Working Paper Series
of the Chair of Management Science /
Operations and Supply Chain Management
University of Passau

Resolving the curse of poor data quality: Optimization perspective on data collection and validation

Benedikt Finnah, Jochen Gönsch
Mercator School of Management, University of Duisburg-Essen, Lotharstraße 65, 47057 Duisburg, Germany,
benedikt.finnah@uni-due.de, jochen.goensch@uni-due.de

Alena Otto
School of Business, Economics and Information Systems, University of Passau, Innstraße 27, 94032 Passau, Germany,
alena.otto@uni-passau.de

Companies possess large quantities of poor-quality data. The validation of which is costly, especially if the required data accuracy is high. In this paper, we focus on a specific subset of such data which is common across many industries and areas of business — data on precedence relations between tasks.

We formulate the data collection and validation problem (*DCVP*) that aims to optimize the business result in some baseline planning problem. However, the initial data, which serves as basis for the quantification of the baseline problem constraints, is partially incorrect. Restricted by the available time budget, an expert dynamically receives queries about specific data entries and corrects or validates them. Thus, the DCVP is an online optimization problem which looks for an optimal policy (interview policy) of stating such queries to optimize the result of the baseline planning problem. Relevant solutions for the baseline planning problem *must be* feasible, although initial data entries may be incorrect.

We analyze the properties of the DCVP, model the DCVP as a dynamic program, and suggest a customized least squares temporal difference algorithm *LSTD*. In extensive computational experiments, the LSTD interview policy considerably outperforms alternative ones. In a case study of a leading automobile manufacturer's assembly line, this policy substantially reduces the station's idle time after selectively addressing only about 8% of the data entries.

Key words: data collection and validation problem, online optimization, assembly line balancing, project scheduling, precedence relations

History: This paper was first submitted on ... and has been with the authors for ...

1. Introduction

Poor-quality data costs companies trillions of dollars each year in terms of data correction, validation, and foregone benefits resulting from managerial decisions based off of insufficient data (Davenport 1997, Escobar et al. 2021, Redman 1998). About 84% of CEOs are concerned about the quality of data used for decision making in their companies, according to the Forbes Insights and KPMG report. Insufficient data quality may also undermine future technologies such as IoT ([Forbes, KPMG] 2016). Often, the only way to validate and correct the data is to do so manually, which is time consuming and tedious. Consequently, research on how to improve the efficiency of expert involvement in collecting and correcting data is of utmost importance.

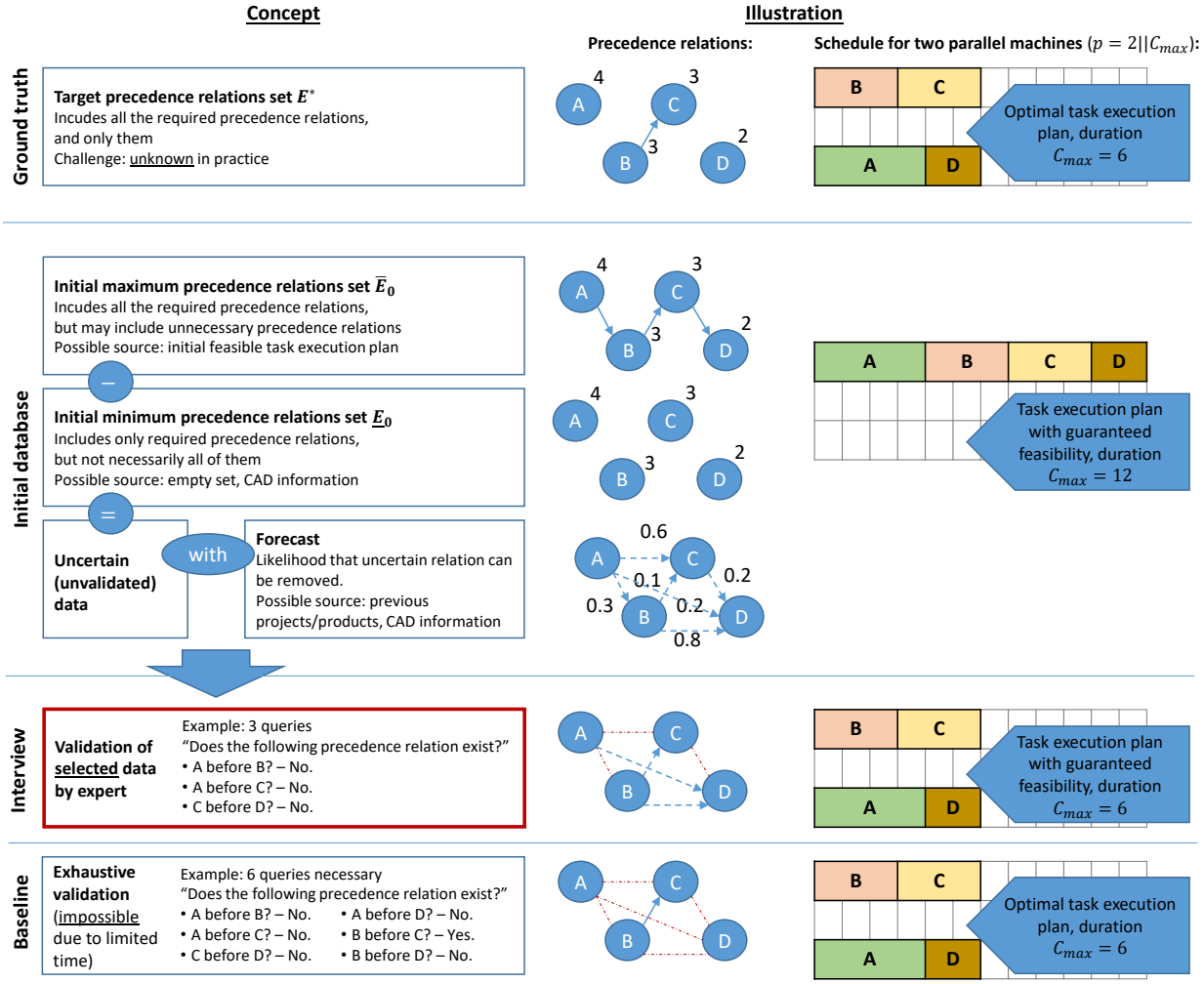
The issue of poor-quality data is especially acute whenever the data describes *hard constraints*, as is the case with the data on precedence relations between tasks. We turn to precedence relations, because they are indispensable in almost all parts of business, including product design (de Fazio et al. 1999, Krishnan and Ulrich 2001), assembly and manufacturing (Boysen et al. 2009), project scheduling (Demeulemeester and Herroelen 1992, Manrique et al. 2018), construction (Wang et al. 2018), as well as transportation and logistics (Gertsbakh and Serafini 1991, Hauptmeier et al. 2001). If task i is a predecessor of task j, then task i must be completed before we can start task j. Precedence relations, or simply *relations*, are usually stored as a set of ordered pairs, or *partially ordered set (poset)*, $E = \{(i, j) \mid i \text{ is predecessor of } j\}$. Obviously, if some relations are missing from E, the execution plan for tasks generated by a decision support system may be infeasible. On the other hand, including unnecessary relations in E should be avoided because they lead to unnecessary constraints, and the potential loss of optimality, including the wasting of resources.

Traditionally, the barriers to data collection and validation by experts are widely perceived as very high. For example, the final car assembly consists of more than $n = 1,000$ tasks and precedence relations of over $\frac{n(n-1)}{2} = 499,500$ task pairs have to be clarified. If an expert spends around 3 minutes per task pair, we would end up with an expert mission working 24 hours /7 days a week for almost three years! Not surprisingly, the idea of investing additional expert time in validating data collected by unreliable (automated) methods is generally dismissed by practitioners as only being reasonable 'with simpler and less complicated products' (Kashkoush and ElMaraghy 2014).

With this paper, we seek to shake this stereotype and elaborate on a *focused validation of carefully selected parts* of unreliable data. The results suggest that with this focused validation, a realistic amount of expert time can considerably improve the quality of management decisions. This is due to the reasons seem to be overlooked in practice and in vast parts of the literature:

- *Anticipation of logical implications.* Before knowing the expert's answer, we can already estimate the amount of additional information (assertions) that we will be able to derive from it, so that, given the limited expert time, we can formulate a small number of carefully selected questions.

Figure 1 Underlying Concept of the Data Collection and Validation Problem.



Note. Given an initial database, the expert has to validate selected data. In the graphs, nodes denote tasks, (integer) node labels denote their processing time, arcs are precedence relations, and arc labels state the probability that the precedence relation is not necessary. A simple example using the two-machine scheduling problem illustrates that an intelligently structured expert interview with three queries may be equivalent to an exhaustive data validation (of six queries) in terms of the business result.

- *Exploitation of automatically collected data.* We can leverage noisy data by using, for instance, automatically collected data as an indication of the experts' most probable answer.
- *Anticipation of optimality.* We are actually interested in a subset of precedence relations that are relevant for an optimal execution plan of tasks, which may be only a small portion of the initial poset.

We state *the data collection and validation problem (DCVP)*. DCVP aims at minimizing certain objective function $F(\cdot)$ over the unknown *target* (true) poset E^* (see Figure 1). The input to this problem is some budget of expert time T and an initially noisy poset E_0, which includes at least one feasible execution plan of tasks. The objective is to choose a sequence of interview questions

to the experts, each using up some of the time budget T, in such a way that a poset $\bar{E}$ is built with $F(\cdot|\bar{E})$ as small as it can be within budget T and $E^* \subseteq \bar{E}$.

The initial poset E_0 may just contain the currently deployed execution plan of tasks, such as the start-of-production task sequence in assembly or the default project plan in construction. This data may be further refined with data mining techniques based on the computer-aided design (CAD-) data (Qu and Xu 2013, Zha et al. 1998) or the bills of material (Niu et al. 2003) in assembly, user demonstrations in robotics (Billard et al. 2008, Pardowitz et al. 2005) or on the data from similar projects (Kashkoush and ElMaraghy 2014).

The contribution of this paper is as follows:

- To the best of our knowledge, we are the first within literature to formulate and analyze the problem of data validation by experts as an optimization problem. We formulate the DCVP for the data on precedence relations between tasks.
- Since the DCVP depends on the specific baseline planning problem at hand, we state a universally-valid problem counterpart for the DCVP – DCVP-gen. As a consequence, the proposed interview policies and the results of our analysis can be directly applied to a wide range of applications. We state the DCVP as a dynamic program and prove that it is correct.
- We analyze the computational complexity of the DCVP-gen and identify a trivial relevant-for-practice subproblem. This is an interesting result, since stochastic multi-stage optimization problems are generally hard to solve.
- We formally analyze the properties of the non-trivial uncertainty set of the DCVP-gen, which the decision maker actively and dynamically influences by selecting a specific interview question.
- We propose a least squares temporal difference algorithm (LSTD) to solve the DCVP-gen.
- We provide rule-of-thumb approximation formulas to estimate expected benefits of partial data validation in case some characteristics of the true (target) poset are known. Computational tests confirm that the proposed approximations are tight, and thus, informative.
- In extensive computational experiments, we illustrate the benefits of partial data validation and the advantage of the LSTD interview policy.
- We also examine the validity of our ideas in a case study, featuring a portion of the final assembly of a leading automobile manufacturer. We show that partial validation guided by the LSTD algorithm results in a significant improvement in the business result.

The paper is structured as follows: Section 2 reviews the relevant literature. Section 3 defines the DCVP formally, discusses its assumptions, proposes a dynamic program, and examines its analytical properties. The formulated dynamic program is solved with an approximate dynamic programming algorithm in Section 4. Section 5 reports on extensive computational experiments, including a real-world case study. We conclude with an outlook in Section 6.

2. Literature review

The problem considered in this paper can be interpreted as a multistage adjustable robust optimization problem, in which the uncertainty set depends on the previously taken decisions (posed queries to the expert) (see Ben-Tal et al. (2009) and Yanıkoğlu et al. (2019) for an overview). It is an adjustable problem, because the information is incomplete and is revealed dynamically throughout the course of the interview, the latter decisions can adjust to the information revealed so far. It is a robust optimization problem, because *after* the interview we seek a best possible solution under the worst possible circumstances, in other words, we seek a solution with guaranteed feasibility.

To the best of our knowledge, we are the first to formulate an adjustable robust optimization problem of this type. Classically, the uncertainty set is considered as given or is estimated in a simplistic way. Recent works from the fields of operations research and engineering have developed methods for deriving the uncertainty set from a pool of available historical data (see, e.g., Bertsimas et al. (2017), Han et al. (2021)). A major challenge in this stream is the adjustment of the conservatism and computational complexities of the resulting optimization problem. We by contrast, rely on costly data collection instead of a pool of available data.

Our work is also related to the vast literature on information acquisition and learning (see, e.g., Powell and Ryzhov (2012) for an overview), which contrasts *passive learning*, or learning by observation, and *active learning*, in which decisions influence the outcome of the learning. The focus of this literature is on the trade-off between exploration and exploitation, i.e. whether to optimize on the available data (exploitation) or acquire costly additional information at an extra cost (exploration). In this context, we seek an active learning strategy, but do not face the exploration/exploitation trade-off in the classical sense as we only incur a reward at the end of the horizon, that is, after the interview.

Furthermore, we can interpret pairs of tasks with a precedence relation $(i, j) \in E$ as a directed edge and depict E and V as a directed acyclic graph $G = (V, E)$. Then the DCVP can be interpreted as learning a network from data, which is relevant for such applications as software testing, biology, and disaster relief. In biology, the data collection problem aims to discover relationships between genes (nodes), which are edges that visualize protein-mediated regulatory influences such as promotion or inhibition of gene expression. In contrast to the DCVP, the edges of the network are derived with a certain probability by statistical analysis based on the (stochastic) output of a limited number of treatments, each activating or inhibiting a bunch of genes (De Jong 2002, Lee et al. 2009, Markowetz and Spang 2007). In software testing, nodes on the graph are the executed portions of the code and the execution sequence of the code corresponds with the directed edges. The aim is to reduce the expert's time needed to localize a fault in the code. In contrast to the DCVP, the edges in this network are given and the aim is to learn the status of the nodes (faulty

or correct) by examining suspicious portions of the code (see Wong et al. 2016, for an overview). In disaster relief, the nodes are clusters of households or communities while the edges are the streets. After a disaster, the streets may be blocked and the needs of the households are unknown. The aim of the need assessment in disaster relief is to clarify these unknown parameters (Hlady et al. 1994, Huang et al. 2013, e.g.), see Kovacs and Moshtari (2019) for an overview. The arising problems differ very much from the DCVP. For instance, the status of the blocked edges can only be revealed if some vehicle approaches them, making routing aspects a central component.

If we turn to the literature on precedence relations, the DCVP resembles the so-called *assembly sequence problem (ASP)*, the problem of finding all or a subset of feasible sequences given the interactive input of experts or a data set of mathematical formulas implicitly describing precedence relations and further constraints (Wang et al. 2009). The motivation for the ASP comes from the design stage of a product. It aims to either assist decision makers in finding an initial feasible assembly sequence for the start of production, or investigates the impact of alternative design decisions as part of DFX approaches (such as design for manufacturing or assembly). Correspondingly, much of the literature focuses on documenting and structuring the process of interaction with experts, such as role assignment, data structures, semantic labeling of data, documentation of the data for further re-use, and visualizations (Demoly et al. 2011, Kardosh et al. 2020, Lin and Chang 1993, Sanders et al. 2009, Su 2009). In recent years especially, a number of papers on the ASP have concentrated on the automated extraction of feasible sequences from the design information on part connections (e.g., from CAD), by applying formal engineering analysis such as motion analysis of parts, or stability analysis of the assembled intermediate products (Deepak et al. 2018, Kashkoush and ElMaraghy 2014, Vigano and Gómez 2013, Wan et al. 2017). However, a number of precedence relations in assembly cannot be derived from the CAD-information, especially if soft components such as clips and gasket rings are involved, or human factors have to be considered, such as accessibility and visibility of the area of work (Antani et al. 2014, Arun and Rao 2010, Vigano and Gómez 2013). Therefore, the resulting data has to be validated by experts.

The articles on the ASP that feature interactions with experts focus on guaranteeing completeness of the collected information and do not usually impose any constraints on the available expert time or cost, which is in contrast to the research question of this article (Bourjault 1987, de Fazio and Whitney 1987, Homem de Mello and Sanderson 1991, Lambert 2006, Wang and Liu 2010).

Bozhko (2020), Lagriffoul et al. (2012), Rodrígez et al. (2019, 2021), and Srivastava et al. (2014) come the closest to the problem statement of this article. Bozhko (2020) aims to minimize the number of time-consuming geometric tests that reveal whether certain assembly states, depicted as nodes in the so-called Hasse diagram, are feasible. The result of a specific geometric test can be extended to similar assembly states. The paper models the results of a geometric test as an

exogenous random variable ('nature') and suggests that different structures of the underlying graph (Hasse diagram) favor different selections and sequences of geometric tests. Bozhko (2020) proposes no formal analysis or optimization approach for the dynamic selection of tests, rather he examines the performance of several simple decision rules in computational experiments on graphs with different structures. Furthermore, Lagriffoul et al. (2012), Rodrígez et al. (2019, 2021) and Srivastava et al. (2014) propose ways to deduce additional information from each expert answer in the context of robotic assembly. The expert in this case is an extensive simulation environment, which evaluates candidate assembly sequences. If the current candidate assembly sequence is infeasible, then either a human expert or some algorithm tries to specify a 'no-goods' constraint, which possibly surmises further infeasible sequences. The general approach resembles constraint propagation, but the derived insights are application-specific and rely, for instance, on the mechanics of motion and on reachability analysis.

To sum up, to the best of our knowledge, in contrast to the DCVP, the literature on the ASP does not consider expert time as a limited resource. Most of the literature focuses on the automated and (in practice) unreliable extraction of information from the available sources, such as CAD data. Unlike the DCVP, the available studies that include expert interviews do not anticipate the potential value of the experts' answers on the final optimization (i.e., on the business outcome).

The current work builds upon the research of Klindworth et al. (2012) and Otto and Otto (2014). Klindworth et al. (2012) introduced the notions of the maximum and the minimum graph, which are akin to the notions of the maximum and the minimum posets described below. As part of their study, Otto and Otto (2014) formulated initial ideas on how to use interviews in the construction of precedence graphs in assembly.

In summary, this paper is, to the best of our knowledge, the first in literature to examine data collection and validation by experts under limited time budget as an optimization problem.

3. The data collection and validation problem

We start with formal statements of the DCVP and of its general, application-independent counterpart DCVP-gen in Sections 3.1 and 3.2, respectively. Section 3.3 examines the implied assumptions. We focus on the DCVP-gen in all our further discussions and state it as a dynamic program in Section 3.4. Section 3.5 concludes with some analytical properties of the formulated problem.

3.1. Problem statement for the DCVP

We start with defining the notions of the transitive closure $t^+(E)$ and the transitive reduction $t^-(E)$ of a poset E, which will be useful in our further discussions.

Definition 1. **Transitive closure**. The transitive closure $t^+(E)$ of a poset E is recursively defined as $t^+(E) = \{(i,j) \in V \times V \mid (i,j) \in E \vee \exists k \in V : (i,k) \in t^+(E) \wedge (k,j) \in t^+(E)\}$.

If the precedence relations $(i,k) \in E$ and $(k,j) \in E$ exist, this implies there is a transitive or indirect precedence relation between the tasks i and j. The transitive closure $t^+(E) \subseteq V \times V$ is the smallest transitive superset of E. It contains E and all indirect precedence relations induced by E.

DEFINITION 2. **Transitive reduction**. The transitive reduction $t^-(E)$ of a poset E is defined as $t^-(E) = \underset{E' \mid t^+(E')=t^+(E)}{\arg\min} |E'|$, i.e. it is the smallest poset sufficient for the derivation of the transitive closure of E.

The complexity of computing $t^+(E)$ and $t^-(E)$ is polynomial in $|E|$. Moreover, the transitive reduction and the transitive closure are unique (Aho et al. 1972).

Now, we define *DCVP* formally as follows: Let $V = \{1, \ldots, n\}$ denote a set of tasks with an associated transitive and acyclic *target poset* $E^* = \{(i,j) \in V \times V \mid i \text{ is a predecessor of } j\}$.

Transitive means that $E^* = t^+(E^*)$. *Acyclic* implies that if E^* is visualized as a graph $G = (V, E^*)$, in which tasks form nodes and precedence relations are depicted as edges, then this graph is topologically sortable.

We seek to minimize an objective function $F(\cdot|E^*)$, which has the following obvious properties:

- The action space is decreasing in E^*, because precedence relations E^* limit feasible execution sequences of the tasks, formally:

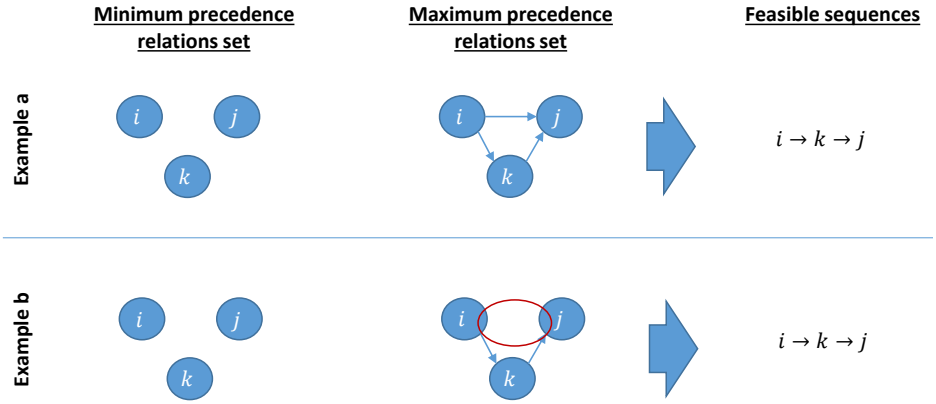
$$E' \subseteq E'' \implies F(\cdot|E') \leq F(\cdot|E'') \tag{1}$$

- The removal of certain precedence relations does not improve $F(\cdot|E)$. For example, if $(i,j) \notin E$, but there is $k \in V$ such that $(i,k) \in E$ and $(k,j) \in E$, we cannot plan as if i and j were independent: We do not know whether such independence is due to (i,k) being independent, or (k,j) being independent, or both (see Figure 2). In other words:

$$E' \subseteq E'' \text{ with } t^+(E') = t^+(E'') \implies F(\cdot|E') = F(\cdot|E'') \tag{2}$$

The challenge now is that the target poset E^* is unknown. However, we have bounds $\underline{E}$ and $\bar{E}$ such that $\underline{E} \subseteq E^* \subseteq \bar{E}$. We call $\bar{E}$ the *maximum poset* and $\underline{E}$ the *minimum poset*, and assume the initial bound $\bar{E}_0$ to be transitive and acyclic. We are conservative in the sense that given this knowledge, we must resort to optimizing $F(\cdot|\bar{E})$. Moreover, we dispose of an oracle (the expert). The oracle accepts only simple and error-resistant questions. More precisely, we can ask the oracle about each pair $(i,j) \in V \times V$, which is related to a *direct* precedence relation in the current maximum poset $\bar{E}$, i.e. $(i,j) \in t^-(\bar{E})$ (see the discussion in Section 3.3). With probability p_{ij}, the oracle's answer is $(i,j) \notin E^*$, otherwise we know $(i,j) \in E^*$. The oracle's answers are used to update the sets $\underline{E}$ and $\bar{E}$. Each question is associated with a cost of τ_{ij} and we have a total (time) budget of T_0.

Figure 2 **The Number of Sequences and the Maximum Precedence Relations Set $\bar{E}$.**



Note. Knowing that i and j are independent does not provide additional flexibility, since we do not know whether such independence is due to (i,k) being independent, or (k,j) being independent, or both. As a consequence, the number of sequences known to be feasible does not increase.

This results in the optimization problem of efficiently using the oracle. More precisely, we are seeking a dynamic policy π that, at each point in time, and depending on our current state of knowledge (the posets $\underline{E}$ and $\bar{E}$), and on the remaining time budget, prescribes the next question to ask in order to obtain the optimal expected objective value after depleting the time budget:

$$\min_{\pi} \mathbb{E}^{\pi}[F(\cdot|\bar{E}_{\pi})] \tag{3}$$

where the expectation is with regards to $\bar{E}_{\pi}$, which denotes the *tightest* maximum poset after having interviewed the oracle within the time budget of T_0 following the policy.

3.2. Generalized problem statement: DCVP-gen

The functional form of $F(\cdot|E)$ depends on the studied application and may contain additional parameters and imply further constraints. Consider the makespan minimization in the project scheduling given resource endowments and resource consumption rates; or the minimization of the number of stations given the cycle time and the duration of tasks in the assembly line balancing. Therefore, it is of interest to study a general, application-independent variant of the DCVP.

Since the oracle only accepts questions to direct precedence relations in the current maximum posit $\bar{E}$, any positive answer of an oracle in the DCVP leads to a removed relation and, thus, to an improvement in terms of Properties 1 and 2. Therefore, in the DCVP-gen, we simply maximize the expected number of the positive answers of the oracle within the time budget T_0:

$$\arg\min_{\pi} \mathbb{E}^{\pi}[|\bar{E}_{\pi}|] = \arg\max_{\pi} \mathbb{E}^{\pi}\left[|\bar{E}_0| - |\bar{E}_{\pi}|\right] \tag{4}$$

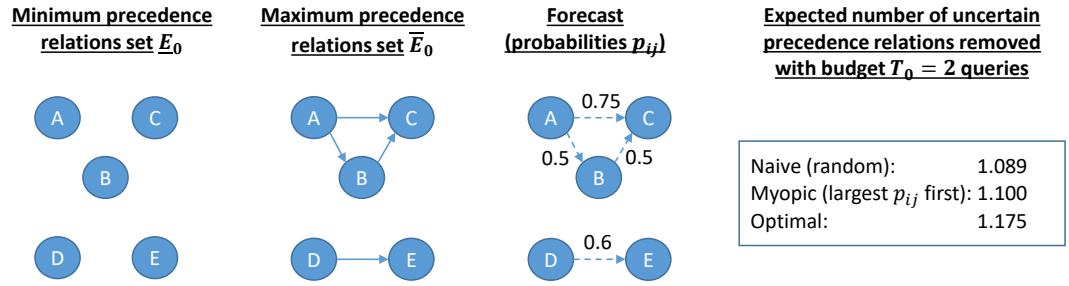
Recall that $\bar{E}_0 = tc(\bar{E}_0)$ according to the problem statement and $\bar{E}_{\pi} = tc(\bar{E}_{\pi})$ according to Proposition 2 (Section 3.5). As the set $\bar{E}_0$ denotes the possible precedence relations known initially, the

number of effectively removed (cf. Property 2) unnecessary precedence relations is $|\bar{E}_0| - |\bar{E}_\pi|$. Figure 3 provides a small instance of the DCVP-gen for illustration, in which the optimal policy is expected to remove about 10% more precedence relations than intuitive alternative policies.

Observe that Function 4 is equivalent to the minimization of the so-called *order strength (OS)*. Order strength $OS(E) \in [0,1]$ of a poset E is the share of task pairs with a precedence relation in the total number of task pairs: $OS(E) = \frac{|tc(E)|}{\frac{n\cdot(n-1)}{2}}$, where $n = |V|$ is the number of tasks. The order strength is known as one of the best-known low-polynomial approximations for the number of feasible task sequences, which are feasible with respect to poset E (**?**) (the computation of this number is #-P (**?**)). Moreover, it is a widely recognized indicator to characterize the solution space and the hardness of the problem instances in the project scheduling and the assembly line balancing literature (**?**).

In our case study on line balancing for the final assembly of automobiles in Section 5.5, we will numerically examine the relationship between the true $F(\cdot|\bar{E}_\pi)$ and the surrogate $(|\bar{E}_0| - |\bar{E}_\pi|)$ objective functions. We will show that the maximization of the surrogate objective translates into a significant improvement of the final task execution plan as measured by the actual objective function $F(\cdot|\bar{E}_\pi)$.

Figure 3 An Instance of the DCVP-gen and Three Alternative Policies.



Note. The instance has five tasks $V = \{A, B, C, D, E\}$ with four precedence relations in $\bar{E}_0$ and empty $\underline{E}_0$. Arc labels denote probabilities p_{ij} that the oracle's answer is $(i,j) \notin E^*$. Costs are constant $\tau_{ij} = 1$ and the total time budget $T_0 = 2$ allows to state two queries. Please note that stating a query on (A,C) is only possible after either (A,B) or (B,C) are already known to be independent. A naive policy that asks randomly deletes 1.089 precedence relations in expectation. A myopic policy prioritizing higher probabilities deletes slightly more precedence relations (1.100). In the optimal interview policy, we first ask about (A,B) (or, equivalently, (B,C)). If this is deleted, we ask about (A,C), else we ask about (D,E). The optimal interview deletes 7 % more (1.175) precedence relations than the myopic policy. Figures 12 - 14 in the appendix provide details on these policies.

3.3. Discussion of assumptions

In this section, discuss the assumptions of the DCVP-(gen).

Assumption 1. *An initial maximum precedence relations set $\bar{E}_0$ and an initial minimum precedence relations set $\underline{E}_0$ are known. The former is transitive and acyclic.*

The consequence of acyclicality for $\bar{E}_0$ is that the direction of a possible relation – whether i is a possible predecessor of j or j is a possible predecessor of i – is known for each pair of tasks. The consequence of transitivity is that all known independencies can be exploited as they lead to more feasible sequences – the case of two tasks being independent but required to be executed in a certain order (see Figure 2 for an example) cannot occur. Transitivity and acyclicality are usually the case in practice, for example, because at least one feasible solution of $F(\cdot|E^*)$ is known (cf. Proposition 1 in Section 3.5).

Acyclicality of $\bar{E}_0$ obviously implies the same for $\underline{E}_0$. Cyclicality of $\underline{E}_0$ would imply that no feasible sequence exists. We do not require transitivity for $\underline{E}_0$ and compute $tc(\underline{E}_0)$ whenever required. The trivial minimum precedence relations set $\underline{E}_0 = \emptyset$ is always known.

ASSUMPTION 2. *Precedence relations in the true set E^* are transitive and acyclic.*

This assumption is straightforward (Baybars 1986, Kolisch et al. 1995, Pinnoi and Wilhelm 1998).

ASSUMPTION 3. *A query to the oracle addresses exactly one precedence relation.*

Possible formulations of the interview questions can oscillate between complicated questions with extended answers like "What is the list of all precedence relations?" and simple questions with less informative answers like "Is i a predecessor of j?". We assume the latter type of questions, because questions that are simple and easy to understand greatly reduce the risk of incorrect answers and the undesirable prospect of oversights, which are especially pronounced in the event of large projects/products. This assumption is in line with Otto and Otto (2014).

ASSUMPTION 4. *The time τ_{ij} needed to reveal the randomness of (i,j) is known and deterministic.*

Usually, the number of experts required to answer a query serves as a good approximation of the required time τ_{ij}

ASSUMPTION 5. *The probabilities p_{ij} for the oracle's answers are known.*

As discussed in Section 2, a plenitude of methods for estimating the true precedence relations set has been suggested in the literature. These include experience with similar products/projects, user demonstrations, and CAD data. Although these methods are susceptible to omissions, they provide indications of a *likely* answer from the oracle and can be used to quantify probabilities p_{ij}.

Observe that these are 'subjective' probabilities, meaning that they are conditional on different and possibly not well-defined populations (such as sets of similar task pairs with and without a precedence relation in other projects of different sizes and functions) and generally cannot be updated based on the oracle's answers.

ASSUMPTION 6. *We can only ask the oracle about precedence relations $(i,j) \in tr(\bar{E})$. The oracle's answers are correct.*

Motivated by our discussions with practitioners, we restrict the set of valid questions to $tr(\bar{E})$ to eliminate possible sources of error. Indeed, to answer a question to some (i,j), for which one or more indirect precedence relations $i \to k \to j$ have to be cleared, is obviously difficult and susceptible to errors.

3.4. Dynamic program for the DCVP-gen

In this section, we model the DCVP-gen as a *dynamic program (DP)* where each stage corresponds to one query posed to the oracle. We do not limit the number of stages upfront. Nevertheless, the problem is finite because at some point we cannot ask further questions since either the time budget has been depleted, or all precedence relations have been clarified. In the following, we define the state variable, the action space, the contribution function, and the resulting Bellman equation. The objective function in the formulated DP is to maximize the expected number of effectively removed unnecessary precedence relations (cf. Expression (4) in Section 3.1).

The state variable S contains everything the interviewer's decision depends on. We limit ourselves to the consideration of state variables that adhere to Condition (6) (later, we will show that this condition is satisfied by the initial state according to the above assumptions as well as by all subsequent states):

$$S = (\bar{E}, \underline{E}, T) \in \mathcal{S} \text{ with} \tag{5}$$

$$\mathcal{S} = \{(\bar{E}, \underline{E}, T) \in V^4 \times \mathbb{R} \quad | \quad \underline{E_0} \subseteq \underline{E} \subseteq \bar{E} \subseteq \bar{E}_0, \bar{E} \text{ is acyclic and transitive}, 0 \leq T \leq T_0\} \tag{6}$$

The interviewer's decisions depend on the currently known database. More precisely, the decisions depend on the set of possible precedence relations $\bar{E}$ and the set of precedence relations known for sure $tc(\underline{E})$, which are those in $\underline{E}$ and those derived from them. Furthermore, the decision depends on the remaining duration of the interview T. The interview starts with initial state $S_0 = (\bar{E}_0, \underline{E}_0, T_0)$. Recall that T_0 is the maximum duration of the interview.

For each state, we formally define the set of feasible interview questions as follows:

$$X(S) = \left\{(i,j) \in V \times V | (i,j) \in tr(\bar{E}) \setminus tc(\underline{E}), \tau_{ij} \leq T\right\} \tag{7}$$

Based on Assumption 6, the interviewer asks only questions concerning elements in the transitive reduction of the maximum precedence relations set $\bar{E}$. As the precedence relations in the transitive closure $tc(\underline{E})$ are known for sure, no questions are asked regarding these. Finally, the interviewer asks only questions that can be answered during the remaining time budget of the interview.

If $X(S) = \emptyset$, no question can be asked and we call state S *absorbing*. This occurs if either the time is elapsed or if all elements in the transitive reduction of the set of possible precedence relations $tr(\bar{E})$ are known for sure. The latter is equivalent to $tc(\underline{E}) = E^* = \bar{E}$.

The interviewer's stage-wise *contribution* is the number of unnecessary precedence relations removed from the maximum precedence relations set after the oracle's answer on the task pair (i, j) in state S, where the Bernoulli random variable Ω_{ij} with success probability p_{ij} denotes the oracle's answer (realization $\omega_{ij} = 1$ signals that there is no precedence relation):

$$C(S, (i,j), \Omega_{ij}) = \begin{cases} |tc(\bar{E})| - |tc(\bar{E} \setminus (i,j))| & \text{if } \Omega_{ij} = 1 \\ 0 & \text{if } \Omega_{ij} = 0. \end{cases} \tag{8}$$

In other words, the interviewer's contribution equals the number of additionally confirmed independencies derived from the oracle's answer. Corollary 2 below further quantifies the value of the contribution function in case an unnecessary precedence relation is found.

Recall that Expression 4 for the objective function requires that $\bar{E}$ is the tightest transitive maximum precedence relations set given the information available. Collorary 1 in Subsection 3.5 confirms that $\bar{E}$ always satisfies this requirement.

Finally, we formulate a recursive Bellman equation:

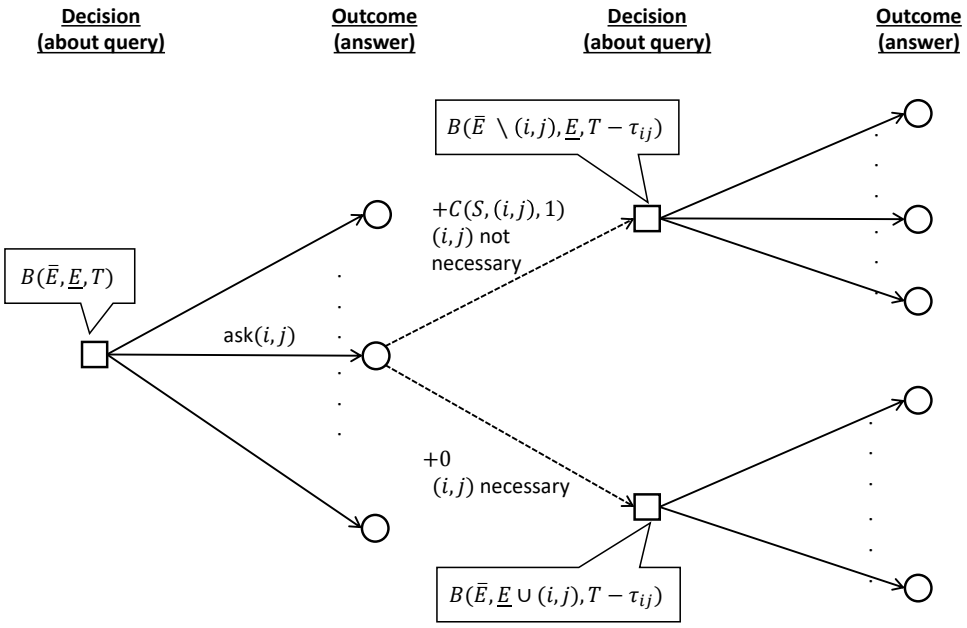
$$B(S) = \begin{cases} \max\limits_{(i,j) \in X(S)} p_{ij} \cdot (C(S, (i,j), 1) + B(\bar{S}_{ij})) + (1 - p_{ij}) \cdot B(\underline{S}_{ij}) & \forall S \in \mathcal{S} | X(S) \neq \emptyset \\ 0 & \text{else} \end{cases} \tag{9}$$

with $\bar{S}_{ij} = (\bar{E} \setminus (i,j), \underline{E}, T - \tau_{ij})$ and $\underline{S}_{ij} = (\bar{E}, \underline{E} \cup (i,j), T - \tau_{ij})$. With probability p_{ij}, the oracle reveals the independence of i and j; the contribution equals $C(S, (i,j), 1)$ (see Expression (8)), the interviewer updates the maximum precedence relations set $\bar{E}$ and transits to state $\bar{S}_{ij}$. Otherwise, with probability $(1 - p_{ij})$, the contribution equals 0 and the interviewer transits to state $\underline{S}_{ij}$ by updating the minimum precedence relations set $\underline{E}$.

$B(S)$ is the value of state S. More precisely, it denotes the expected amount of unnecessary precedence relations that will be discovered during the rest of the interview, provided the interviewer follows an optimal policy. The value of an absorbing state equals zero. As the amount of precedence relations is bounded, the Bellman equation converges without discounting. The resulting state graph is illustrated in Figure 4.

The interviewer's decision problem is to select a query in each state (i.e. find an interview policy) so as to maximize the total expected contribution incurred when starting from initial state $S_0 = (\bar{E}_0, \underline{E}_0, T_0)$.

Figure 4 **Illustration of the State Graph.**



Note. Squares denote decision nodes, circles denote chance nodes. At decision nodes, the query with maximum expected value is selected. At chance nodes, the state's value is the expectation over the two possible events (the oracle answers yes or no, and, accordingly, we know that the corresponding precedence relation does (not) exist for sure).

3.5. Analytical results

In this section, we prove a number of useful analytical properties of the modelled DCVP-gen, including the correctness of the formulated dynamic program in Proposition 4. Afterwards, we reflect on the computational complexity of the DCVP-gen in Propositions 5 and 6. In particular, we show a realistic special case of the DCVP-gen, which is trivial to solve. This is a very interesting result in view of the complexity of general stochastic multi-stage optimization problems (e.g. Dyer and Stougie (2006), Hanasusanto et al. (2016).)

We start with formulating a condition that is needed for most of the analytical results.

Condition 1 *For the maximum precedence relations set holds* $\bar{E} = tc(\bar{E})$.

If Condition 1 applies, then the situation as illustrated in Figure 2 is impossible: although independency (i,j) is confirmed, it is still unknown whether this independency is attributed to the missing edge between (i,k) or to the missing edge between (k,j); therefore, pair (i,j) is not removed from $tc(\bar{E})$. In practice, this means that until the status of the pairs (i,k) and (k,j) can be cleared, we have to ignore the available information and treat (i,j) as a precedence relation in order to guarantee feasible solutions.

The following proposition shows that Condition 1 holds in practice, where the initial maximum precedence relations set often represents one feasible sequence of tasks or is derived from a number

of feasible sequences of tasks (Klindworth et al. 2012). An example of the data for such initial sequence is the start-of-production plan for the execution of tasks.

PROPOSITION 1. *If the set of possible precedence relations $\bar{E}_0$ is derived based on several ($L \geq 1$) feasible sequences, then $\bar{E}_0$ satisfies Condition 1.*

Proof. Let there be L feasible sequences of the tasks. Each sequence $l \in \{1, \cdots, L\}$ induces a set of possible precedence relations $\bar{E}_l$ for which obviously holds $\bar{E}_l = tc(\bar{E}_l)$.

For the set of possible precedence relations $\bar{E}_0$ holds

$$\bar{E}_0 := \bigcap_{l=1}^{L} \bar{E}_l \tag{10}$$

Now, consider two arbitrary but fixed precedence relations $(i,j),(j,k) \in \bar{E}_0$, then $(i,j),(j,k) \in \bar{E}_l \ \forall l$. Furthermore, as $\bar{E}_l = tc(\bar{E}_l)$, it follows that $(i,k) \in \bar{E}_l \ \forall l$. Therefore $(i,k) \in \bar{E}_0$. We conclude that $\bar{E}_0 = tc(\bar{E}_0)$. □

The following proposition shows that Condition 1 holds for all databases during the interview.

PROPOSITION 2. *Provided $\bar{E}_0 = tc(\bar{E}_0)$, $\bar{E} = tc(\bar{E})$ holds for all states S of the dynamic program formulated in Section 3.4, that can be reached from the initial state S_0 (i.e. for which there is a path from initial state S_0 to S in the defined state graph).*

Proof. The proof is by induction, and $\bar{E}_0 = tc(\bar{E}_0)$ forms the basis.

Let $\bar{E}$=$tc(\bar{E})$ be the maximum precedence relations set of some state S and let action $(i,j) \in X(S)$ be selected. Furthermore, let $\bar{E}'$ denote the maximum precedence relations set of the following state.

- If $\Omega_{ij} = 0$, then $(i,j) \in E^*$ and the maximum precedence relations set in the next state remains unchanged: $\bar{E}' = \bar{E} = tc(\bar{E}) = tc(\bar{E}')$.
- If $\Omega_{ij} = 1$, then $(i,j) \notin E^*$ and the maximum precedence relations set is now $\bar{E}' = \bar{E} \setminus (i,j)$.

Now assume for a proof by contradiction that there exist $i', j', k \in V$ such that $(i',k),(k,j') \in \bar{E}'$ and $(i',j') \notin \bar{E}'$. As $\bar{E} = \bar{E}' \cup (i,j)$ and $\bar{E} = tc(\bar{E})$ must follow $(i',j') = (i,j)$. Since $\bar{E}' \subset \bar{E}$ follows $(i',k) = (i,k),(k,j') = (k,j) \in \bar{E}$ and therefore $(i,j) \notin tr(\bar{E})$ which contradicts the feasible set $X(S)$. The same reasoning applies for more than one intermediate node k. □

Given the limited time of the oracle, it is important to see, whether the information that we request from her can be derived without using the oracle's precious time. For this purpose, we introduce the notion of the uncertainty set.

DEFINITION 3. **Uncertainty set**. Consider the state graph as defined in Section 3.4 for a DCVP-gen instance with the sets $\bar{E}_0$, $\underline{E}_0$ and time budget T_0. Then, the initial state S_0 in the state graph is $S_0 = (\bar{E}_0, \underline{E}_0, T_0)$. Consider also state $S = (\bar{E}, \underline{E}, T)$ that can

be reached in this state graph starting from the initial state S_0 via some interview $\mathcal{I} = \big((i_1, j_1, \omega_{i_1,j_1}), (i_2, j_2, \omega_{i_2,j_2}), \ldots, (i_p, j_p, \omega_{i_p,j_p}), \ldots, (i_P, j_P, \omega_{i_P,j_P})\big)$ consisting of P questions to task pairs (i_p, j_p) and the respective responses of the oracle $\omega_{i_p,j_p} \in \{0,1\}$, $p \in \{1, \ldots, P\}$.

The *Uncertainty set* $\mathcal{U}(S)$ is the set of task pairs whose status ($(i,j) \in E^*$ vs. $(i,j) \notin E^*$) cannot be derived from the information available, which is:

(i) initial sets $\bar{E}_0, \underline{E}_0$,

(ii) answers of the oracle $(\omega_{i_1,j_1}, \omega_{i_2,j_2}, \ldots, \omega_{i_p,j_p}, \ldots, \omega_{i_P,j_P})$ and

(iii) the known properties of the target graph (transitive and acyclic).

PROPOSITION 3. *Consider a state $S = (\bar{E}, \underline{E}, T)$ such that there is a path from the initial state S_0 to S in the state graph of the defined dynamic program, then the respective uncertainty set is $\mathcal{U}(S) = \bar{E} \setminus tc(\underline{E})$.*

Proof. The proof is in three steps. Together, the first two steps demonstrate that the uncertainty set is not too small, i.e. does not miss any uncertain relations. The third shows that it is not too big, i.e. all of its elements are really uncertain.

• First, observe that $\bar{E}$ indeed contains all possible precedence relations, that is, a precedence relation $(i,j) \notin \bar{E}$ does not exist for sure. This is obvious from the definition of $\bar{E}_0$ and the updating rule $\bar{S}_{ij}$.

• Second, observe that $tc(\underline{E})$ only contains possible precedence relations that exist for sure. Again, this is obvious from the definition of $\underline{E}_0$, the updating rule $\underline{S}_{ij}$ and the transitivity of E^*.

• Third, we need to show that all $(i,j) \in \bar{E} \setminus tc(\underline{E})$ are really uncertain. Obviously, they have not yet been asked nor initially known. However, it might be possible to derive their status from the properties of the transitivity and acyclicality of the target precedence relations set E^*. To show that the status of some task pair $(i,j) \in \bar{E} \setminus tc(\underline{E})$ is uncertain, we will construct two alternative transitive and acyclic target sets E_1^* and E_2^* with $(i,j) \in E_1^*$ and $(i,j) \notin E_2^*$. Obviously, if the construction of two such alternative target graphs is possible, we cannot infer about the status of (i,j) for sure.

Consider any $(i,j) \in \bar{E} \setminus tc(\underline{E})$ and set $E_1^* := \bar{E}$ and $E_2^* := tc(\underline{E})$. Both E_1^* and E_2^* are transitive and acyclic. Indeed, since $\bar{E}_0$ is acyclic, E_1^* and E_2^* are acyclic as subsets of $\bar{E}_0$. E_2^* is obviously transitive. E_1^* is transitive by Proposition 2. □

Now, we are ready to conclude that given some state $S = (\bar{E}, \underline{E}, T)$ which can be reached by some interview from the initial state S_0, $\bar{E}$ is the tightest maximum precedence relations set. This is required by the definitions of the objective functions of the DCVP(-gen) in (3) and (4).

Collorary 1 *Assume that a series of queries and answers brought us from the initial state S_0 to state $S = (\bar{E}, \underline{E}, T)$ in the formulated dynamic program. Then, the set $\bar{E}$ is the tightest possible*

maximum precedence relations set given $\bar{E}_0$, $\underline{E}_0$ and the available answers of the oracle. Likewise, $tc(\underline{E})$ is the largest minimum precedence relations set.

Proof. The proof follows from the proof of Proposition 3. Since by definition, $\bar{E}$ contains all true precedence relations of E^*, and $\bar{E} = tc(\underline{E}) \cup \mathcal{U}(S)$, $tc(\underline{E})$ is the largest minimum precedence relations set because it includes all possible precedence relations whose status is not uncertain. Now, again because $\bar{E} = tc(\underline{E}) \cup \mathcal{U}(S)$, set $\bar{E}$ is the tightest possible maximum precedence relation set. □

Collorary 2 *In the constructed DP, the contribution is*
$C(S,(i,j),\Omega_{ij}) = \begin{cases} 1 & \text{if } \Omega_{ij} = 1 \\ 0 & \text{if } \Omega_{ij} = 0. \end{cases} \quad \forall S, (i,j) \in X(S).$

Proof. According to the definition, the contribution is the number of possible precedence relations removed after the oracle's answer. Given Collorary 1, the contribution equals the change in the cardinality of $\bar{E}$. Recall that by Proposition 2, $tc(\bar{E}) = \bar{E}$ and $tc(\bar{E} \setminus (i,j)) = \bar{E} \setminus (i,j)$ for a valid query (i,j) in the formualted dynamic program. According to the definition of the transition between the states, if $\Omega_{ij} = 1$, then $C(S,(i,j),1) = |tc(\bar{E})| - |tc(\bar{E} \setminus (i,j))| = |\bar{E}| - |\bar{E} \setminus (i,j)| = 1$. If $\Omega_{ij} = 0$, then $C(S,(i,j),0) = |tc(\bar{E})| - |tc(\bar{E})| = 0$. □

Based on Collorary 2, the evaluation of the contribution function is trivial. This reduces the computational burden of the DCVP-gen. Furthermore, this shows that the immediate value of a query depends only on the probability p_{ij} of the respective task pair. The long-term consequences of the query, however, also depend on the structure of $\bar{E}$. Imagine $\bar{E}$ as the set of arcs in a graph. With targeted queries, we may 'reach' the areas of this graph with low τ_{ij} (easy-to-answer queries) and large p_{ij} (high probabilities of a beneficial answer), that is, AC in Figure 3. If both sets of parameters are constant, the DCVP-gen turns trivial as we show in Proposition 5 below.

PROPOSITION 4. *The DP stated in Section 3.4 indeed optimizes Equation* (4).

Proof. For each policy π and each state S, let $X^\pi(S)$ denote the decision prescribed by policy π in state S.

In the DCVP-gen as stated in Section 3.1, the queries are only limited to the transitive reduction of the current maximum precedence relations set $tr(\bar{E})$, but in the dynamic program of Section 3.4 we additionally exclude queries to $tc(\underline{E})$. Observe that the excluded actions are dominated in the sense that there exists at least one optimal policy for the DCVP-gen in which no such queries are asked.

We denote the random variable of the state which is based on the policy π after q queries as $S_{\pi(q)}$ with maximum precedence relations set $\bar{E}_{\pi(q)}$. If $X^\pi(S_{\pi(q)}) = \emptyset$, no query can be asked and the state remains unchanged: $S_{\pi(q+1)} = S_{\pi(q)}$. Finally, for Equation (4) holds

$$\begin{aligned}\max_\pi \mathbb{E}^\pi \left[|\bar{E}_0| - |\bar{E}_\pi|\right] &= \max_\pi \mathbb{E}^\pi \left[\sum_{q=0}^{\infty} |\bar{E}_{\pi(q)}| - |\bar{E}_{\pi(q+1)}|\right] \\ &= \max_\pi \mathbb{E}^\pi \left[\sum_{q=0}^{\infty} C(S_{\pi(q)}, X^\pi(S_{\pi(q)}), \Omega_{X^\pi(S_{\pi(q)})})\right] = B(S_0) \end{aligned} \tag{11}$$

With $\bar{E}_{\pi(0)} = \bar{E}_0$ and $\lim_{q\to\infty} \bar{E}_{\pi(q)} = \bar{E}_\pi$. For a proof of the last equality see Powell (2011). □

The following lemma provides insights into the action set, which we will use, for instance, in stating the computational complexity of a special case of the DCVP-gen in Proposition 5. More precisely, it states that we can ask feasible interview questions in an arbitrary order.

LEMMA 1. *Given state $S = (\bar{E}, \underline{E}, T)$ with $|X(S)| \geq 2$ for $e, e' \in X(S), e \neq e'$, if $T \geq \tau_e + \tau_{e'}$, the following statements are true:*

a) $e' \in X(\bar{S}_e)$

b) $e' \in X(\underline{S}_e)$

Proof. Recall that $\bar{S}_e$ and $\underline{S}_e$ are the states to which we transit from S after having stated a query to task pair e if the answer of the oracle is $\omega_e = 1$ and $\omega_e = 0$, respectively.

From $e' \in X(S)$ it follows that $e' \in tr(\bar{E})$ and $e' \notin tc(\underline{E})$. Let denote $e' = (i', j')$.

a): Let $e' \notin X(\bar{S}_e) = \left\{(i,j) \in V \times V \mid (i,j) \in tr(\bar{E} \setminus \{e\}) \setminus tc(\underline{E}), \tau_{ij} \leq T - \tau_e\right\}$. Since $e' \in X(S)$ and $T \geq \tau_e + \tau_{e'}$, then $e' \notin tc(\underline{E})$, which implies that $e' \notin tr(\bar{E} \setminus \{e\})$. In other words, $\exists k \in V : \{(i',k),(k,j')\} \subseteq \bar{E} \setminus \{e\}$. However, since $e' \in tr(\bar{E})$, it holds that $\{(i',k),(k,j')\} \nsubseteq \bar{E}$. Proven by contradiction.

b): Let $e' \notin X(\underline{S}_e) = \left\{(i,j) \in V \times V | (i,j) \in tr(\bar{E}) \setminus tc(\underline{E} \cup \{e\}), \tau_{ij} \leq T - \tau_e\right\}$. Since $e' \in X(S)$ and $T \geq \tau_e + \tau_{e'}$, it follows that $e' \in tc(\underline{E} \cup \{e\})$. In other words, $\exists k \in V : (i',k) \in \underline{E} \cup \{e\}, (k,j') \in \underline{E} \cup \{e\}$. Recall that $(\underline{E} \cup \{e\}) \subseteq \bar{E}$ (cf. the definition of the state variables and actions in (6) and (7)), meaning that $(i',k) \in \bar{E}$ and $(k,j') \in \bar{E}$, but then $e' = (i',j') \notin tr(\bar{E}) \implies e' \notin X(S)$. Proven by contradiction. □

PROPOSITION 5. *For DCVP-gen with constant parameters $p_{ij} = p$ and $\tau_{ij} = \tau$, any policy (as defined by (7)) is optimal.*

Proof. Observe that the expected value of the state-action pair (S, e) is $Q(S,e) = p_e \cdot (1 + B(\bar{S}_e)) + (1 - p_e) \cdot B(\underline{S}_e)$ for all states $S \mid X(S) \neq \emptyset$ and any task pair $e \in X(S)$ (cf. Bellman equation (9)). For an arbitrary $S = (\bar{E}, \underline{E}, T)$, we obtain:

$$Q(S,e) = p(1 + B(\bar{E} \setminus e, \underline{E}, T - \tau)) + (1-p)B(\bar{E}, \underline{E} \cup e, T - \tau) \tag{12}$$

Below, we will use a proof by induction to show that for any state $S \mid X(S) \neq \emptyset$, $Q(S,e)$ has the same value irrespective of the selected query e, that is:

$$Q(S,e) = Q(S,e') \qquad \forall e,e' \in X(S) \tag{13}$$

Basis: The proposition obviously holds if either of the following conditions is true:

- $|X(S)| \leq 1$ or
- $T < 2\tau$.

Induction step: Let the proposition hold for the states with $T \leq (T' - \tau)$, where $(T' - \tau)$ is some non-negative number. As we will show below, then it holds for the states with $T \leq T'$.

Consider $S = (\bar{E}, \underline{E}, T')$. The case of $|X(S)| \leq 1$ is trivial. Hence, assume that $|X(S)| \geq 2$.

Take any $e, e' \in X(S)$ and write out $Q(S,e)$ explicitly for two stages:

$$\begin{aligned} Q(S,e) =& p(1 + \max_{e^*} \left[p(1 + B(\bar{E} \setminus e \setminus e^*, \underline{E}, T' - 2\tau)) + (1-p)B(\bar{E} \setminus e, \underline{E} \cup e^*, T' - 2\tau)\right] + \\ & (1-p) \max_{e^*} \left[p(1 + B(\bar{E} \setminus e^*, \underline{E} \cup e, T' - 2\tau)) + (1-p)B(\bar{E}, \underline{E} \cup e \cup e^*, T' - 2\tau)\right] \end{aligned} \tag{14}$$

From Lemma 1, we know that $e' \in X(\bar{S}_e)$ and $e' \in X(\underline{S}_e)$. So, by the induction hypothesis, we can rewrite Expression (14) as follows:

$$\begin{aligned} Q(S,e) =& p(1 + \left[p(1 + B(\bar{E} \setminus e \setminus e', \underline{E}, T' - 2\tau)) + (1-p)B(\bar{E} \setminus e, \underline{E} \cup e', T' - 2\tau)\right] + \\ & (1-p) \left[p(1 + B(\bar{E} \setminus e', \underline{E} \cup e, T' - 2\tau)) + (1-p)B(\bar{E}, \underline{E} \cup e \cup e', T' - 2\tau)\right] \end{aligned} \tag{15}$$

By a simple rearrangement, we receive:

$$\begin{aligned} Q(S,e) =& p(1 + \left[p(1 + B(\bar{E} \setminus e' \setminus e, \underline{E}, T' - 2\tau)) + (1-p)B(\bar{E} \setminus e', \underline{E} \cup e, T' - 2\tau)\right] + \\ & (1-p) \left[p(1 + B(\bar{E} \setminus e, \underline{E} \cup e', T' - 2\tau)) + (1-p)B(\bar{E}, \underline{E} \cup e' \cup e, T' - 2\tau)\right] \end{aligned} \tag{16}$$

The second part of Expression (16) resembles a situation, where we would select query e' in state S first and state query e afterwards. Since $e \in X(\bar{S}_{e'})$ and $e \in X(\underline{S}_{e'})$ by Lemma 1 and given the induction hypothesis, we can write:

$$\begin{aligned} Q(S,e) =& p(1 + \max_{e^*} \left[p(1 + B(\bar{E} \setminus e' \setminus e^*, \underline{E}, T' - 2\tau)) + (1-p)B(\bar{E} \setminus e', \underline{E} \cup e^*, T' - 2\tau)\right] + \\ & (1-p) \max_{e^*} \left[p(1 + B(\bar{E} \setminus e^*, \underline{E} \cup e', T' - 2\tau)) + (1-p)B(\bar{E}, \underline{E} \cup e' \cup e^*, T' - 2\tau)\right] = \\ & = Q(S,e') \end{aligned} \tag{17}$$

□

According to Proposition 5, we obtain an optimal policy by selecting an arbitrary action in each state of the formulated dynamic program if the queries regarding precedence relations require the same time, and are successful with equal probability. Another interpretation is the following: If the interviewer is unable to quantify the benefits of the queries (the probability to remove a precedence relation), as well as the consumption of the resource (the time to answer a query) and therefore assumes these to be constant, the interviewer can not optimize the interview beyond stating queries to any task pair from $tr(\bar{E}) \setminus tc(\underline{E})$ for the current state S. Therefore, it is necessary for the interviewer to estimate the parameters of the DCVP(-gen) prior to the interview.

PROPOSITION 6. *The DCVP-gen, whose parameters are rational numbers, is NP-hard in the strong sense.*

Proof. Observe that the concept of 'language' used in the definition of P- and NP-problem instances, accepts only a finite string of symbols and cannot work, for instance, with real numbers. Therefore, we formulate the proof for rational numbers. The knapsack problem with weights and profits that are rational numbers is NP-hard in the strong sense (Wojtczak 2018). Any such knapsack instance can be reduced to the DCVP-gen in polynomial time, as we show below. Hence, the DCVP-gen is NP-hard in the strong sense as well.

Consider an arbitrary but fixed instance of the knapsack problem with capacity T and K items. The value of the items $k \in \{1, \dots, K\}$ is denoted as $v_k \in \mathbb{R}_0^+$ and the weight as $w_k \in \mathbb{R}_0^+$, $\sum_{k \in V} w_k > T$. W.l.o.g. the values are scaled such that $0 \leq v_k \leq 1$. The objective is to select a subset of items $\mathcal{V} \subseteq \{1, \dots, K\}$ with the highest total value $\sum_{k \in \mathcal{V}} v_k$ such that the capacity of the knapsack is not violated, i.e. $\sum_{k \in \mathcal{V}} w_k \leq T$.

Now let there be a DCVP-gen with the set of tasks $V := \{1, \dots, K+1\}$ and initial maximum precedence relations set $\bar{E}_0 = \{(i,j) \in V \times V \mid i < j\}$. Observe that $\bar{E}_0$ is transitive and acyclic. The set of precedence relations known for sure is $\underline{E} = \emptyset$. The probability p_{ij} is given by $p_{ij} = \begin{cases} v_i & ,j = i+1 \\ p & ,else \end{cases}$, where $p \in (0,1)$ is an arbitrary number, and the cost $\tau_{ij} = \begin{cases} w_i & ,j = i+1 \\ T+1 & ,else \end{cases}$. Therefore each precedence relation in $tr(\bar{E}_0)$ corresponds to one of the K items. The target precedence relations set, which the oracle needs for her answers, equals $E^* := \emptyset$.

Given Proposition 4, without loss of generality, we assume that the DCVP-gen is stated as the dynamic program from Section 3.4. Observe that a feasible sequence of interview queries $\mathcal{I} = ((i_1, j_1), (i_2, j_2), \dots)$ that exhausts the time budget in the formulated DCVP-gen requires $\sum_{(i,j) \in \mathcal{I}} \tau_{ij}$ time and has the expected value of $\sum_{(i,j) \in \mathcal{I}} p_{ij}$. The latter can be found by applying the Bellman equation (9) to the constructed state graph in a backward induction starting with the corresponding absorbing state.

Any feasible solution $\mathcal{V} \subseteq \{1, \ldots, K\}$ of the knapsack problem which exhausts the knapsack's capacity T corresponds to a sequence of interview questions and the respective answers of the oracle (i.e., to a path from the initial state S_0 to some absorbing state) in the formulated DCVP-gen with the same expected value. Indeed, since all the knapsack items correspond to the task pairs in $tr(\bar{E}_0)$, the respective interview queries can be stated in an arbitrary order according to Lemma 1.

And the other way around, any feasible path from the initial state S_0 to some absorbing state in the state graph of the DCVP-gen (i.e. a feasible sequence of not superfluous interview questions and answers of the oracle) corresponds to a feasible solution of the knapsack problem that exhausts the knapsack's capacity T. The expected value of the path and the objective value of the knapsack solution are the same. Finally, note that as we only have non-negative values v, if the knapsack has an optimal solution, then it has an optimal solution that exhausts the knapsack's capacity. □

4. Approximate dynamic programming

In this section, we state how we solve the DCVP-gen via approximate dynamic programming. For this we use a least squares temporal difference (LSTD) approach (see, e.g., Powell (2011) for an introduction into this technique). The aim of LSTD is to fit a value function approximation (VFA)

$$\tilde{B}(S) = \sum_{k=1}^{K} \theta_k \phi_k(S) \tag{18}$$

with features ϕ_k and regression parameters θ_k.

A careful selection of a small number of meaningful, problem-specific features is crucial to a satisfactory execution of this approach. Observe attractive queries (i, j) with high probability p_{ij} and low time τ_{ij} are impossible to ask for if $(i, j) \notin tr(\bar{E})$, i.e. if (i, j) is also an indirect precedence relation caused by some other precedence relations and these potential dependencies must be clarified first. However, as soon as all promising queries are 'accessible', the remaining DCVP-gen instance can be interpreted as a knapsack problem (cf. our analysis in Proposition 6 of Section 3.5 for intuition). Based on these insights, we selected the following parameters for the VFA in Expression 18: $K = 5$ and $\phi_1(S) = \sum_{(i,j)\in X(S)} \frac{p_{ij}}{\tau_{ij}}$, $\phi_2(S) = \sqrt{T} \cdot \sum_{(i,j)\in X(S)} \frac{p_{ij}}{\tau_{ij}}$, $\phi_3(S) = T \cdot \sum_{(i,j)\in X(S)} \frac{p_{ij}}{\tau_{ij}}$, $\phi_4 = T$, $\phi_5 = \sqrt{T}$. The VFA takes into account the remaining duration of the interview T, the feasible set of questions $X(S)$, and attractiveness indices $\frac{p_{ij}}{\tau_{ij}}$. The latter are known to perform well in knapsack problems. In addition, their inclusion allows for rewarding a larger cardinality of $X(S)$. We do not add a constant feature $\phi_k = 1$ for two reasons: First, the interviewer's decisions (the solutions in Step 10 of the Algorithm 1 stated below) are invariant to a constant shift. Second, this poses additional technical challenges in the solution algorithm: The matrix A in Step 14 Algorithm 1 would not be invertible for $\lambda = 1$ (artificial discount factor explained below). We

performed intensive pretests, which explored alternative features, including more complex ones, as well as experimented with different numbers of features. Our pretests show that the proposed architecture of the VFA is a simple and robust one among well-performing alternative architectures.

Algorithm 1 below provides a general overview of the LSTD algorithm. Step 1 initializes the vector of regression parameters $\theta = (\theta_1, \cdots, \theta_K)^T$. These regression parameters are updated M times (Step 4 to Step 17). Step 2 and Step 3 initialize a matrix and a vector of zeros. These are needed for the sums in Step 11 and Step 14. Step 5 initializes the state as the starting state S_0 while Step 6 initializes the counter for the while loop (Step 8 to Step 15). Step 7 evaluates the K features for the initial state S_0. This column vector is needed in the first iteration of the while loop. The while loop in Step 8 iterates as long as the interviewer has a feasible question. Step 9 increments the loop counter. Step 10 finds the best question to ask based on the current VFA which depends on the regression parameters θ_k. Step 11 adds up the vector b for Step 16. Based on the optimal decision in Step 11 and the probability p_{i^*,j^*}, Step 12 transits the system to the new state S. The features are evaluated for the new state in Step 13. The resulting column vector is used to build up the matrix A in Step 14. After the while loop ends (Step 15), the regression parameters are updated using A and b.

Some perceptivity as to why the algorithm works: We can restate the Bellman equation (9) in the vector form for the optimal policy $X^\pi(\cdot)$

$$B(s) = p_{X^\pi(s)} C(s, X^\pi(s), 1) + \mathbb{E}_{X^\pi(s)}[B(s')], \tag{19}$$

where $B(\cdot)$ is the vector based value function and s' the vector of downstream states resulting from the vector s and the optimal decisions $X^\pi(s)$ as well as the stochastic outcome. Rearranging yields

$$0 = p_{X^\pi(s)} C(s, X^\pi(s), 1) + \mathbb{E}_{X^\pi(s)}[B(s')] - B(s). \tag{20}$$

Since we cannot enumerate the whole state space in appropriate time, we replace the vectors of states s and s' with the much smaller vectors $\hat{s}$ and $\hat{s}'$ which correspond to the visited states in Algorithm 1. As not all downstream states of the current state are visited, we cannot compute the expectation in (20). We can only observe a single outcome. This results in:

$$\delta(\hat{s}) = p_{X^\pi(\hat{s})} C(\hat{s}, X^\pi(\hat{s}), 1) + B(\hat{s}') - B(\hat{s}), \tag{21}$$

where $\delta(\hat{s})$ is the error based on replacing the expectation with a single observation. This error is called the temporal difference (TD) and this should be close to zero. Finally, we replace the value function B by the VFA $\tilde{B}$ and expand it as in Equation (18):

$$\hat{\delta}(\hat{s}) = p_{X^\pi(\hat{s})} C(\hat{s}, X^\pi(\hat{s}), 1) + \theta^T \cdot \phi^l - \theta^T \cdot \phi^{l-1} \tag{22}$$

Algorithm 1 LSTD

1: Initialize K dimensional column vector of regression parameters $\theta := 0_K$
2: Initialize $K \times K$ matrix $A := 0_{KK}$
3: Initialize K dimensional column vector $b := 0_K$
4: **for** $m = 1$ to M **do**
5: Initialize $S := S_0$
6: Initialize counter $l := 0$
7: $\Phi^0 := \Phi(S)$
8: **while** $X(S) \neq \emptyset$ **do**
9: $l := l + 1$
10: $(i^*, j^*) := \underset{(i,j) \in X(S)}{\arg\max}\, p_{ij} \cdot (C(S, (i,j), 1) + \tilde{B}(\bar{S}_{ij})) + (1 - p_{ij}) \cdot \tilde{B}(\underline{S}_{ij})$
11: $b := b + \Phi^{l-1} \cdot p_{i^*,j^*} C(S, (i^*, j^*), 1)$
12: Stochastic state transition
13: $\Phi^l := \Phi(S)$
14: $A := A + \Phi^{l-1} \cdot (\Phi^{l-1} - \lambda \Phi^l)^T$
15: **end while**
16: $\theta := A^{-1} b$
17: **end for**

The temporal difference $\hat{\delta}(\hat{s})$ is the error based on replacing the value function by the VFA and the single observation of the downstream state. The aim of LSTD, especially Step 16 of Algorithm 1, is to find regression parameters θ which minimize $\hat{\delta}(\hat{s})$ in the least squares sense. More precisely, Step 16 of Algorithm 1 is the solution to the Gaussian normal equations which minimizes (22). For faster convergence, we use an artificial discount factor λ in this regression.

The purpose of Algorithm 1 is to compute VFAs. The induced policy is

$$X^{VFA}(S) = \underset{(i,j) \in X(S)}{\arg\max}\, p_{ij} \cdot (C(S, (i,j), 1) + \tilde{B}(\bar{S}_{ij})) + (1 - p_{ij}) \cdot \tilde{B}(\underline{S}_{ij}) \tag{23}$$

5. Computational experiments

In this section, we perform computational experiments on benchmark data sets and examine a case study of data collection and validation from the final assembly of automobiles.

We start with the description of the benchmark data sets, of the evaluated policies as well as of further general aspects of the experimental design in Section 5.1. Afterward, we first explore the potential of partial data validation based on approximation formulas (Section 5.2). Then, the core of the computational study starts and numerically quantifies the benefits of interview optimization in

Section 5.3. Section 5.4 focuses solution quality and benchmarks the designed approximate dynamic programming approach. Finally, Section 5.5 illustrates the benefits of a structured ('optimized') interview in the case study.

5.1. Benchmark data sets and algorithms evaluated

As discussed in the introduction, data collection on precedence relations is relevant to many areas of business. This includes product design, assembly and manufacturing, project scheduling, construction, as well as transportation and logistics. We examined these fields within the literature in search of established benchmark data sets structured around the characteristics of the precedence relations set, since the precedence relations set is in the focus of the DCVP-gen. As a result, we selected the benchmark data sets of Otto et al. (2013), which are well-known in the assembly line literature, and the benchmark data sets of Kolisch and Sprecher (1996) and Kolisch et al. (1999), which are prominent in the resource-constrained project scheduling literature. We refer to the first group of data sets as *ALPLib, Assembly Line Problem Library*, and to the second group of data sets as *PSPLib, Project Scheduling Problem Library*. In the main part of our computational experiments, we use the data sets of ALPLib. These consist of instances with 50 tasks (525 instances in total) and the data sets of PSPLib consisting of instances with 30 tasks (480 instances in total). Therefore, if not otherwise stated, when we speak about ALPLib, we imply instances with 50 tasks from these data sets, and when we speak about PSPLib, we imply instances with 30 tasks from these data sets. The effect of the instance's size (i.e. the number of tasks) is examined systematically as part of Section 5.4.

The instances of ALPLib are structured around the *order strength* $OS \in [0,1]$, where 0 implies that no precedence relation exists and 1 means that all possible precedence relations exist. More formally, OS is the share of task pairs with a precedence relation in the total number of task pairs: $OS(E) = \frac{|tc(E)|}{\frac{n\cdot(n-1)}{2}}$, where $n = |V|$ is the number of tasks. The ALPLib consists of 225 instances with order strength 0.2, 225 instances with order strength 0.6, and 75 instances with order strength 0.9.

The 480 instances of PSPLib contain instances with rather low order strength of $[0.25, 0.65]$ with an average of 0.45. Thereby each task has between 1 and 3 immediate successors and 1 and 3 immediate predecessors. To report the results, we form two groups: 305 instances with order strength < 0.5 and 175 instances with order strength ≥ 0.5.

We treat each instance I in ALPLib and PSPLib as an unknown target precedence relations set E_I^* and ignore possible additional characteristics of the instances other than precedence relations. We assume that, initially, we do not know any precedence relations for sure, that is $\underline{E}_{I,0} := \emptyset$. To generate the initial maximum precedence relations set $\bar{E}_{I,0}$ for an instance with n tasks, we use the transitive closure induced by the sequence $< 1, 2, \ldots, n >$, which corresponds to the numbering of the tasks, and is known to be feasible for all instances.

Based on our intensive talks with practitioners, we decided on the following rather conservative approach in generating parameters τ_{ij} and p_{ij}. We assign high probabilities of $p_{ij} := 0.9$ to the task pairs in $tc(E_I^*)$ and low probabilities of $p_{ij} := 0.1$ to the remaining task pairs. Next, we swap parameter values for a certain number (e.g. $\xi \cdot n$) of task pairs randomly drawn with replacement to reflect uncertainty in the baseline data. We call ξ the *noise index* and set its default value to 0.2. In more detail, the generation process for instance I is given by Algorithm 2:

Algorithm 2 Generation of probabilities p_{ij}

1: Set probabilities $p_{ij} := 0.9 \cdot 1_{(ij) \notin tc(E_I^*)} + 0.1 \cdot 1_{(ij) \in tc(E_I^*)}$.
2: Sample (with replacement) $\xi \cdot n$ task pairs (i, j) and set $p_{ij} := 1 - p_{ij}$.

We simply set $\tau_{ij} := 1$, i.e. the interviewer asks up to T_0 questions, since no comprehensive data on the characteristics of the real-world distributions of τ_{ij} is available to us yet.

The stated data generation approach for τ_{ij} and p_{ij} is conservative since it relates to the current state of data at companies. We expect that with further progress in digitization, and as a result of increased scientific support regarding the problem of data validation, companies will improve their ability to precisely estimate these values. Thus, in the future, values for τ_{ij} and p_{ij} will become more varied at companies. Observe that, given insights from Proposition 5, more variability in parameters τ_{ij} and p_{ij} should make the positive effect of interview optimization even more prominent.

In our computational experiments, we use a straightforward interview policy (*Naive*) as the baseline and compare it with two structured interview policies (*Myopic* and *LSTD*). We call the latter two policies structured, since the selection of the next interview question is not arbitrary, but follows some not purely random algorithm.

Naive: Naive chooses the next interview query randomly from the feasible set $X(S)$. In contrast with the other methods, *Naive* ignores information on p_{ij} and τ_{ij} beyond a simple check for the remaining time budget.

Myopic: This policy chooses a query with the highest short-term value without anticipating future queries. The short-term value coefficient is motivated by the similarity between the DCVP-gen and the knapsack problem (cf. Proposition 6). Specifically,

$$X^{Myopic}(S) := \underset{(i,j) \in X(S)}{\arg\max} \frac{p_{ij}}{\tau_{ij}}. \tag{24}$$

LSTD: LSTD is the policy proposed in Section 4. We update the regression parameters $M := 50$ times with $\lambda := 0.9$.

In all algorithms, we break ties randomly. For each examined policy, we report the 'average number of removed precedence relations': we compute $(|\bar{E}_{I,0}| - |\bar{E}_I^{\pi}|)$ for each instance I and average this number over the respective instances.

We conducted our experiments on a computer with an AMD Ryzen 5 2400G CPU (4×3.6 Ghz), 16 GB RAM and Windows 10 operating system. The interview policies were programmed with Matlab R2020a.

5.2. Exploring the potential of partial data validation: Analytical approximation

According to first-hand obtained communication, practitioners do not perceive partial validation of the data on precedence relations as an investment alternative worthy of pursuit given how expensive and limited expert time is. In our opinion, managers may underestimate:

- The amount of additional information – indirect precedence relations – that can be derived from the expert's answers.
- The effect that the removal of even a small number of unnecessary precedence relations would have on the optimal solution of the baseline planning problem.

As for the latter, we provide an anchor for the reader and relate the number of removed precedence relations to the number of feasible sequences of tasks – which is a more telling number for managers in logistics, manufacturing, or project scheduling. Even small changes in $|\bar{E}_{I,0}| - |\bar{E}_I^{\pi}|$ usually lead to a *surge* in the number of feasible task sequences, and thus, in the number of feasible solutions in the baseline optimization problem. Generally, the larger the number of tasks n is, the larger the effect on the number of feasible sequences. For example, for instances of $n = 20$ from ALPLib with $T_0 = 40$ and $OS(\bar{E}_{I,0}) = 1.0$, the removal of just 28.24 precedence relations (the result of the receding horizon approach *Receding* introduced later on in the text, see Table 1) leads to 74,400 additional feasible sequences on average. Whereas if we instead remove 31.37 precedence relations (the result of the *LSTD* approach), the number of additional feasible sequences increases by 90% to 141,300.

As for the first, we state an analytical approximation for the required number of queries, which takes into account the derivation of indirect precedence relations from the answers of the expert. This formula works surprisingly well in our experiments. Observe that in practice, the order strength of the target graph is unknown. As Otto and Otto (2014) showed, no queries in the DCVP-gen can be formulated regarding task pairs that have an indirect precedence relation in the target graph, i.e. $(i.j) \notin E_I^* \setminus tr(E_I^*)$. Thus, the number of nontrivial queries to derive the target precedence relations set with certainty cannot exceed:

$$|tr(E_I^*)| + \big(OS(\bar{E}_{I,0}) - OS(E_I^*)\big) \cdot \frac{n(n-1)}{2}, \tag{25}$$

which is the number of task pairs with a direct precedence relation on the target graph ($|tr(E_I^*)|$) as well as the number of independent task pairs, whose status is initially unknown, $\left(\left(OS(\bar{E}_{I,0}) - OS(E_I^*)\right)\right) \cdot \frac{n(n-1)}{2}$). The first summand is usually proportional to n in 'real-world' graphs (it is between n and $3n$ in PSPLib by construction and does not exceed $2.36 \cdot n$ in ALPLib). For example, if $OS(\bar{E}_{I,0}) = 0.3$, $OS(E_I^*) = 0.2$, $n = 50$ and assuming $|tr(E_I^*)| = 1.5n$, about $1.5 \cdot 50 + 0.1 \cdot \frac{50 \cdot 49}{2} \approx 197$ queries are required to completely validate the data set. The estimate is supported by computational results in Figure 5a. This number is much lower than a naive estimate of $\frac{n \cdot (n-1)}{2} = 1225$ queries, which is to state queries to each pair of tasks. It is also much lower than an alternative naive estimate of $|\bar{E}_{I,0}| = 0.3 \cdot \frac{n \cdot (n-1)}{2} \approx 368$.

To provide guidance for the following computational experiments, we analytically estimate the probability that one of *Naive's* queries removes a precedence relation from $\bar{E}_{I,0}$.

A trivial estimate is given by the share of existing precedence relations in the maximum precedence relations set:

$$p_{trivial} = \frac{\left(OS(\bar{E}_{I,0}) - OS(E_I^*)\right) \cdot \frac{n(n-1)}{2}}{OS(\bar{E}_{I,0}) \cdot \frac{n(n-1)}{2}} = 1 - \frac{OS(E_I^*)}{OS(\bar{E}_{I,0})}. \tag{26}$$

However, this bound neglects that *Naive*, as all our approaches, only asks about direct precedence relations (see Assumption 6). Next, we improve the estimate by accounting for this, which also nicely illustrates the value of Assumption 6:

$$p_{Ass6} = \frac{\left(OS(\bar{E}_{I,0}) - OS(E_I^*)\right) \cdot \frac{n(n-1)}{2}}{|tr(E_I^*)| + \left(OS(\bar{E}_{I,0}) - OS(E_I^*)\right) \cdot \frac{n(n-1)}{2}} \tag{27}$$

Obviously, $p_{trivial} < p_{Ass6}$ since $|tr(E_I^*)| < OS(E_I^*) \cdot \frac{n(n-1)}{2}$. For example, if $OS(\bar{E}_{I,0}) = 1.0$, $OS(E_I^*) = 0.2$, $n = 50$, $T_0 = 150$ and assuming $|tr(E_I^*)| = 1.5n$, then *Naive* will remove approximately 139 precedence relations, or 90% to 95% of T_0. If $OS(E_I^*) = 0.9$, instead, *Naive* will remove precedence relations in just about 66% of queries.

Therefore, in the following experiments, we expect that even a limited number of queries will remove a considerable number of precedence relations, which will result in a large number of additional task sequences being recognized as feasible. Since *Naive* already performs very well in certain settings, we expect the LSTD interview policy to have the largest advantage over *Naive*, if $OS(E_I^*)$ is high or $(OS(\bar{E}_{I,0}) - OS(E_I^*))$ is low.

5.3. Exploring benefits of partial data validation: Computational experiments

In a series of computational studies in this section, we attempt to quantify the benefits of structured interviews in different settings. More precisely, we vary the proximity of the initial precedence relations sets $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ to the target set in Section 5.3.1, the time budget T_0 in Section 5.3.2, as well as the quality of the baseline data on p_{ij} in Section 5.3.3.

Vertical dashes in the figures of the following subsections mark the largest improvement between the respective policies. The absolute numbers refer to the difference in the average number of the removed precedence relations, and the percentages translate this number into the percentage points of the respective outperformed policy. For example, in Figure 5b, the largest 'belly' between *Myopic* and *Naive* is observed at $L = 5$ with $OS(\bar{E}_{I,0}) = 0.79$; at this point, *Myopic* removes 41.3 more precedence relations on average and outperforms *Naive* by 46.9%. The largest 'belly' between *LSTD* and *Myopic* in the same figure occurs at $\bar{E}_{I,0} = 0.71$, where *LSTD* removes 10.5 more precedence relations on average and outperforms *Myopic* by 10.3%.

If not otherwise stated, we set $T_0 := 150$ for the ALPLib instances of $n = 50$ tasks and $T_0 := 100$ for PSPLib instances of $n = 30$.

5.3.1. Influence of the initial precedence relations sets. In this section, we investigate how the behaviour of *Naive*, *Myopic* and *LSTD* depends on the quality (i.e. the tightness) of $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$. Recall that $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ directly impact the cardinality of the initial uncertainty set. We simulate two realistic business scenarios:

(a) Set $\bar{E}_{I,0}$ is based on one or several initial feasible sequences of tasks.

This setting is motivated by practices in manufacturing described by Klindworth et al. (2012). Initial feasible sequences of tasks stem from actual past production plans that differ from each other (are re-balanced) because, e.g., of changing demand.

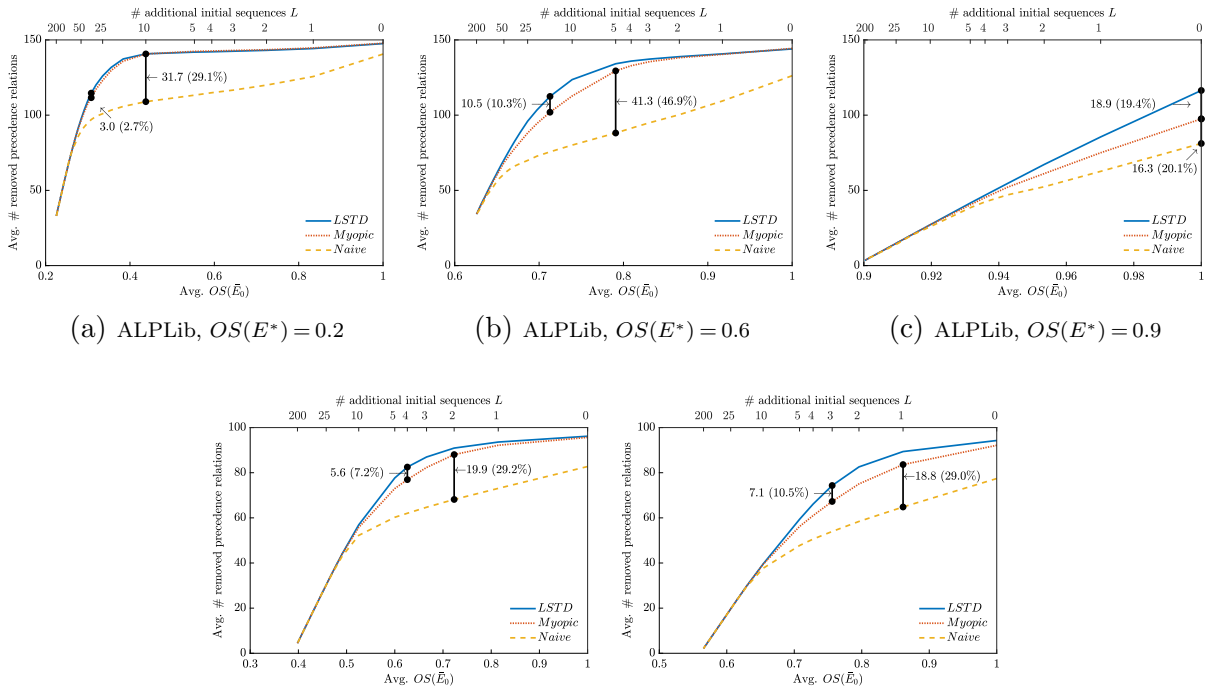
(b) Sets $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ are based on initial limited data entry by experts.

We simulate this setting by conducting a myopic pre-interview (as in the *Myopic* algorithm) with a limited number of queries.

(a) $\bar{E}_0$ based on one or several initial feasible sequences of tasks

We generate the initial sequences of tasks as follows. The first sequence is $< 1, \ldots, n >$. Afterwards for each instance, we continue adding further feasible sequences of tasks one by one and by doing so, we remove more and more precedence relations from the resulting set $\bar{E}_{I,0}$. Each such additional sequence of tasks is generated randomly based on E_I^*. Figure 5 reports the results of the interviews with $\bar{E}_{I,0}$ that resulted from $L \in \{1, 2, 3, 4, 5, 10, \cdots, 195, 200\}$ additional sequences of tasks. Value L is depicted on the *upper* horizontal axis.

For the sake of comparability with further computational studies, we add an additional (*lower*) horizontal axis that depicts the average order strength of $\bar{E}_{I,0}$. More precisely, each point i corresponds to the instances with $L := L_i$ additional sequences of tasks, and the horizontal coordinate of point i equals to the average order strength of the resulting $\bar{E}_{I,0}$, correspondingly. Note that, obviously, for each instance $OS(\bar{E}_{I,0}) \geq OS(E_I^*)$ holds.



(a) ALPLib, $OS(E^*) = 0.2$ (b) ALPLib, $OS(E^*) = 0.6$ (c) ALPLib, $OS(E^*) = 0.9$

(d) PSPLib, $OS(E^*) < 0.5$ (e) PSPLib, $OS(E^*) \geq 0.5$

Figure 5 **The influence of the initial maximum precedence relations set.**
Note. $\bar{E}_{I,0}$ is derived based on one or several initially known feasible sequences of tasks.

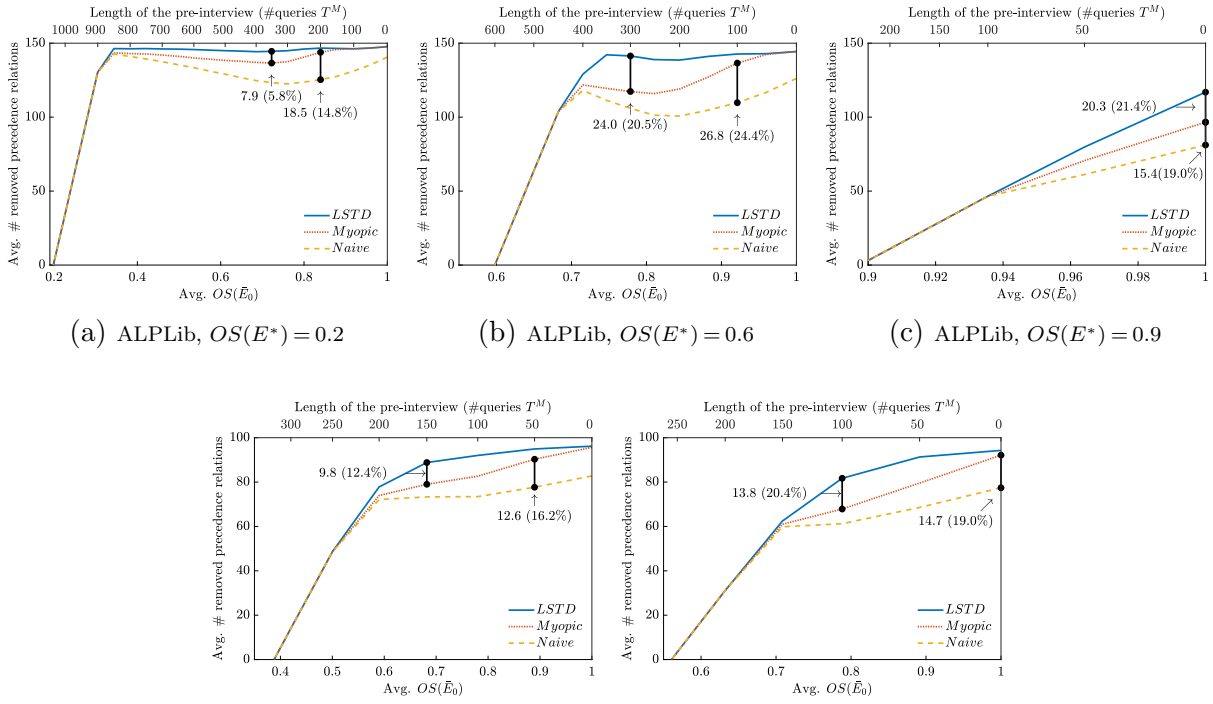
As Figures 5 shows, structured interviews significantly outperform the *Naive* policy if $OS(\bar{E}_{I,0})$ is not too similar to $OS(E_I^*)$. The improvement between *LSTD* and *Naive* reaches 34-54% depending on the setting. Between *LSTD* and *Myopic*, improvements of up to 3%-19% are observed in the examined settings. For example, for ALPLib and $OS(E^*) = 0.6$ in Figure 5b, the difference between *Naive* and *Myopic* is the biggest at $OS(E^*) = 0.79$ (corresponds to $L = 5$), here *Myopic* outperforms *Naive* by 41.3 or 46.9%. At this point, *LSTD* removes 4.7 more precedence relations than *Naive*, so that the total advantage of *LSTD* over *Naive* equals 46.0 (or 52.2%). The improvement of *LSTD* over *Naive* in this figure remains significant, even if only one feasible sequence of tasks is available and equals 17.8 (or 14.1%). For both ALPLib and PSPLib, we observe that if $OS(\bar{E})$ only slightly exceeds $OS(E^*)$, all three approaches perform very similarly. For example, for ALPLib with $OS(E^*) = 0.6$ (Figure 5b) this is the case for $OS < 0.65$ ($L \geq 100$). There, the initial maximum precedence relations set is very close to the target precedence relations set, and the interview questions are sufficient to derive the target precedence relations set almost exactly (cf. Expression 25).

For all the policies, the number of removed precedence relations increases in the order strength of $\bar{E}_{I,0}$. This is as expected, since the share of unnecessary precedence relations in $\bar{E}_{I,0}$ increases.

(b) $\bar{E}_0$ and $\underline{E}_0$ based on initial limited data entry by experts

In this study, we generate $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ from a myopic 'pre-interview'. Specifically for each instance I, based on the first sequence $<1,\ldots,n>$, we perform a myopic pre-interview by revealing information an increasing number of queries (generated as described by the *Myopic* policy) and, as a consequence, by changing $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ incrementally. Figure 6 reports results based on the pre-interviews with $T^M \in \{0, 50, \ldots 1000\}$ queries. The value of T^M is depicted on the *upper* horizontal axis. An additional *lower* horizontal axis shows the resulting average order strength of $\bar{E}_{I,0}$.

The overall pattern revealed by Figure 6 is similar to that of Figure 5 from the previous study. The advantage of the structured interviews over the baseline *Naive* policy is significant if $OS(\bar{E}_{I,0})$ is not too similar to $OS(E^*_I)$ and reaches 18.3-44.0% depending on the setting. The main difference with the previous study is that for the same $OS(\bar{E}_{I,0})$ on the lower horizontal axis, more information is available since $\underline{E}_{I,0} \neq \emptyset$ is usually true. Overall, the difference in the performance of *Myopic* and *LSTD* in case of low and medium $OS(E^*_I)$ got larger. Whereas the observed improvement of *LSTD* over *Myopic* for $OS(E^*_I) = 0.6$ in the previous study equaled 10.3%, it reaches 20.5 % in this study in the same ALPLib instances. Apparently, *LSTD* is much better in exploiting the additional information on the available precedence relations of $\underline{E}_{I,0}$ than *Myopic*. The specific reasons for this behavior remain to be explored in future research.



(a) ALPLib, $OS(E^*) = 0.2$ (b) ALPLib, $OS(E^*) = 0.6$ (c) ALPLib, $OS(E^*) = 0.9$

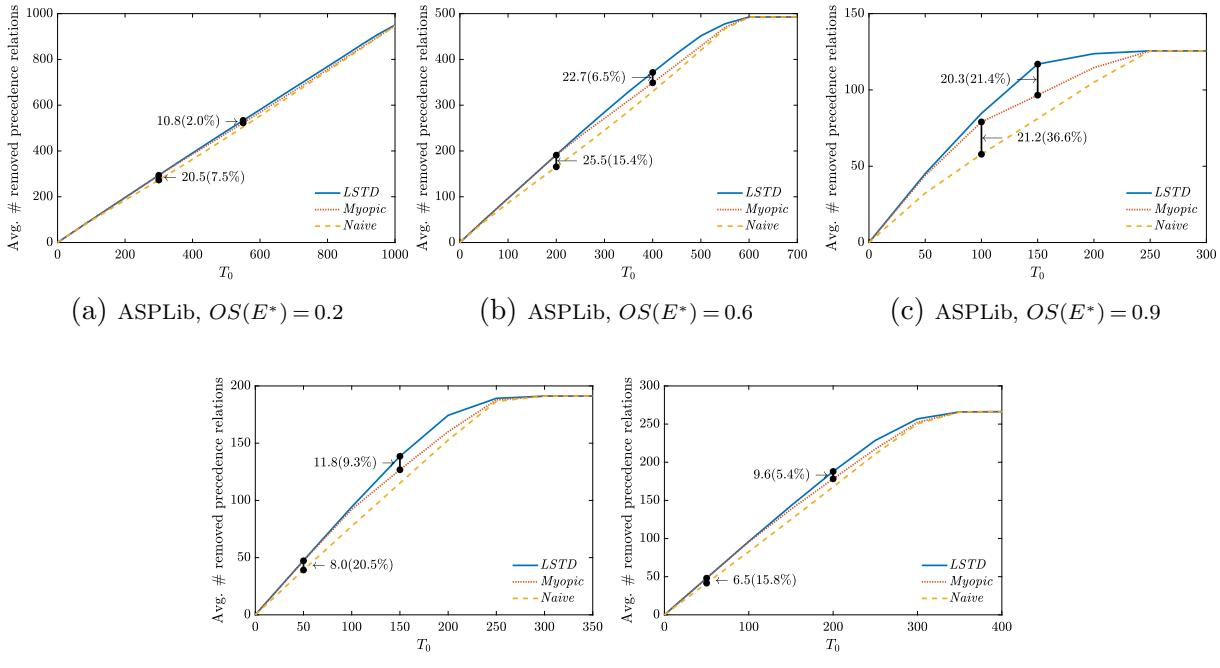
(d) PSPLib, $OS(E^*) < 0.5$ (e) PSPLib, $OS(E^*) \geq 0.5$

Figure 6 **The influence of the initial maximum and minimum precedence relations sets: $\bar{E}_{I,0}$ and $\underline{E}_{I,0}$ are derived based on initial limited data entry by experts**

5.3.2. Influence of the time budget T_0. Figure 7 reports on the performance of the three policies depending on the duration $T_0 \in \{0, 50, \cdots, 950, 1000\}$ of the interview. Observe that the maximum value of the vertical axis is several times larger than in the previous figures.

This experiment confirms significant improvements in structured interviews following the *Myopic* and *LSTD* policies over *Naive*. The largest improvement in *LSTD* over *Naive* is achieved at $T_0 := 350, 250$ and 100 for ALPLib with $OS(E_I^*) = 0.2, 0.6$ and 0.9, respectively, as well as at $T_0 := 100$ for PSPLib instances. The best results of *Myopic* over *Naive* are achieved at $T_0 := 300, 200, 100$ and 50 for the above cited settings, correspondingly.

Interesting is the position of the 'belly' in the relative dynamics of the curves: the three policies have similar absolute performances in the absolute values at low T_0, perform differently at medium T_0 and show similar results once again at big T_0. The flattening-out points are predicted quite well by Expression (25). We also observe that *LSTD* reaches its maximum advantage over *Myopic* if the interview is sufficiently long (the 'belly' lies to the right of that between *Myopic* and *Naive*). Indeed, recall that initially only queries to certain task pairs (task pairs in $tr(\bar{E}_{I,0})$) are possible. *LSTD* strategically attempts to access the regions of $\bar{E}_{I,0}$ with attractive queries. Such a trade off of immediate effects for longer term benefits is only possible if the time budget is large enough. Therefore, at small values of T_0, *LSTD* resembles *Myopic*.

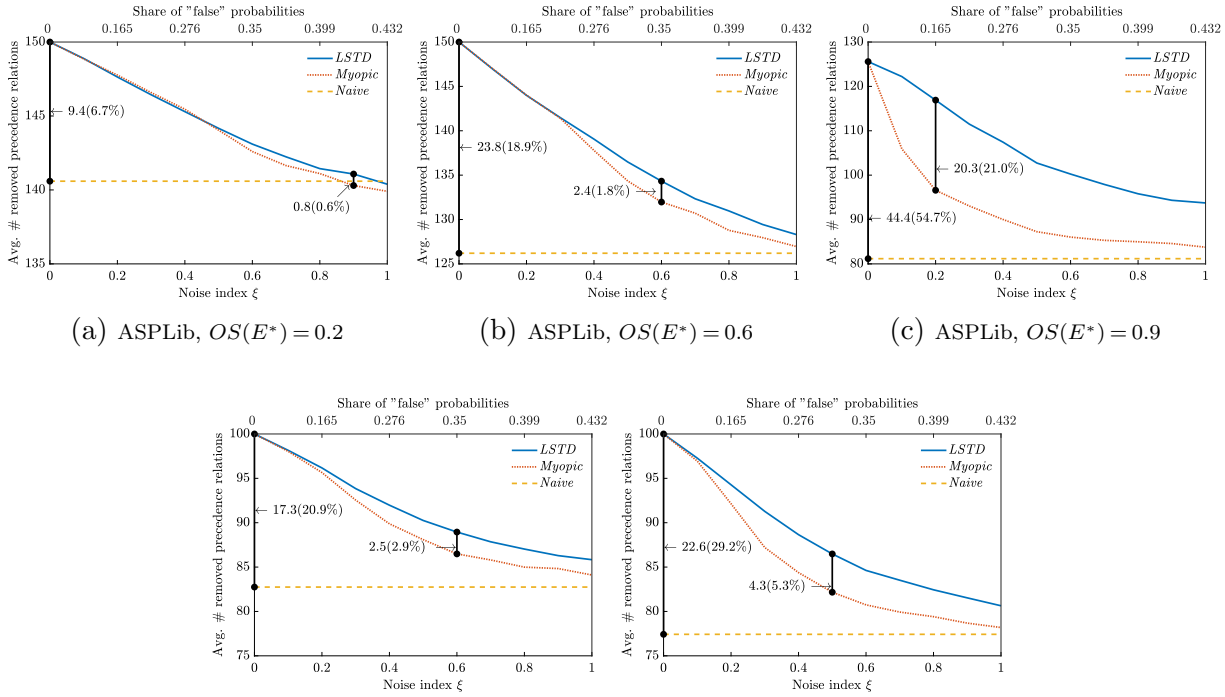


(a) ASPLib, $OS(E^*) = 0.2$ (b) ASPLib, $OS(E^*) = 0.6$ (c) ASPLib, $OS(E^*) = 0.9$

(d) PSPLib $OS(E^*) < 0.5$ (e) PSPLib, $OS(E^*) \geq 0.5$

Figure 7 **Influence of time budget T_0**

5.3.3. Influence of the baseline data quality. In this section, we investigate how the outcome of the interview depends on the correct estimation of the probabilities p_{ij}. We estimate probabilities p_{ij} as described in Algorithm 2, but vary the noise index ξ (whose default value was 0.2) from 0.0 to 1.0. The share of task pairs in $\bar{E}_{I,0}$ with 'falsely' estimated probabilities (0.9 for $(i,j) \in E_I^*$ and 0.1 for $(i,j) \notin E_I^*$) increases *subproportionately* with the increase of ξ. For example, at $\xi = 0.2, 0.6$ and 1.0, the share of task pairs in $\bar{E}_{I,0}$ with 'falsely' estimated probabilities equals 0.165, 0.350, and 0.433, respectively.



(a) ASPLib, $OS(E^*) = 0.2$ (b) ASPLib, $OS(E^*) = 0.6$ (c) ASPLib, $OS(E^*) = 0.9$

(d) PSPLib $OS(E^*) < 0.5$ (e) PSPLib, $OS(E^*) \geq 0.5$

Figure 8 Influence of data quality

Figure 8 shows the performance of *Naive*, *Myopic*, and *LSTD* for different data qualities. The noise index is denoted on the lower horizontal axis, the share of "false" probabilities on the upper horizontal axis. If no probabilities are manipulated ($\xi = 0.0$), *Myopic* and *LSTD* remove exactly $T_0 = 150$ precedence relations, because they only ask questions regarding relations with $p_{ij} = 0.9$ and can remove all of them. As expected, *Naive* is not influenced by the data quality as it does not use data on probabilities.

The results of structured interviews with *LSTD* and *Myopic* remain more attractive than the baseline *Naive* policy even with very inaccurate data, although this advantage diminishes with lower data quality. Although, *LSTD* has an edge over *Myopic* in dealing with inaccurate data, it is especially pronounced with higher order strength of the target graph – for ALPLib instances with

$OS(E_I^*) = 0.9$. A possible explanation is the following: *Myopic* states queries to task pairs with high p_{ij} ($p_{ij} = 0.9$) and breaks the ties randomly. At high $OS(E_I^*)$, where the number of independent task pairs is low, the share of task pairs with a precedence relations among the task pairs (i, j) with $p_{ij} = 0.9$ is high, so the probability of a 'mistake' is also high. *LSTD* that relies on multiple features is less sensitive for this effect. In other settings, *LSTD* outperforms *Myopic* moderately by up to 5.3%.

5.4. Performance analysis of LSTD

In this section, we analyze the proposed *LSTD* policy in more detail. We use only instances of ALPLib since the results of PSPLib and ALPLib were similar in all the previous computational studies. We use, however, additional data sets of ALPLib – those with $n = 20$ and $n = 100$ – to investigate the dynamics in the computational time with the growing instance size. In other words, we investigate $525 \cdot 3 = 1575$ instances in total. Since the number of possible precedence relations increases quadratically in the number of tasks n, we choose disproportionally more questions for increasing n. We set $T_0 := 40, 150$ and 600 for $n = 20, 50$ and 100, respectively.

Observe that an exact solution to the DCVP-gen instances is too computationally expensive even for reasonably small instances, therefore we benchmark the performance of *LSTD* to a receding horizon policy, which we dub *Receding*. This type of benchmark is quite common in approximate dynamic programming.

Receding: In *Receding* we exactly solve the DCVP-gen via backwards recursion, but only for a limited horizon of at most T^{RH}, more precisely,

$$X^{RH}(S) = \underset{(i,j)\in X(S)}{\arg\max}\, p_{ij} \cdot (C(S,(i,j),1) + B(\bar{S}_{ij}^{RH})) + (1 - p_{ij}) \cdot B(\underline{S}_{ij}^{RH}) \tag{28}$$

with $\bar{S}_{ij}^{RH} = (\bar{E} \setminus (i,j), \underline{E}, \min(T, T^{RH}) - \tau_{ij})$ and $\underline{S}_{ij}^{RH} = (\bar{E}, \underline{E} \cup (i,j), \min(T, T^{RH}) - \tau_{ij})$. In the following, we set the horizon to $T^{RH} = 2$ such that the interviewer anticipates the next two questions, because of the large computational times per query for instances with $n = 100$.

Table 1 **Comparison of the Approaches for ALPLib,** $n = 20, T_0 = 40$.

	Avg. # removed precedence relations			machine time per
approach	$OS(E^*) = 0.2$	$OS(E^*) = 0.6$	$OS(E^*) = 0.9$	query [s]
Naive	35.7	31.7	22.1	0.000
Myopic	39.0	37.7	27.8	0.000
Receding	39.0	37.7	28.2	0.027
LSTD	39.0	37.8	31.4	0.001

As Tables 1, 2, and 3 show, the performance of *Receding* is quite close to that of *Myopic* and *Receding* never outperforms *LSTD* in the settings considered. For small order strength ($OS(E_I^*) \leq$

Table 2 **Comparison of the Approaches for ALPLib,** $n = 50, T_0 = 150$.

approach	Avg. # removed precedence relations $OS(E^*) = 0.2$	$OS(E^*) = 0.6$	$OS(E^*) = 0.9$	machine time per query [s]
Naive	140.8	126.3	79.8	0.000
Myopic	147.8	144.4	97.5	0.000
Receding	147.7	144.2	98.4	0.322
LSTD	147.7	144.2	116.2	0.011

Table 3 **Comparison of the Approaches for ALPLib,** $n = 100, T_0 = 600$.

approach	Avg. # removed precedence relations $OS(E^*) = 0.2$	$OS(E^*) = 0.6$	$OS(E^*) = 0.9$	machine time per query [s]
Naive	564.7	523.1	383.1	0.001
Myopic	591.9	580.6	418.6	0.001
Receding	591.9	580.7	417.5	3.628
LSTD	591.9	583.7	466.6	0.138

0.6, except $n = 100$), there is no statistically significant difference between *Myopic*, *Receding*, and *LSTD*, whereas *Naive* has an inferior performance. For higher order strength ($OS(E_I^*) = 0.9$), *LSTD*, which anticipates the whole interview, outperforms *Receding* by 14% to 28%.

Overall, *LSTD* is over 25 times faster than *Receding* in terms of the required time per query. Note that for *LSTD*, the value function approximation must be learned upfront. As this task is done before the interview, the runtime is not critical. It is quite fast, with under 1 min for an instance with $n = 50$ and $T_0 = 150$ and under 15 min for an instance with $n = 100$ and $T_0 = 600$.

5.5. Case study

In this section, we show with real-world data that an interview with a limited number of queries can indeed achieve a considerable improvement in the business result.

In our case study, we examine line balancing in the cockpit assembly of a large German automobile manufacturer. The data of the case study was originally described by Gebler (2021).

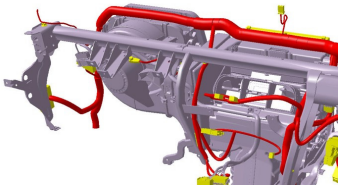


Figure 9: Cockpit

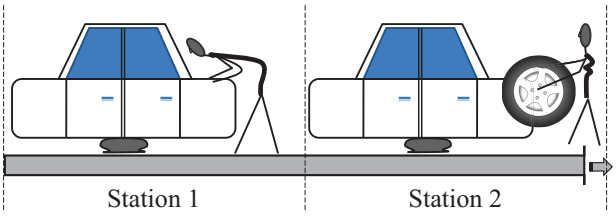


Figure 10: Schematic example of an assembly line with two stations

The cockpit is one of the most complex components of an automobile (see Figure 9). During the cockpit assembly, a standardized pre-assembled cockpit module is installed semi-automatically. A

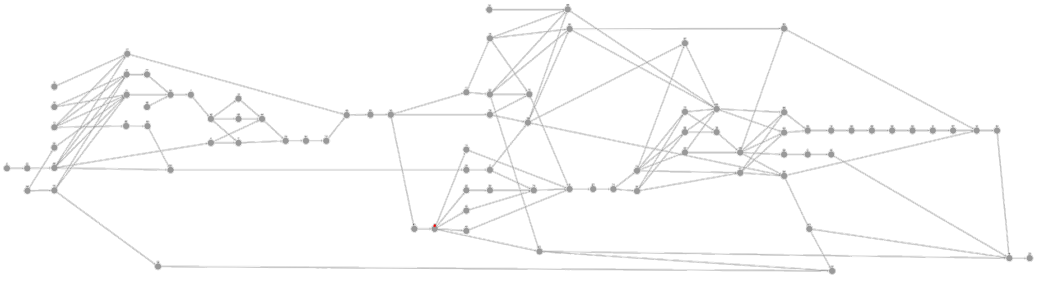
number of additional parts are mounted depending on the customer's order, such as the instrument panel, gear shift levers, navigation and entertainment systems, operating elements, as well as decor elements. The cockpit assembly includes preparatory operations, the mounting operations of the above mentioned parts, as well as the final quality tests.

In the cockpit assembly line, workpieces (painted car bodies) are moved along sequentially arranged stations (see Figure 10 for a schematic representation of an assembly line). Each workpiece is available for assembly at a specific station for a certain amount of time, the so-called *cycle time* c. After that, it is moved to the next station. Each task $i \in V$ is non-preemptive and its duration t_i is given. The precedence relations are described by the true precedence relations set E^* . The objective of assembly line balancing is to assign tasks to stations, such that:

- the total duration of tasks assigned to the same station does not exceed cycle time c,
- the precedence relations are respected,
- the number of required stations is minimized (which is equivalent to the minimization of total unproductive time).

The formulated problem is known as the *simple assembly line balancing problem* (SALBP) in the literature. The SALBP is a well researched problem (see, e.g., Battaia and Dolgui (2013), Baybars (1986), Becker and Scholl (2006), Boysen et al. (2008) and Scholl and Becker (2006) for an overview of the SALBP and its extensions). In the company we studied, further objectives and constraints are important besides the minimization of the total unproductive time. These are, e.g., the minimization of the ergonomic risks, the minimization of the maximum number of different parts requested by one station, and fairness. For the sake of data anonymization and according to the consent with our cooperation partner, we focus on the discussion of the single 'classical' SALBP-objective.

Figure 11 Target Precedence Relations Set Depicted as a Graph.



Note. Nodes are tasks and arcs depict direct precedence relations.

The target graph in the case study consists of 86 tasks (see Figure 11), has an order strength of 0.79 and contains $tr(|E^*|) = 136$ direct precedence relations. Task times vary between 0.05 and

50.69 time units, with an average of 8.85 and a standard deviation of 12.01. Such a distribution of task times that is skewed to the left is common in assembly operations in practice (Dudley 1968, Mason et al. 2005). The remaining parameters are based on consultations with practitioners. In particular, we set the cycle time to $\max_{i \in V} t_i$ and derive the initial maximum precedence relations set $\bar{E}_0$ from a single feasible sequence of tasks. The optimal number of stations that corresponds to $\bar{E}_0$ is 20. We generate probabilities p_{ij} as described in Section 5.1 with the noise index $\xi := 0.2$. We set all times $\tau_{ij} := 1$ since the cockpit assembly belongs to the area of responsibility and expertise of a single industrial engineer. As the minimum precedence relations set, we use $\underline{E} := \{(i,j) \in V \times V | (i,j) \in E^*, t_i + t_j \leq 5\}$. In other words, the precedence relations between small tasks are known.

We perform 20 simulation runs, each with a distinct set of generated probabilities and report an average result for these runs. For each run and examined approach, the SALBP was solved to optimality with the exact algorithm SALOME of Scholl and Klein (1997).

As Table 4 shows, *LSTD* clearly outperforms the other approaches. It removes 5% more precedence relations than the second best approach *Myopic* and 23.4% more precedence relations than the baseline approach *Naive*. Most interesting is how this removal of unnecessary precedence relations translates into the improvement of $F(\cdot|\bar{E})$. In our case, $F(\cdot|\bar{E})$ is the SALBP's objective function – the minimization of the number of stations. After $T_0 := 300$ queries, which is only $\frac{300}{\frac{n(n-1)}{2}} = 8.2\%$ of all possible queries (task pairs), *LSTD* removes 0.25 stations (or about $\frac{0.25}{2} = 12.5\%$) more on average than the other tested interview approaches. This is a significant reduction for the company. Moreover, as our collaborating partner noticed, with 281.40 precedence relations removed by the *LSTD*, a large number of additional task sequences are now recognized as feasible (cf. Section 5.2), which permits the achievement of much better results for the aforementioned additional decision criteria as well (such as ergonomics, fairness etc.).

In total, *LSTD* removes 2.25 stations, which makes for a $\frac{2.25}{20} = 11.25\%$ improvement of the 20-station solution based on $\bar{E}_0$. Assuming that answering one query takes about 1 minute on average, with around 5 hours of the expert's time invested in the data validation guided by LSTD, a significant improvement in the business result can be achieved.

6. Conclusion and future research

In this article, we formulated the Data Collection and Validation Problem (DCVP) centered around the data on precedence relations, which aims to maximize the business result given the limited expert time available for data processing. The DCVP takes one or several known (e.g., manually constructed) feasible sequences of tasks as input. Afterwards, queries can be dynamically stated to the expert about precedence relations regarding selected pairs of tasks (i,j). The duration of the

Table 4 **Performance of the Interview Approaches in the Study of the Cockpit Assembly with $n = 86$, $T_0 = 300$.**

approach π	avg # removed precedence relations	avg reduction in # stations, $F(\cdot\|\bar{E}_0) - F(\cdot\|\bar{E}_\pi)$
Naive	227.9	2.00
Myopic	268.9	2.00
Receding	267.1	2.00
LSTD	281.4	2.25

resulting interview is limited by the time budget T_0. The DCVP outputs a best possible solution for the baseline problem, for which the data is collected,– such as a production plan or a project schedule – which is guaranteed to be feasible given the collected and validated data. In other words, the DCVP can change the relative execution sequence of only those tasks relative to each other, whose precedence relation was *'removed'* by the knowledgeable expert. To the best of our knowledge, we are the first to formulate and analyze the DCVP in literature.

The formulated DCVP is considered a novelty also in terms of the methodological side. This is because it can be interpreted as a robust adjustable multistage optimization problem, in which the uncertainty set depends on the *previously* taken decisions. To the best of our knowledge, no such dynamic approach to uncertainty sets in robust optimization problems has been previously studied within literature.

Since the DCVP depends on the *application-specific* objective function F, we extracted a universally-valid problem counterpart *DCVP-gen* (*general* DCVP) by formulating an appropriate surrogate objective function based on general properties of F. As a consequence, the proposed interview policies and the results of our analysis can be directly applied to a wide range of applications. We modeled the DCVP-gen as a dynamic program and analyzed its complexity and a number of its additional relevant properties. We also suggested a customized least squares temporal difference algorithm LSTD to solve it. We showed that partial validation of the data on precedence relations is valuable and illustrated that structured ('optimized') interview policies with experts can considerably outperform the baseline random interview policy *Naive*. Depending on the setting, *LSTD* outperforms *Naive* by up to 20-40% in terms of removed unnecessary precedence relations, helping to recognize many thousands of additional feasible task sequences. The main performance drivers are the quality of the initially known information on $\bar{E}_0$ and $\underline{E}_0$ as well as the time budget T_0. In a case study on data provided by a large German automaker, we confirm the advantages of the structured interview approach *LSTD*. Most importantly, we show that a limited investment of expert time in data collection and the validation of 300 queries (or 5 hours of expert time assuming 1 minute per query on average) leads to an improvement of about 11.25% in the business result.

In future research, the proposed general ADP-methodology should be adapted to specific applications and specific objective functions F by, for example, adapting the surrogate function and extending the features ϕ_k in the value function approximation.

Future research should also study further application-specific data structures and forms of querying the expert knowledge. For instance, hierarchical data structures of precedence relations are common in the final assembly. Here, certain statements on precedence relations can be formulated for groups of tasks, called modules; e.g., module 'door' can be assembled independently from module 'motor' (Otto and Otto 2014). Moreover, in certain settings, experts may provide information for a group of geometrically related tasks based on CAD-schemes, or comment on the feasibility of (reasonably short) subsequences of tasks. Most importantly, the optimization lens on data validation should be applied to further classes of data.

Acknowledgments

This project was funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) as part of the SuPerPlan project GZ: OT 500/4-1 as well as by the Federal Ministry of Transport and Digital Infrastructure within the framework of the joint project 'KIMoNo' (FKZ: 45KI01A011).

The authors cooperated with Dr. Gebler, Volkswagen AG, and are very grateful to him for his valuable insights into the operational planning on assembly lines.

References

Aho AV, Garey MR, Ullman JD (1972) The transitive reduction of a directed graph. *SIAM Journal on Computing* 1(2):131–137, URL `http://dx.doi.org/10.1137/0201008`.

Antani K, Pearce B, Mears L, Renu R, Kury M, Schulte J (2014) Application of system learning to precedence graph generation for assembly line balancing. *Proceedings of the ASME 2014 International Manufacturing Science and Engineering Conference* (MSEC2014-3906):1–10.

Arun M, Rao C (2010) A CAD system for extraction of mating features in an assembly. *Assembly Automation* 30(2):142–146.

Battaia O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics* 142(2):259–277.

Baybars İ (1986) A survey of exact algorithms for the simple assembly line balancing problem. *Management Science* 32(8):909–932.

Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* 168(3):694–715.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust Optimization* (Princeton University Press: New Jersey).

Bertsimas D, Gupta V, Kallus N (2017) Data-driven robust optimization. *Mathematical Programming* 167(2):235–292.

Billard A, Calinon S, Dillmann R, Schaal S (2008) Survey: Robot programming by demonstration. *Springer Handbook of Robotics* 1371–1394.

Bourjault A (1987) *Contribution a une approche méthodologique del'assemblage automatisé: Elaboration automatique des séquences opératiores.* Ph.D. thesis, L'Université de Franche-Comté.

Boysen N, Fliedner M, Scholl A (2008) Assembly line balancing: Which model to use when? *International Journal of Production Economics* 111(2):509–528.

Boysen N, Fliedner M, Scholl A (2009) Assembly line balancing: Joint precedence graphs under high product variety. *IIE Transactions* 41(3):183–193.

Bozhko A (2020) Minimizing geometric tests in CAAP-systems. *International Conference: Actual Issues of Mechanical Engineering* 111(27th-29th of October).

Davenport T (1997) *Information Ecology: Mastering the Information and Knowledge Environment* (Oxford University Press: New York).

de Fazio T, Rhee S, Whitney D (1999) Design-specific approach to design for assembly (DFA) for complex mechanical assemblies. *IEEE Transactions on Robotics and Automation* 15(5):869–881.

de Fazio T, Whitney D (1987) Simplified generation of all mechanicalassembly sequences. *IEEE Journal of Robotics and Automation* 3(6):640–658.

De Jong H (2002) Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology* 9(1):67–103.

Deepak B, Murali G, Bahubalendruni M, Biswal B (2018) Assembly sequence planning using soft computing methods: A review. *Proceedings of the Institution of Mechanical Engineers Part E: Journal of Process Mechanical Engineering* 233(3):653–683.

Demeulemeester E, Herroelen W (1992) A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science* 38(12):1803–1818.

Demoly F, Yan X, Eynard B, Rivest L, Gomes S (2011) An assembly oriented design framework for product structure engineering and assembly sequence planning. *Robotics and Computer-Integrated Manufacturing* 27(1):33–46.

Dudley NA (1968) *Work Measurement: Some Research Studies* (McMillan Education: London).

Dyer M, Stougie L (2006) Computational complexity of stochastic programming problems. *Mathematical Programming* 106:423–432.

Escobar C, McGovern M, Moralez-Menendez R (2021) Quality 4.0: A review of big data challenges in manufacturing. *Journal of Intelligent Manufacturing* 32:2319–2334.

[Forbes, KPMG] (2016) Now or Never: 2016 Global CEO Outlook. Technical Report June, Forbes Insights, KPMG International.

Gebler M (2021) *Industrialisierung von Optimierungsmethoden zur automatisierten Fließbandabstimmung in der Automobilindustrie*. Ph.D. thesis, Friedrich Schiller University of Jena.

Gertsbakh I, Serafini P (1991) Periodic transportation schedules with flexible departure times: An interactive approach based on the periodic event scheduling problem and the deficit function approach. *European Journal of Operational Research* 50(3):298–309.

Han B, Shang C, Huang D (2021) Multiple kernel learning-aided robust optimization: Learning algorithm, computational tractability, and usage in multi-stage decision-making. *European Journal of Operational Research* 292(3):1004–1018.

Hanasusanto GA, Kuhn D, Wiesemann W (2016) A comment on "Computational complexity of stochastic programming problems". *Mathematical Programming* 159:557–569.

Hauptmeier D, Krumke S, Rambau J, Wirth HC (2001) Euler is standing in line dial-a-ride problems with precedence-constraints. *Discrete Applied Mathematics* 113(1):87–107.

Hlady W, Quenemoen LE, Armenia-Cope RR, Hurt KJ, Malilay J, Noji EK, Wurm G (1994) Use of a modified cluster sampling method to perform rapid needs assessment after hurricane andrew. *Annals of Emergency Medicine* 23(4):719–725.

Homem de Mello L, Sanderson A (1991) A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation* 7:228–240.

Huang M, Smilowitz KR, Balcik B (2013) A continuous approximation approach for assessment routing in disaster relief. *Transportation Research Part B: Methodological* 50:20–41.

Kardosh C, Kovács A, Ványa J (2020) A constraint model for assembly planning. *Journal of Manufacturing Systems* 54:196–203.

Kashkoush M, ElMaraghy H (2014) Consensus tree method for generating master assembly sequence. *Production Engineering: Research and Development* 8:233–242.

Klindworth H, Otto C, Scholl A (2012) On a learning precedence graph concept for the automotive industry. *European Journal of Operational Research* 217(2):259–269.

Kolisch R, Schwindt C, Sprecher A (1999) *Handbook on recent advances in project scheduling*, chapter Benchmark instances for project scheduling problems, 197–212 (Springer: Boston).

Kolisch R, Sprecher A (1996) PSPLIB – A project scheduling library. *European Journal of Operational Research* 96(1):205–216.

Kolisch R, Sprecher A, Drexl A (1995) Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science* 41(10):1693–1703.

Kovacs G, Moshtari M (2019) A roadmap for higher research quality in humanitarian operations: A methodological perspective. *European Journal of Operational Research* 276(2):395–408.

Krishnan V, Ulrich K (2001) Product development decisions: A review of the literature. *Management Science* 47(1):1–21.

Lagriffoul F, Dimitrov D, Saffiotti A, Karlsson L (2012) Constraint propagation on interval bounds for dealing with geometric backtracking. *Proceedings IEEE International Conference on Robotics and Automation* 957–964.

Lambert A (2006) Generation of assembly graphs by systematic analysis of assembly structures. *European Journal of Operational Research* 168(3):932–951.

Lee WP, Lee WP, Tzou WS (2009) Computational methods for discovering genenetworks from expression data. *Briefings in Bioinformatics* 10(4):408–423.

Lin A, Chang T (1993) An integrated approach to automated assembly planning for three-dimensional mechanical products. *The International Journal of Production Research* 31(5):1201–1227.

Manrique R, Sosa J, Marino O, Nunes BP, Cardozo N (2018) Investigating learning resources precedence relations via concept prerequisite learning. *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 198–205 (IEEE).

Markowetz F, Spang R (2007) Inferring cellular networks – a review. *BMC Bioinformatics* 8(S5).

Mason S, Baines T, Kay J, Ladbrook J (2005) Improving the design process for factories: Modeling human performance variation. *Journal of Manufacturing Systems* 24(1):47–54.

Niu X, Ding H, Xiong Y (2003) A hierarchical approach to generating precedence graphs for assembly planning. *International Journal of Machine Tools and Manufacture* 43(14):1473–1486.

Otto A, Otto C, Scholl A (2013) Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research* 228(1):33–45.

Otto C, Otto A (2014) Multiple-source learning precedence graph concept for the automotive industry. *European Journal of Operational Research* 234(1):253–265.

Pardowitz M, Zollner R, Dillmann R (2005) Learning sequential constraints of tasks from user demonstrations. *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 424–429 (IEEE).

Pinnoi A, Wilhelm W (1998) Assembly system design: A branch and cut approach. *Management Science* 44(1):1–148.

Powell WB (2011) *Approximate Dynamic Programming* (John Wiley & Sons, Inc.).

Powell WB, Ryzhov IO (2012) *Optimal Learning* (John Wiley & Sons, Inc.).

Qu LM, Xu X (2013) Relationship matric based automatic assembly sequence generation from a CAD model. *Computer-Aided Design* 45(7):1053–1067.

Redman T (1998) The impact of poor data quality on the typical enterprises. *Communications of the ACM* 41(2):70–82.

Rodrígez I, Bauer A, Nottensteiner K, Leidner D, Grunwald G, Roa M (2021) Autonomous robot planning system for in-space assembly of reconfigurable structures. *2021 IEEE Aerospace Conference* (6-13th of March).

Rodrígez i, Nottensteiner K, Leidner D, Kaßecker M, Stulp F, Albu-Schäfer A (2019) Iteratively refined feasibility checks in robotic assembly sequence planning. *IEEE Robotics and Automation Letters* 4(2):1416–1423.

Sanders Y D ad Tan, Rogers I, Tewkesbury G (2009) An expert system for automatic design-for-assembly. *Assembly Automation* 29(4):378–388.

Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research* 168(3):666–693.

Scholl A, Klein R (1997) SALOME: A bidirectional branch-and-bound procedure for assembly line balancing. *INFORMS Journal on Computing* 9(4):319–451.

Srivastava S, Fang E, Riano L, Chitnis R, Russell S, Abbeel P (2014) Combined task and motion planning through an extensible planner-independent interface layer. *Proceedings IEEE International Conference on Robotics and Automation* (31st of May - 5th of June):639–646.

Su Q (2009) A hierarchical approach on assembly sequence planning and optimal sequences analyzing. *Robotics and Computer-Integrated Manufacturing* 25(1):224–234.

Vigano R, Gómez G (2013) Automatic assembly sequence exploration without precedence definition. *International Journal on Interactive Design and Manufacturing* 7:79–89.

Wan W, Harada K, Nagata K (2017) Assembly sequence planning for motion planning. *Assembly Automation* 38(2):195–206.

Wang L, Keshavarymanesh S, Feng H, Buchal R (2009) Assembly process planning and its future in collaborative manufacturing: A review. *The International Journal of Advanced Manufacturing Technology* 41:132–144.

Wang Y, Liu J (2010) Chaotic particle swarm optimization for assembly sequence planning. *Robotics and Computer-Integrated Manufacturing* 26(2):212–222.

Wang Y, Yuan Z, Sun C (2018) Research on assembly sequence planning and optimization of precast concrete buildings. *Journal of Civil Engineering and Management* 24(2):106–115.

Wojtczak D (2018) On strong NP-completeness of rational problems. *Lecture Notes in Computer Science* 10846:308–320.

Wong WE, Gao R, Li Y, Abreu R, Wotawa F (2016) A survey on software fault localization. *IEEE Transactions on Software Engineering* 42(8):707–740.

Yanıkoğlu İ, Gorissen BL, den Hertog D (2019) A survey of adjustable robust optimization. *European Journal of Operational Research* 277(3):799–813.

Zha X, Lim S, Fok S (1998) Integrated intelligent design and assembly planning: A survey. *The International Journal of Advanced Manufacturing Technology* 14:664–685.

Appendix. Details for Example 2

In Figures 12 - 14, we provide decision trees for the evaluation of the policies mentioned in Figure 3. Decision nodes are illustrated as squares, chance nodes as circles. If a random decision is made as a tie-breaker, this is illustrated as a circle inside the decision node.

Figure 12 Naive Policy for the Example from Figure 3.

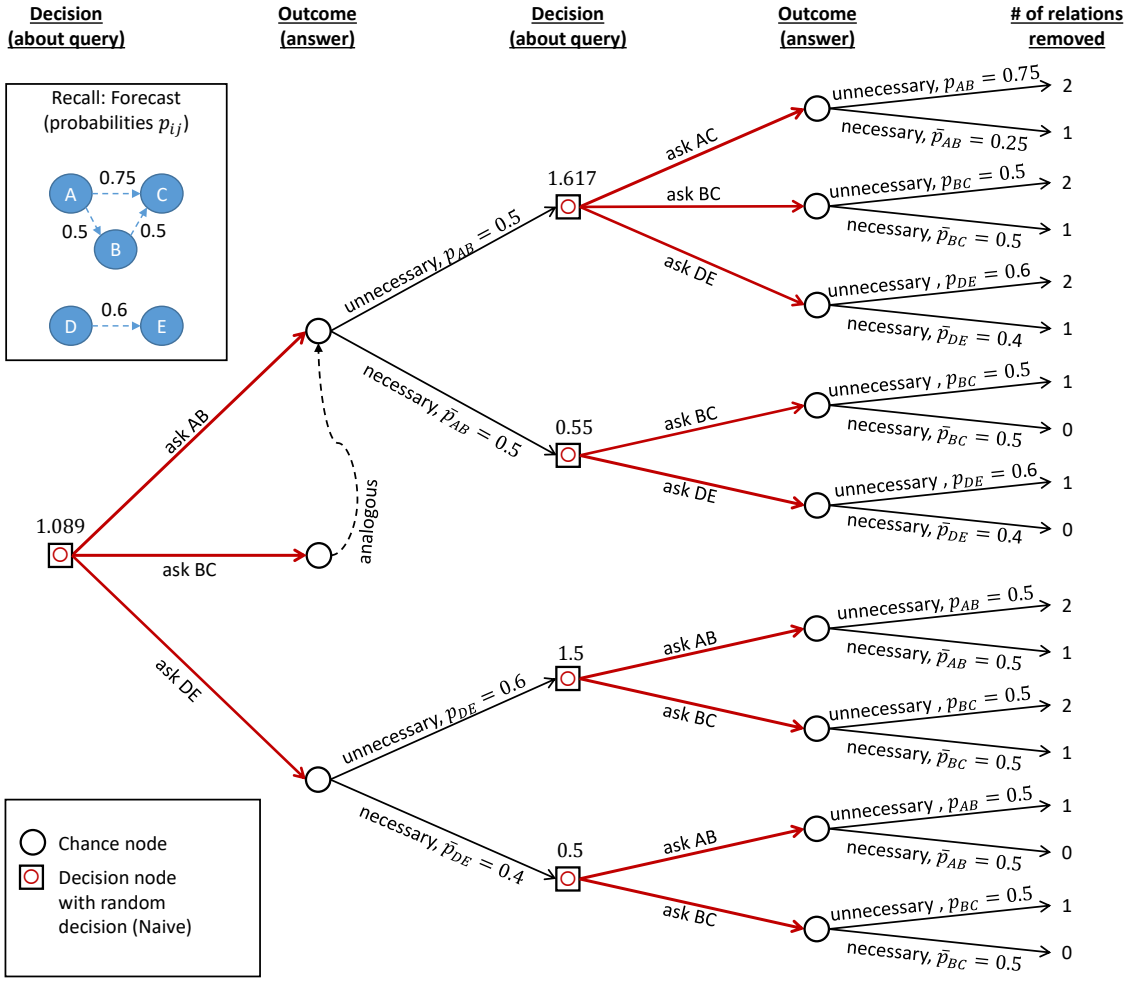


Figure 13 Myopic Policy for the Example from Figure 3.

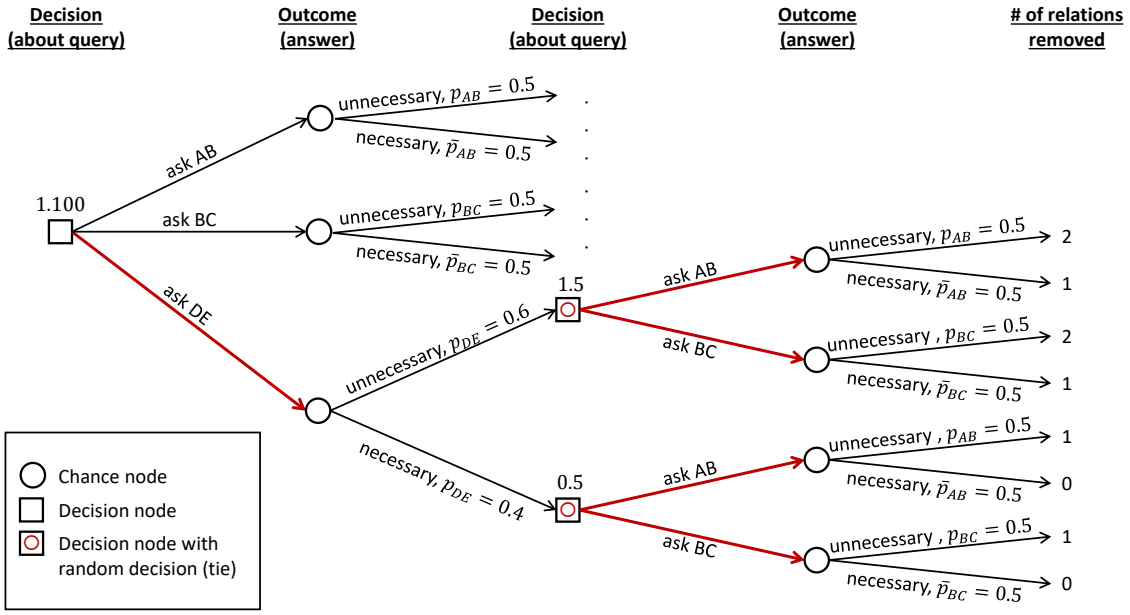


Figure 14 Optimal Policy for the Example from Figure 3.