. 2025. Towards resilient and sustainable global industrial systems: An evolutionary-based approach.

Martins, J. R. R. A., and Ning, S. A. 2021. *Engineering design optimization*. Cambridge and New York NY: Cambridge University Press.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273.

Rajaratnam, D.; Schaub, T.; Wanko, P.; Chen, K.; Liu, S.; and Son, T. C. 2023. Solving an industrial-scale warehouse delivery problem with answer set programming modulo difference constraints. *Algorithms* 16(4).

Reiter, R. 1978. On closed world data bases. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*. Springer. 55–76.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Salton, G., and McGill, M. J. 1983. *Introduction to modern information retrieval*. McGraw-Hill computer science series. New York: McGraw-Hill Book Comp.

Thulasiraman, K., and Swamy, M. N. 2011. *Graphs: theory and algorithms*. John Wiley & Sons.

Zheng, X.; Hu, X.; Arista, R.; Lu, J.; Sorvari, J.; Lentes, J.; Ubis, F.; and Kiritsis, D. 2024. A semantic-driven tradespace framework to accelerate aircraft manufacturing system design. *Journal of Intelligent Manufacturing* 35(1):175–198.