Maintaining the Utility of Privacy-Aware Schedules

Ali Kaan Tutak[1], **Stephan Fahrenkrog-Petersen**[1], **Arik Senderovich**[2], **J. Christopher Beck**[2], **Matthias Weidlich**[1]

[1]Humboldt-Universität zu Berlin, [2]University of Toronto

{ali.kaan.tutak, stephan.fahrenkrog-petersen,matthias.weidlich}@hu-berlin.de, {sariks,jcb}@mie.utoronto.ca

Abstract

Schedules define how resources process jobs in diverse domains and, therefore, denote a valuable starting point for analysis of the underlying system. However, publishing a schedule may leak private information on the considered jobs. In this paper, we provide a first formulation of a privacy attack on published schedules through the definition of an inverse scheduling problem. Moreover, we show how to quantify the resulting privacy leakage and outline a framework to reduce the attack's impact. That is, we show how a schedule may be distorted to ensure privacy guarantees while limiting the loss in its utility.

Introduction

Schedule-driven systems are pervasive in our lives: outpatient clinics in hospitals, production lines, and public transportation systems are a small sample of applications where a schedule is present (Senderovich et al. 2016). To investigate potential improvements in system performance, it is common for an analyst to have access to the schedules. For example, (Senderovich, Booth, and Beck 2019) analyze the log of a schedule execution to learn an optimization model of the underlying system.

Making schedules public (or at least available to an analyst) is a basic requirement when aiming at data-driven system improvement. However, publishing the schedule, as in any data publishing scenario, may result in leakage of private information (Fung et al. 2010). Specifically, we shall demonstrate that given a published *optimal* schedule, an adversary can potentially infer private information regarding the processed jobs. For example, in a hospital setting, with high probability, an attacker may infer the medical priorities of patients, especially in the presence of even minimal background knowledge about certain patients within the published data (Narayanan and Shmatikov 2008).

Considering the perspective of both the analyst and attacker leads to a privacy-utility trade-off: an increase in privacy decreases the utility of a dataset (Brickell and Shmatikov 2008). To balance both perspectives, we adopt ideas presented in the context of privacy-aware recommendation systems (Yang, Qu, and Cudré-Mauroux 2018). That is, we introduce a privacy model that aims for a strong privacy guarantee, under a limited utility loss budget. As a starting point, our model considers a relatively easy scheduling problem, namely minimizing the weighted sum of completion times in a single-machine setting (Pinedo 2016, Chap. 3.1). We summarize our contributions as follows:

1. We present a threat model that defines an attack on published schedules using inverse scheduling and also propose a probabilistic notion of privacy leakage.
2. We formulate the attack as a constraint satisfaction problem where the adversary attempts to maximize the privacy leakage of the published schedule.
3. We propose a privacy-and-utility protection framework that aims at reducing the damage inflicted by the respective attack.

Background

Privacy Preservation

The privacy-preserving release of datasets has been widely studied (Fung et al. 2010) and two forms of privacy guarantees have received particular attention. First, k-*anonymity* (Sweeney 2002) ensures that at least k individuals are indistinguishable from each other in a published dataset. Second, *differential privacy* (Dwork 2008) bounds the information an adversary can learn about one individual by limiting the impact one individual has on the published dataset. Differential privacy is usually achieved by adding noise to the published data.

Both approaches guarantee a certain level of privacy without bounding the generally inevitable loss in utility of the published data for other purposes (Brickell and Shmatikov 2008). As such, techniques to optimize the resulting utility are an active field of research (LeFevre, DeWitt, and Ramakrishnan 2006; Fioretto and Van Hentenryck 2019) with approaches typically being tailored to specific types of data and analysis purposes.

Single-Resource Scheduling

We consider scheduling problems that comprise a set of n jobs to be processed by a single resource. Each job is assigned a vector of m input features (e.g., due dates and processing times). The resulting schedule must satisfy a set of

constraints and attempt to minimize a given objective function (e.g., the weighted sum of completion times).

Formally, a single-resource scheduling problem is captured as a tuple $\Pi = (J, X, C, \phi)$ where:

- $J = \{j_i\}_{i=1}^{n}$ is the set of jobs to be processed;
- X is an $n \times m$ job feature matrix with $x_{i,j}$ being the j-th feature of the i-th job;
- C is the set of constraints imposed; and
- ϕ is the objective function measuring schedule quality.

A solution or schedule is a vector of start times for the n jobs: $s \in \mathbb{R}^n$.

We denote the domain of all possible job feature combinations as $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ with $\mathcal{X}_j$ representing the domain of feature $j = 1, \ldots, m$. When we consider a specific feature, e.g., weights of jobs, we write $j = w$ and thus the domain is written with the subscript of the corresponding feature, namely $\mathcal{X}_w = [1, 10]$, which implies that the weights of jobs can take values between 1 and 10.

The set C contains constraints that the schedule must satisfy. For example, a schedule may be required to respect precedence constraints between jobs and to avoid any overlap of job executions. Formally, a constraint $c \in C$ can be defined as a subset of schedules, with every schedule in c satisfying a set of conditions that may depend on inputs X.

To assess the quality of a schedule, we define an objective function, $\phi : \mathbb{R}^n \times \mathcal{X} \to \mathbb{R}$ that assigns a real value to a given schedule and job features. An optimal solution to Π is a schedule, $s^* \in \mathbb{R}^n$, that satisfies the constraints, while $\phi(s^*)$ is minimal among all vectors that satisfy the constraints. By $\sigma^* \in \mathbb{N}^+$, we denote the job permutation derived from s^*, e.g., if s_i^* is the smallest start time in s^* then $\sigma_1^* = i$.

The above model captures a class of relatively simple scheduling problems. Note though that it is expressive enough to include both polynomially solvable and NP-complete problems, depending on the specific job features, constraints, and objective function (Pinedo 2016). In this work, we focus on the weighted sum of completion times objective. It is well-known that for a single-machine problem without release dates, the total weighted completion time schedule can be found in polynomial time using the Weighted Shortest Processing Time first (WSPT) rule (Pinedo 2016): by ordering jobs in a non-increasing order of w_i/p_i with p_i and w_i being the processing time and weight of job i, respectively.

Privacy-and-Utility Aware Schedules

In this section, we define the privacy attack on a published schedule and pose the problem of privacy protection under a limited utility loss budget. We conclude the section with an overview of our approach to achieve privacy and utility by distorting optimal schedules.

Threat Model

We start by defining the threat to a published schedule. A single resource is processing n jobs with the objective of minimizing the sum of weighted completion times. Job features, X, that we consider are processing times, $x_{i,1} = p_i$, and weights $x_{i,2} = w_i$ for job $i = 1, \ldots, n$ and $m = 2$. When publishing a schedule s, we assume that the setting is fully known to the attacker, i.e., the job set J, the constraint set C, and the objective function ϕ are known, and the attacker only wishes to determine the values of X. Note that given the published schedule (protected or truly optimal), we assume that the adversary can identify and map the jobs in the schedule to J, fixes the observed processing times, and only attempts to infer the private weights. In case the published schedule is the actual optimal schedule, this assumption of the adversary holds true.

We denote by $\Pi(X)$ the single-resource scheduling problem with fixed J, C, and ϕ, and varying X. Thus, a solver f for $\Pi(X)$, is a function of only X: it solves $\Pi(X)$ and returns an optimal schedule $s^* \in \mathcal{S} \subseteq \mathbb{R}^n$ (illustrated on the left-hand side of Figure 1). Here, the space of schedules is bound by the inputs J, C, and ϕ for some values of X and, hence, denoted by $\mathcal{S}_\Pi$.

The inverse scheduling problem (ISP) to $\Pi(X)$ aims at finding $X \in \mathcal{X}$ s.t., $f(X) = s$ for a given schedule s. In this paper, we assume that the privacy attack is an adversarial attempt to solve the ISP of the weighted sum of completion times given the published schedule. Furthermore, we focus on preventing the leakage of the true weights, which we denote by w_{true}. In other words, we assume that the adversary only aims to infer w_{true}, while processing times are taken from the published schedule.

Two Attacks on Privacy

We return to Figure 1 for a high-level illustration of the considered attack. The adversary assumes that $f(X_{true} = (p_{true}, w_{true})) = s$, i.e., that $s = s^*$. Subsequently, the attacker applies a procedure g (which we shall elaborate on later in this section) in an attempt to find a set $W \subseteq \mathcal{X}_w$ (with $\mathcal{X}_w$ being the domain of the weight feature) such that $W = \{w \in \mathcal{X}_w \mid f(X_{true}) = s\}$. Here, we define the *information leakage* of an attack as the probability of selecting the true weight vector w_{true}. Assuming a lack of an informative prior on W, the leakage of such an attack is $\frac{1}{|W|}$, which is equivalent to the chance of guessing w_{true} out of the set W. Note that the adversary assumes that w_{true} is always within W, an assumption that holds only when $s = s^*$. For the case where $w_{true} \notin W$, we define the leakage to be zero. Additionally, there is the possibility that, given a schedule, there is no possible weight vector in $\mathcal{X}_w$ that yields the given schedule, in which case w_{true} is not in W and hence, by our definition, the leakage is zero again. These two special cases are favorable, as the adversary is not able to infer w_{true}.

We consider two types of attacks: a naive attack that does not take the published schedule into account, and an informed attack that considers the published schedule s, assuming it to be the optimal schedule for the original problem, i.e., $s = s^*$. In the naive attack, the obtained result is $W = \mathcal{X}_w$, i.e., any of the weights in $\mathcal{X}_w$ is a result candidate. Clearly, for both scenarios, the set W may vary in size, and hence the two corresponding leakages may differ from each other. Below, we describe both attacks including their associated information leakages.

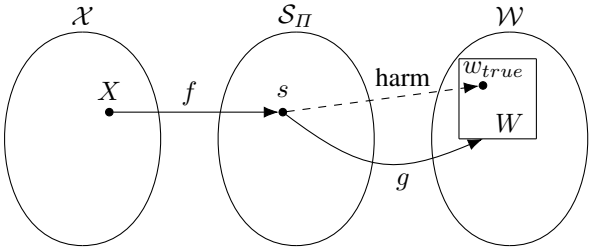

Figure 1: Threat model with inverse scheduling attack.

Naive Attack. Assuming the adversary only knows the parameter domain $\mathcal{X}$ without additional information, the leakage ξ_1, which is defined as the probability that the adversary guesses w_{true}, is given as:

$$\xi_1 = \frac{1}{\Pi_{i=1}^{m} |\mathcal{X}_i|}, \tag{1}$$

with m being the number of job features. Alternatively, if the attacker incorporates the schedule, the following inverse scheduling attack may be attempted.

Inverse Scheduling Attack. To illustrate the intuition of the attack, we revisit Figure 1. In addition to the domain knowledge $\mathcal{X}$ used in the naive attack, the adversary now considers the job permutation σ derived from the published schedule s. Furthermore, the adversary assumes that $s = s^*$ and that processing times p reflected in s are the true job durations. Subsequently, the attacker exploits the knowledge of the objective function to solve an inverse scheduling problem with the decision variables being w_{true}, i.e., the adversary searches for a solution w_{guess} in a potential solution set $W \subseteq \mathcal{X}_w$ that is the set of all weights that may lead to $f(X_{true}) = s$. This becomes the new domain for the weights (all other weights should not be considered since they do not lead to s) and random guessing (as in the naive attack) is applied. We associate the informed attack with the following definition of information leakage:

$$\xi_2 = \begin{cases} 0 & \text{if } w_{true} \notin W \\ \frac{1}{|W|} & \text{otherwise.} \end{cases} \tag{2}$$

As before, the expression ξ_2 for information leakage corresponds to the probability that $w_{guess} = w_{true}$. Note that multiple vectors in $\mathcal{W}$ may lead to the same schedule, and hence, we cannot assume that $W = \{w_{true}\}$. Furthermore, the case in which $W = \emptyset$ is included in $w_{true} \notin W$, and hence we do not run into division by zero.

The inverse scheduling procedure, g, can be represented as the following constraint satisfaction problem (CSP). The input parameters of the CSP are the schedule (that contains the permutation and processing times) and the domain of weights. The weight vector that the adversary attempts to find is the only decision variable. Furthermore, since the objective function ϕ is to minimize the total weighted completion time, we also know that the Weighted Shortest Processing Time first (WSPT) rule was used to get the optimal schedule. This means that the jobs must be sorted in a non-increasing order of their weight to processing time ratio, which are used to add a set of constraints into the CSP.

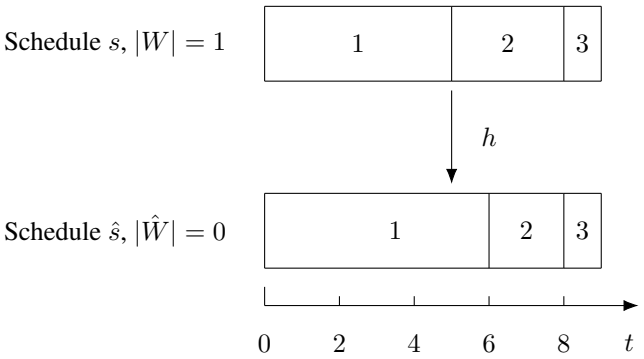

Figure 2: Schedule example for weight domain of $\mathcal{X}_W = [1, 5]$.

The ISP can be formulated as the following CSP:

Inputs:
σ, p: permutation and processing times inferred from s.
$\mathcal{K}_w$: weight domain.

Decision variables:
$w \in \mathcal{X}_w$: vector of weights.

Constraints:

$$\frac{w_{\sigma_i}}{p_{\sigma_i}} \geq \frac{w_{\sigma_{i+1}}}{p_{\sigma_{i+1}}} \qquad \forall\, i \in \{1, \ldots, n-1\}.$$

Output:
w_{guess}: a weight vector s.t., $f(p, w_{guess}) = s$.

Figure 2 illustrates the risk of publishing a schedule s without considering the potential attack. Schedule s consists of three jobs with processing times $p = (5, 3, 1)$ and permutation $\sigma = (1, 2, 3)$. We assume that the adversary knows the weight domain $w_i \in \mathcal{X}_w = [1, 5]$. The result of the inverse scheduling attack on schedule s is $W = \{(5, 3, 1)\}$. There is only a single possible weight vector, so that the adversary does not even need to guess and the private information is inadvertently leaked with the schedule.

While the goal of the adversary is to guess a single 'true' vector of weights that satisfies the constraints, the amount of information leaked in an attack is tied to the number of possible solutions $|W|$. Aiming to reduce the chances of the attacker to succeed, one may either inflate the size of W such that privacy is sufficiently preserved or try to ensure that $w_{true} \notin W$. Both of these strategies are enabled by our proposed protection framework.

Privacy-and-Utility-Aware Publishing of Schedules

We wish to publish a schedule that provides the desired privacy protection, while considering another player beside the adversary, namely the analyst. The analyst is interested in analyzing the data for useful causes, e.g., for improving the underlying system. More formally, we shall assume that a published schedule can be associated with a utility score that would decline as the schedule is distorted. In the extreme, the utility score of an unpublished schedule is 0, while the utility score of the optimal schedule is the maximal one.

With these considerations in mind, we wish to publish a schedule that would strike a trade-off between privacy (protecting from harm) and utility (assisting the analyst).

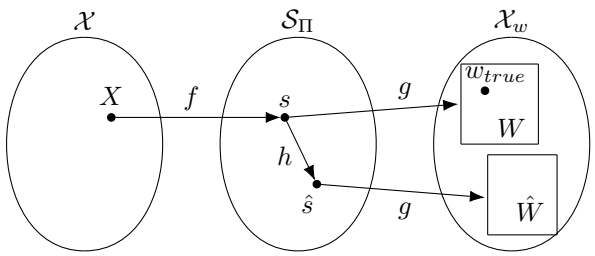

Figure 3: Distortion model.

Figure 3 shows our idea of distorting the schedule, which extends the threat model presented earlier in Figure 1. In addition to the scheduling function f and inverse scheduling attack function g, we consider a distortion function $h : \mathcal{S} \rightarrow \mathcal{S}$ that changes the schedule by, for instance, modifying the start times of jobs.

Our goal is to publish a distorted schedule $\hat{s}$ (not necessarily optimal for the scheduling problem), that attempts to decrease information leakage by increasing the size of W.

The model shows an inverse scheduling attack on the distorted schedule with $g(\hat{s}) = \hat{W} = \{w \in \mathcal{X}_w \mid f(\hat{p}, w) = \hat{s}\}$, with $\hat{p}$ being the changed processing times that can be inferred from $\hat{s}$. In this example, the results of the inverse scheduling attacks for s and $\hat{s}$ are different, i.e., $W \neq \hat{W}$, and $w_{true} \notin \hat{W}$, which leads to an information leakage quantified by ξ_2. The observation actually points at a feature of our approach: it may return a 'useful' schedule that prevents the ISP attack from being effective (in fact, it will have zero probability to succeed).

So far, we have discussed the distortion mechanism as a tool that attempts to increase privacy. However, we also aim at preserving utility for the analyst. To this end, we shall define a schedule property as a function, $z : \mathbb{R}^n \rightarrow \mathbb{R}^v$, that maps a schedule to v real-values (e.g., makespan, average waiting time, and weighted sum of completion times). Note that our objective function ϕ is, in essence, a one-dimensional property function. Next, we define $d : \mathbb{R}^v \times \mathbb{R}^v \rightarrow \mathbb{R}$ to be a metric that measures the difference between two schedules in the corresponding property space. For example, d can be the absolute difference between the average waiting time of jobs in the optimal schedule and in the published schedule.

Lastly, we define two thresholds: ϵ, which is a privacy threshold on the minimal allowed information leakage ξ_2 that the user of our approach targets, and a utility threshold δ that bounds $d(z(s^*), z(\hat{s}))$. With the setting described in Figure 3 and the inputs of z, d, δ, ϵ and s^* in mind, the privacy-and-utility preservation problem (PUP) is another CSP in which we aim at finding $\hat{s}$ that would remain within δ proximity from s^*, yet will assure that $\xi_2 \leq \epsilon$.

Returning to Figure 2, we show one possible way of protecting the schedule s from information leakage. To this end, we assume that our utility is to preserve the makespan of the schedule in the interest of the analyst. Then, the distorted schedule $h(s) = \hat{s}$ (under minimal changes) provides the protection from leaking the private data. The start time of the second job is shifted by one time unit, resulting in the ISP having no solution $|\hat{W}| = \emptyset$. The adversary is not able to identify the true weight vector when applying the attack.

Conclusion

In this paper, we considered a setting where a published schedule may expose private information of the jobs processed by a single resource. In an attempt to conceal private job features, we propose to distort the schedule such that a probabilistic privacy guarantee is achieved. In addition, we preserve the utility of the published schedule in the eyes of an analyst that aims to improve the underlying system.

As our contributions, we first formulated a privacy attack on published schedules as an inverse scheduling problem (ISP). Second, we provided a framework for protecting the schedule against such an ISP attack, while assuring utility for an analyst. The framework, however, does not guarantee that a schedule satisfying all constraints will be found, as a schedule that satisfies all constraints may not exist. Under such circumstances, there might not exist a schedule that would be fit for publishing for privacy and/or utility reasons.

In future work, we shall formalize the privacy-and-utility preserving problem (PUP) and develop a distortion mechanism to balance between privacy preservation and the analyst's utility function. In addition, we will explore the computational aspects of both ISP and PUP for different classes of scheduling problems, including the aforementioned minimization of weighted sum of completion times.

References

Brickell, J.; and Shmatikov, V. 2008. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 70–78.

Dwork, C. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, 1–19. Springer.

Fioretto, F.; and Van Hentenryck, P. 2019. Differential privacy of hierarchical census data: An optimization approach. In *International Conference on Principles and Practice of Constraint Programming*, 639–655. Springer.

Fung, B. C.; Wang, K.; Chen, R.; and Yu, P. S. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)* 42(4): 1–53.

LeFevre, K.; DeWitt, D. J.; and Ramakrishnan, R. 2006. Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06)*, 25–25. IEEE.

Narayanan, A.; and Shmatikov, V. 2008. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 111–125. IEEE.

Pinedo, M. 2016. *Scheduling : theory, algorithms, and systems*. Cham : Springer, fifth edition edition. ISBN 9783319265803. URL http://er.llcc.edu:2048/login?url=http://link.springer.com/10.1007/978-3-319-26580-3. Includes bibliographical references and indexes.

Senderovich, A.; Booth, K. E.; and Beck, J. C. 2019. Learning Scheduling Models from Event Data. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 401–409.

Senderovich, A.; Rogge-Solti, A.; Gal, A.; Mendling, J.; Mandelbaum, A.; Kadish, S.; and Bunnell, C. A. 2016. Data-driven performance analysis of scheduled processes. In *International Conference on Business Process Management*, 35–52. Springer.

Sweeney, L. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10(05): 557–570.

Yang, D.; Qu, B.; and Cudré-Mauroux, P. 2018. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31(3): 507–520.