This is the author's accepted version.The publisher's version is availabe at htt

Designing Strong Privacy Metrics Suites Using Evolutionary Optimization

ISABEL WAGNER, De Montfort University, UK
IRYNA YEVSEYEVA, De Montfort University, UK

The ability to measure privacy accurately and consistently is key in the development of new privacy protections. However, recent studies have uncovered weaknesses in existing privacy metrics, as well as weaknesses caused by the use of only a single privacy metric. Metrics suites, or combinations of privacy metrics, are a promising mechanism to alleviate these weaknesses, if we can solve two open problems: which metrics should be combined, and how. In this paper, we tackle the first problem, i.e. the selection of metrics for strong metrics suites, by formulating it as a knapsack optimization problem with both single and multiple objectives. Because solving this problem exactly is difficult due to the large number of combinations and many qualities/objectives that need to be evaluated for each metrics suite, we apply 16 existing evolutionary and metaheuristic optimization algorithms. We solve the optimization problem for three privacy application domains: genomic privacy, graph privacy, and vehicular communications privacy. We find that the resulting metrics suites have better properties, i.e. higher monotonicity, diversity, evenness, and shared value range, than previously proposed metrics suites.

CCS Concepts: • **Security and privacy → Privacy protections**; *Privacy-preserving protocols*; • **General and reference → Metrics**; **Measurement**; • **Computing methodologies → Genetic algorithms**; • **Theory of computation → Evolutionary algorithms**.

Additional Key Words and Phrases: privacy metrics, privacy measurement, metrics suites, genomics privacy, graph privacy, vehicular privacy, monotonicity, evolutionary optimization, multiobjective optimization

ACM Reference Format:
Isabel Wagner and Iryna Yevseyeva. 2020. Designing Strong Privacy Metrics Suites Using Evolutionary Optimization. 1, 1 (November 2020), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Methods to measure privacy as accurately and consistently as possible are important for the development of new privacy-enhancing technologies (PETs): without a means to measure and compare the effectiveness of PETs, it is unclear to what extent new PETs improve on existing PETs.

Privacy measurement is a hard problem because the concept of privacy can be defined in different ways depending on the privacy domain, and on who is making the definition. As a result, privacy measurement has been an active research area for many years, and many different metrics have been proposed [54]. However, not all metrics are equally good and research has identified many weaknesses of existing metrics. For example, some privacy metrics are hard to interpret [13], inconsistent [53], or only suitable for narrow application domains [25]. As a result, researchers

Authors' addresses: Isabel Wagner, De Montfort University, Cyber Technology Institute, The Gateway, Leicester, LE19BH, UK, isabel.wagner@dmu.ac.uk; Iryna Yevseyeva, De Montfort University, Cyber Technology Institute, The Gateway, Leicester, LE19BH, UK, iryna@dmu.ac.uk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
XXXX-XXXX/2020/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

have called for using combinations of metrics to evaluate new PETs instead of using single metrics [22, 45, 54, 59]. Intuitively, the use of combinations of metrics is easily justified because it improves the diversity of measurement and can improve the interpretation of results by giving insight into different aspects of privacy. In addition, combinations of metrics can make the ranking of alternative PETs more robust. Recent work has also shown that combinations of metrics, or metrics suites, can improve the monotonicity of privacy measurement [60], which is a key requirement for consistent measurement.

However, there are two open issues: first, how the metrics should be combined, and second, which metrics should be selected. To address the first issue, the use of multi-criteria decision analysis methods (MCDA) was recently proposed [60]. The second issue has so far been addressed in a rather ad-hoc fashion, either relying on convenience, i.e. selecting metrics that are implemented and readily available, or relying on professional judgment, i.e. selecting metrics that are likely to "work" in a given application, or relying on prior work, i.e. selecting metrics that have previously been used [54]. In this paper, we use optimization methods to systematically address this issue.

Privacy metrics suites selection can be presented as a knapsack optimization problem [35], which can be difficult to solve due to its large combinatorial search space, especially for problems with more than three objectives. To tackle this problem of computational complexity, various approximation methods, such as *heuristics* and *metaheuristics* [49], many of which are nature-inspired, have been developed for both single-objective and multi-objective (MO) optimization problems. These methods are particularly useful for finding approximate solutions in cases when exact analytic methods struggle to reach optimum in feasible time.

Prior work has suggested a number of criteria for privacy metrics, including monotonicity, evenness, and shared value range [59]. We argue that privacy metrics suites need to perform well on these criteria, and in addition they should include a diverse selection of metrics, and the number of metrics in a suite should be limited to make them easy to apply. In this paper, we contribute a new optimization-based method to create strong suites of privacy metrics. In particular, we aim to answer three research questions:

RQ1. What is the optimal composition of metrics suites, in terms of which metrics are selected and how they are weighted?

RQ2 Which optimization algorithms and which open source packages are most suitable to solve our optimization problem?

RQ3 What are the trade-offs between different optimization objectives for privacy metrics suites?

In detail, our contributions are as follows.

- We formally define the selection and weighting of privacy metrics in a metrics suite as a many-objective optimization problem that maximizes five objectives that indicate the quality of privacy metrics: monotonicity, diversity, evenness, shared value range, and number of metrics (Section 3).
- We use 16 evolutionary optimization algorithms to solve our optimization problem for three application domains: genomic privacy, graph privacy, and vehicular communications privacy (Sections 4 and 5).
- We evaluate the performance of different evolutionary optimization algorithms on our many-objective problem and find that older, well-established algorithms like NSGA-II perform better on our problem and datasets than modern algorithms specifically developed to solve many-objective problems (Section B).
- We analyze the metrics suites generated by the optimization and find that trade-offs have to be made especially between evenness and shared value range, but also between the effort of measurement (i.e., the number of metrics in a suite), monotonicity, and diversity. We have

published our dataset of optimization results so that researchers can choose metrics suites according to their own trade-offs [55] (Section 6).
- We apply our metrics suites to the problem of ranking anonymization algorithms in graph privacy. We find that our metrics suites rank the algorithms more consistently than individual metrics (Section 7.1).
- We compare the composition of metrics suites for each application domain and find that the selection and weights of metrics differs significantly between domains, leading to the conclusion that there is no one-size-fits-all metrics suite (Section 7.2).

2 BACKGROUND

The research field of creating new privacy-enhancing technologies (PETs) has been very active in the last two decades, proposing new PETs in various domains, for various data types, and for various assumptions about the adversary's capabilities.

To illustrate our general optimization-based approach to to the creation of strong privacy metrics suites, we have selected three domains – genomic privacy, graph privacy, and vehicular network privacy – that differ in their data types and assumptions about the adversary. We explain these three domains below, followed by an overview of existing privacy metrics and their characteristics, and a discussion of optimization algorithms.

2.1 Privacy Domains

In *genomic privacy*, the aim is to protect knowledge about which genomic variations are present in an individual's genome because these variations can predict, for example, the individual's susceptibility to diseases like cancer. The adversary is assumed to have prior knowledge, for example about the population-wide distributions of genomic variations or about the individual's relatives or ethnicity. To preserve the utility in genomic information, for example in research or healthcare, PETs for genomic privacy have included homomorphic encryption and secure multi-party computation [2].

In *graph privacy*, the aim is to protect user identities in anonymized social graphs. The adversary is assumed to have knowledge of an auxiliary, non-anonymized, partially overlapping social graph as well as the identities of some seed nodes in both graphs and then attempts to re-identify other users based on similarities in graph structures [38]. PETs for graph privacy have included approaches to use noise to anonymize graph structures or to publish synthetic graphs.

In *vehicular network privacy*, the aim is to prevent the adversary from linking subsequent positions of the user's vehicle. The adversary is assumed to be able to overhear messages sent by the vehicle, such as safety broadcasts, and tries to track individual vehicles. PETs in this area have to make sure that privacy protection does not interfere with the safety function provided by the vehicular broadcasts, which can, for example, help avoid collisions at intersections with limited visibility [14].

In Section 5.1, we give technical details for how each domain was included in our study.

2.2 Privacy Measurement

Many privacy metrics have been proposed in the literature: a recent survey of technical privacy metrics, i.e. privacy metrics that measure the technical effectiveness of PETs, discussed more than 80 different privacy metrics [54]. These metrics can be grouped into different categories depending on what aspect of privacy they measure. For example, some metrics measure the adversary's success probability while others focus on the adversary's uncertainty (e.g., entropy) or the information gained by the adversary (e.g., mutual information). Eight categories were proposed in [54], and Table 1 shows that the datasets we used for the three privacy domains relied on metrics from 5–7 of these categories. We refer to [54] for further details on individual metrics.

Table 1. Privacy metrics for each of the three privacy domains in this study and their categories from [54].

Category	Metric	Genomics [53]	Graph [60]	Vanet [59]
Uncertainty	Anonymity set size	–	x	x
	Asymmetric entropy	x	–	–
	Asymmetric entropy per SNP	x	–	–
	Collision entropy	–	x	x
	Conditional entropy	x	x	x
	Conditional privacy	–	x	x
	Cross entropy	–	–	x
	Cumulative entropy	x	–	x
	Entropy	x	x	x
	Genomic privacy	x	–	–
	Inherent privacy	x	x	x
	Max entropy	–	x	x
	Mean entropy	x	–	–
	Min entropy	x	x	x
	Normalized entropy	x	x	x
	Quantiles anonymity set size	–	x	–
	Quantiles entropy	–	x	x
Information gain	Amount of leaked information	x	x	x
	Conditional privacy loss	x	x	x
	Increase in adversary belief	–	–	x
	Information surprisal	x	x	–
	Loss of anonymity	–	x	–
	Mutual information	x	x	x
	Normalized mutual information	x	–	x
	Pearson correlation	–	x	x
	Relative entropy	x	x	x
	Variation of information	x	–	–
Similarity	Coefficient of determination	x	–	–
	Normalized variance	–	x	x
Indistinguishability	Information privacy	–	–	x
Success	Adversary's overall success	–	x	–
	Adversary's success rate	x	x	x
	Hiding property	–	x	x
	Privacy breach level	–	–	x
	User-specified innocence	x	x	x
Error	Absolute error	–	x	–
	Expected distortion	–	–	x
	Expected error	x	–	x
	Incorrectness	–	x	x
	Mean error	x	–	–
	Mean squared error	x	x	–
	Percentage incorrectly classified	x	x	x
Time	Distance to confusion	–	–	x
	Mean tracking duration	–	–	x
	Time to confusion	–	–	x

2.2.1 Criteria for Privacy Metrics. Several criteria have been proposed to judge the quality of privacy metrics. One criterion is monotonicity – the notion that privacy metrics should indicate decreasing privacy with increasing adversary strength [53]. Monotonicity is an important requirement because non-monotonic metrics may not be able to distinguish high and low privacy levels.

Another set of criteria focuses on different aspects of privacy. For example, metrics should indicate the adversary's chances of success [1], the resources required by the adversary [48], the portion of data that is not protected [6], and the accuracy, uncertainty, and correctness of the adversary's estimate [45]. Taken together, these criteria indicate that a single privacy metric is unlikely to be sufficient to measure privacy.

The final set of criteria evaluates how well privacy metrics can compare privacy levels within a scenario (e.g. privacy for two users of the same social network) and between scenarios (e.g. privacy for users of two different social networks). Metrics whose values are spread evenly across their value range (*evenness*) are suitable for within-scenario comparisons, and metrics that have a large *shared value range* across different scenarios are suitable for between-scenario comparisons [59].

2.2.2 Combinations of Privacy Metrics. It is unlikely that any single privacy metric can meet all of the above criteria. Several authors have therefore proposed to use several different metrics and combine their evaluation [22, 41, 54, 59].

For example, Oya et al. [41] studied location privacy in location-based services and used two metrics: the adversary's expected error to quantify correctness, and conditional entropy to quantify uncertainty. They argue that considering both metrics is essential to avoid designing privacy mechanisms that provide only low privacy levels.

In addition, Oya et al. observe that none of the privacy mechanisms in their study performed equally well on the two metrics. This observation has two important implications. First, there is a need to use multiple metrics to understand all aspects of the privacy protection provided by a mechanism. Second, to achieve an objective ranking of privacy mechanisms, there is a need to systematically combine the results from multiple privacy metrics.

In prior work, we have proposed radar charts as a visual method for combination [53] and methods from multi-criteria decision analysis as an analytic method [59]. In this paper, we build on this analytic method to improve the design of privacy metrics suites (see Section 3.2).

2.2.3 Utility. Data utility is a concern that is orthogonal to privacy measurement, i.e. developers of new PETs need to measure both privacy and utility to determine whether the new PET is suitable for the intended application. In this paper, we are concerned with improving the quality of *privacy measurement* through optimized metrics suites. As such we are concerned with including metrics for different properties of privacy in our metrics suites. Because utility is not a property of privacy, but rather of the system to which PETs are applied, we do not include utility measures in our study. A promising line of future work may be to study whether utility measurement can be improved in a similar manner, or whether it is possible to combine privacy and utility metrics into a single measure.

2.3 Optimization Problems

Selecting a strong suite of privacy metrics can be formulated as a *knapsack problem*. The knapsack problem is a well-known combinatorial NP-hard problem [35]. In this problem, the most valuable items should be selected out of a given set of items in such a way that the value of the selected set is maximized. The knapsack is usually limited to a certain capacity or weight, hence, the selected knapsack should not exceed that capacity. Depending on the way of presenting items, binary, integer and continuous formulations are available, addressing either selection of an item (with

0/1 variables), a number of each type of items or assignment of a real weight value to each item, respectively [34].

Solving each type of knapsack problem is often difficult by exact methods [26, 35]. The difficulty can be related to the problem size, e.g., number of available items, size of knapsack, number of objectives to optimize simultaneously and constraints to be satisfied. Similarly, selecting the strongest suite of privacy metrics by solving a knapsack problem is computationally hard due to the large number of metrics combinations, each of which takes a number of parameters for evaluating a number of their qualities (objectives), such as monotonicity, diversity in inputs/outputs, evenness, and shared value range. For this reason, we use heuristics, in particular evolutionary algorithms, to solve our metrics suite optimization problem. In this work we represent a privacy metric in a suite with a real weight. Using real-valued weights allows performing selection and weighting of metrics in one step.

2.3.1 Evolutionary Optimization. Evolutionary Algorithms (EAs) are nature-inspired optimization algorithms that search for a good approximation of the optimal solution in feasible time. EAs have been shown to be efficient in searching large search spaces when solving single-objective and multi-objective, discrete and continuous, constrained and unconstrained problems [3, 4], including knapsack problems. Moreover, various benchmark knapsack problems are often used to test newly developed EAs [19], [63].

Genetic Algorithms (GAs) [17] are among the most well-known EAs, mimicking genetic evolution and the process of natural selection of species observed in nature. Initially, a GA randomly generates a population of individuals, each of which is a potential solution. At each iteration a sub-population of individuals is selected, mutated and recombined, and the best individuals of this sub-population are selected as the population for the next iteration. The evolutionary process is repeated until a stopping criterion is met, for example until the maximal number of iterations or function evaluations is reached. The best individuals are presented as the final approximation solution. The difference between various EAs are in the way they perform selection and variation (mutation and recombination) of individuals.

A number of improvements were proposed to make EAs more powerful, such as *elitism*, which is preserving best individuals in the population for the future generations; and *diversity maintenance*, which is preserving diversity in genotype and/or phenotype across population [4].

Due to the stochastic nature of EAs, it is recommended to run evolutionary processes for a large enough number of iterations and for a large enough number of runs to ensure stable approximation results [4]. For instance, the number of iterations per run is usually selected based on some test runs and observing convergence of performance.

EAs are particularly suitable for our problem because they typically find good approximations, are efficient, can explore large search spaces, and can handle many objectives. We describe the single- and multi-objective evolutionary algorithms we used in this study in more detail in Sections A.1 and A.2.

Aiming at a three different privacy application domains, we used as many algorithms as possible, as according to the "no free lunch theorem in optimisation", there is no single algorithm that performs best across all applications. Hence, in addition to EAs, some other nature-inspired metaheuristics, such as particle swam optimisation, were used whenever their implementation in Python was available.

2.3.2 Multi-Objective Optimization. Optimization with multiple objectives is difficult because it is not enough to search for a single solution with the best value on a single objective, but instead for a set of solutions that are optimal on at least one of the objectives. These solutions are called *Pareto optimal solutions*. The set of Pareto optimal solutions is called the *Pareto set* in the solution

space and the *Pareto front* in the objective space. The Pareto front represents trade-offs between objectives, where each solution is better than the others on at least one objective. To select the Pareto optimal solutions, all candidate solutions have to be evaluated on all objectives and compared pairwisely based on the *weak non-dominance* relation [36]. The solution x_i is considered to be *weakly non-dominated* with respect to another solution x_j if it is as good as x_j on all objectives, and is better on at least one of the objectives.

Formally, weak non-dominance can be written as follows assuming minimisation of all objectives:

$$x_i \prec x_j \Leftrightarrow f_m(x_i) \leq f_m(x_j) \forall m \in M \text{ and } \exists l \in M | f_l(x_i) < f_l(x_j), m \neq l. \quad (1)$$

Other types of dominance are available for search of optimal solutions but are less common.

One of the approaches to solving multi-objective problems is to combine all objectives into a single one by assigning relative weights to each objective. Even though this approach looks attractive due to its simplicity, it has some drawbacks, e.g. not being able to consider all Pareto optimal solutions in case of non-convex objective functions [36]. This approach is often used in an a posteriori approach to multi-objective optimization [36] and MCDA [51], where a single solution needs to be selected from the Pareto set, e.g. based on the preferences of a decision maker. Such preferences can be expressed in different forms, e.g. as objectives weights representing importance of each objective to decision maker(s), or as reference solutions, which decision maker(s) would want to obtain if possible and towards which the optimization search can be guided [32].

3 PROBLEM FORMULATION

Our aim is to find a weighted selection of privacy metrics – a metrics suite – that can be used to evaluate PETs. The selection and weighting of metrics is guided by *objectives* that express desirable qualities of privacy measurement. In this paper, we focus on five objectives: monotonicity, diversity, evenness, shared value range, and number of metrics. We selected these five objectives because they have been used as criteria for the strength of privacy metrics in the literature [59, 60], and because they could be expressed formally and implemented in our optimization.

In general, additional objectives and alternative definitions of the objectives are possible. For example, additional objectives may include whether metrics are easy to interpret, whether they can measure privacy for individual users, or whether they can measure privacy over time. Alternative definitions for objectives are common in other areas, for example in the evaluation of EA solution sets, where 10+ metrics exist for uniformity (similar to our *evenness*). We hope to see future work that explores other objectives and improves on our work.

In this section, we formulate the problem of selecting and weighting privacy metrics as a mathematical optimization problem, independent of the application domain, and describe the five objectives that should be optimized to find the best metrics suite. We cast this problem as a real-valued optimization problem because this allows us to solve two problems at once: the problem of *selecting* which metrics will be included in a metrics suite, and the problem of *weighting* each metric so that the properties of the resulting metrics suites are optimized.

In Section B, we will use published empirical data to compute metrics suites for three specific application domains: genomic privacy, vehicular communications privacy, and graph privacy.

3.1 Preliminaries

Metrics suites are composed of a number of privacy metrics v_k drawn from a pool V of candidate metrics. V varies depending on the application domain because different privacy metrics may be applicable in each domain. Each metric contributes to the metrics suite according to its weight $w_k \in [0, 1]$. A weight of $w_k = 0$ indicates that the metric v_k is not part of the metrics suite. We use the Weighted Product Model (described in detail below), a method from multi-criteria decision

analysis, to combine individual metrics into a metrics suite. We are interested in finding the best metrics suite $\vec{v}$ with corresponding weights $\vec{w}$.

Each application domain consists of a set of scenarios S. Each scenario combines user behavior, user data, adversary behavior, and an ordered list of adversary strength levels L. For example, user behavior and user data in graph privacy are given by a specific anonymization algorithm applied to a specific graph; adversary behavior is a specific de-anonymization algorithm; and the adversary's strength is determined by how much prior information is at the adversary's disposal, with more prior information indicating a stronger adversary.

3.2 Objective M_1: Monotonicity

The most important requirement in privacy measurement is that the measurement is monotonic, that is, privacy metrics should indicate lower privacy for stronger adversaries [53, 59]. Monotonicity is essential because non-monotonic metrics could indicate the same privacy levels for a strong privacy-enhancing technology (PET) and for a weak one, and as a result they could make PETs appear stronger than they are.

To evaluate the monotonicity of *a single privacy metric*, the metric is computed for an ordered list of adversary strength levels L. For each pair of successive strength levels, the change in the metric's value is analyzed: if the change is in the expected direction, i.e. shows decreasing privacy for increasing adversary strength, then the metric is monotonic for this pair of adversary strengths. The percentage of pairwise strength levels where the change in metric value is in the expected direction is an indicator for monotonicity.

To evaluate the monotonicity of *a metrics suite*, we aggregate the individual privacy metrics in a suite using a multi-criteria decision analysis (MCDA) method. Specifically, we use the Weighted Product Model (WPM) as a well-established MCDA method [51]. WPM computes the relative ranking of a set of alternatives (i.e., it does not compute absolute privacy values). We use the list of adversary strength levels L in each scenario to represent this set of alternatives. Because we know the ground truth of how the strength levels *should* be ranked, we can judge the monotonicity of metrics suites [60].

The advantage of using a multiplicative model (WPM) instead of an additive model like the Weighted Sum Model is that the multiplicative model can eliminate bias due to the magnitude or units of individual metrics, can correctly integrate higher-better and lower-better metrics, and avoids rank reversals that can occur with additive models [50]. However, the score computed by WPM is useful only for ranking and should not be interpreted [50].

To rank the alternatives, WPM computes the metrics suite score Q_l for each alternative:

$$Q_l = \prod_{k=1}^{n} (\overline{x}_{lk})^{w_k},$$

where x_{lk} is the value of metric v_k for strength level l, and $\overline{x}_{lk}$ is normalized:

$$\overline{x}_{lk} = \begin{cases} \frac{x_{lk}}{\max_l x_{lk}}, & \text{if higher values indicate higher privacy,} \\ \frac{\min_l x_{lk}}{x_{lk}}, & \text{if lower values indicate higher privacy.} \end{cases}$$

The monotonicity criterion c_m formalizes the notion whether the WPM ranking for a pair of alternatives is monotonic or not. Note that this criterion depends on knowing the correct ordering of the alternatives L.

$$c_m(Q_l, Q_{l+1}) = \begin{cases} 1 & \text{if } Q_l - Q_{l+1} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the objective function M_1 computes the portion of these pairwise rankings that are monotonic across all scenarios S.

$$M_1(S) = \frac{1}{|S|(|L|-1)} \sum_{s \in S} \sum_{l=1}^{|L|-1} c_m(Q_l, Q_{l+1}). \tag{2}$$

To simultaneously solve the two problems of selecting/deselecting privacy metrics and choosing their weights, we adjust the value range for weights to $w_k \in [-1, 1]$. Metrics with weights $w_k \leq 0$ are considered not selected. In this way, we ensure that there is a reasonably high probability for de-selecting metrics when the optimization algorithms generate random weights. The weights are clipped to $[0, 1]$ before computing the Q_l scores.

3.3 Objective M_2: Diversity of metrics

It is important to ensure that metrics suites contain a diverse set of metrics. As argued in [54], privacy metrics measure *privacy* only indirectly, and different privacy metrics can thus provide information about different aspects of privacy, such as the adversary's success rate, uncertainty, or estimation error. In addition, the interpretation of privacy metrics can be easier if results from different metrics are available [41].

We consider the diversity of privacy metrics in two main respects: the types of output measured, and the types of input used for the calculation. We follow [54] for the classification of privacy metrics into output categories and input types used. Our datasets use metrics belonging to seven of the eight output categories described in [54], namely uncertainty, error, information gain/loss, diversity, adversary's success, time, and indistinguishability. Each metric v_k belongs to exactly one output category $O(v_k)$.

Metrics may use one or more input types $I(v_k)$ to compute their measurement. The metrics in our study use three types of input data: the adversary's estimate, ground truth, and prior knowledge available to the adversary.

To define the diversity objective, we count how many different output categories are present in a metrics suite, and how many different input types are used for their computation.

$$M_2 = \Big| \bigcup_{v_k \in V, w_k > 0} O(v_k) + I(v_k) \Big|. \tag{3}$$

3.4 Objective M_3: Evenness

Evenness indicates how uniformly metric values are spread across their value range [59]. Metrics that score high on evenness enable easy comparisons within a scenario, for example between users in a graph, and are thus a useful tool for interpretation. Metrics with low evenness are undesirable because they make it hard to distinguish between alternatives that are very close.

The evenness $c_e(v)$ of a metric v is computed as the normalized Cramér-von Mises criterion, which indicates how well the distribution of metric values fits a uniform distribution.

Because the aim of easy within-scenario comparisons can be achieved by including a single highly even metric in a metrics suite, our objective is to maximize the maximum evenness of any metric in the suite (instead of maximizing the evenness of the entire metrics suite). This approach is similar to previous recommendations [59, 60].

$$M_3 = \max_{v \in V} c_e(v). \tag{4}$$

3.5 Objective M_4: Shared Value Range

The shared value range of a metric indicates to what extent its value range depends on the characteristics of the scenario, for example on the input graph or the adversary's de-anonymization algorithm. Metrics with a high shared value range allow for easy comparisons between different scenarios and are thus helpful for interpretation and result analysis.

The shared value range $c_s(v)$ is computed as the portion of the global value range that is used in each scenario. Similarly to evenness, a single metric with high shared value range is sufficient to achieve the aim of easy between-scenario comparisons. We therefore take a similar approach and define our objective so that only one metric with a high shared value range needs to be included in a metrics suite, that is, we define the shared value range of a metrics suite as the maximum individual shared value range across the metrics in the suite.

$$M_4 = \max_{v \in V} c_s(v). \tag{5}$$

3.6 Objective M_5: Number of metrics

Even though it is desirable to evaluate privacy using a diverse set of metrics, it is unrealistic to expect that real metrics suites can include all possible metrics: evaluating all metrics would be too time-consuming, make the analysis unnecessarily cumbersome, may lead to the use of undesirable metrics (e.g. in terms of monotonicity), or may result in the use of several similar metrics.

We therefore assume that shorter metrics suites are preferable, provided they perform equally well on the other objectives. To guide the optimization process, we design the objective function to have its highest value in a range between the lower and upper thresholds t_l and t_u. The objective function is zero when metrics suites contain fewer than t_l metrics to ensure diversity, and it decreases with the inverse of the number of metrics when the suite contains more than t_u metrics to ensure usability. In our optimization, we choose $t_l = 4$ to discourage generation of overly short metrics suites and $t_u = 8$ to keep them short enough to be evaluated by a human decision maker. There is a well-studied limitation of human capacity of processing information in short-term memory with seven +/- two objects that most of people can evaluate simultaneously [44].

We define W as the set of weights for all metrics in a suite and let $w = 0$ indicate that a metric is not selected for this metrics suite. Let $|w|$ denote the number of metrics with a weight $w > 0$, and t_l resp. t_u the thresholds that indicate our desired lower resp. upper limits for the number of metrics in a suite.

$$M_5(W) = \begin{cases} 0 & |w| \leq t_l \\ 1 & t_l < |w| < t_u \\ \frac{1}{|w|} & |w| \geq t_u. \end{cases} \tag{6}$$

We evaluate the effect of this objective on the quality of the resulting metrics suites in Section 6.2. Our evaluation will show how this objective affects the optimization results, and whether larger metrics suites have better properties, e.g. in terms of monotonicity.

3.7 Five-objective Optimization Problem

We finally integrate the five objectives introduced above into a five-objective optimization problem:

$$\begin{aligned} M_1(S) &\rightarrow \max; \\ M_2 &\rightarrow \max; \\ M_3 &\rightarrow \max; \\ M_4 &\rightarrow \max; \\ M_5(W) &\rightarrow \max. \end{aligned} \tag{7}$$

4 SOLUTION ALGORITHMS

To solve the optimization problem defined in Equation 7, we use evolutionary algorithms (EAs). Because different EAs have different solution approaches, and because EAs have so far not been used to solve the problem of metrics suite optimization, it is not known which EA will yield the best solutions. We have therefore evaluated several different types of EA to determine which EA performs best on our problem.

We have included both single-objective (SO) and multi-/many-objective (MO) algorithms for two reasons. First, we used SO algorithms to estimate the best value of the monotonicity objective that we could hope to obtain. Although we could have used the extreme solutions of multi-objective algorithms, we could run SO algorithms for longer (more generations and more replications) and thus get more precise estimates. Second, we used SO algorithms to motivate that the added effort of multi-objective optimization is indeed necessary because the metrics suites found by SO are much longer and therefore harder to apply in practice.

In our selection of MO algorithms, we aimed for a diverse set of algorithms, including dominance-based, decomposition-based, and indicator-based evolutionary algorithms as well as meta-heuristics. However, the final selection of specific SO and MO algorithms was also influenced by their availability in open source Python implementations. We include the detailed description of the algorithms and the performance measures for optimization algorithms in appendix A.

5 EXPERIMENTS

We apply the 16 optimization algorithms described in appendix A (4 single-objective algorithms and 12 multi-objective algorithms) to three previously published datasets. In this section, we first describe the datasets and how we encoded our multi-objective problem defined in Equation (7) for each dataset. Then, we explain which implementations we used for the optimization algorithms and how we configured each algorithm.

5.1 Datasets

We use one dataset for each of the three privacy domains introduced above. In particular, we use the datasets [55] from three papers, each of which systematically analyzed the performance of individual privacy metrics in its specific application domain. In general, each paper used a real-world data source to represent the user's data or behavior, then applied state-of-the-art adversary models or algorithms and finally used a range of privacy metrics to measure the privacy level. The graph privacy paper additionally considered defense mechanisms, i.e. graph anonymization techniques.

To evaluate the strength of privacy metrics, each paper relied on a ranked set of 6–9 adversary strengths, i.e. different configurations of the adversary model that represent stronger and weaker adversaries. For example, [60] used a sequence of six overlap percentages (60, 70, 80, 85, 90, 95), assuming that an adversary who knows a 95% overlapping graph is stronger than one who only knows a 60% overlapping graph. The resulting privacy levels should reflect this sequence and indicate the lowest privacy level for the strongest adversary. Our monotonicity objective is then computed based on the percentage of adversary strength levels that each privacy metric can rank correctly.

For *genomic privacy*, the user data consisted of publicly available genome data for a large number of individuals [53]. The paper constructed adversary models by assuming that all adversaries guess randomly, but that strong adversaries guess correctly with a higher probability than weak adversaries. The mean of the adversary's probability distribution was varied between 10–90% . In addition, stronger adversaries were constructed by giving prior knowledge about population-wide

frequencies of genomic variations, knowledge about relationships between genomic variations, and knowledge about the genome of a relative. The evaluation considered 24 privacy metrics belonging to five different output categories (see Table 1) and analyzed 102 scenarios.

For *graph privacy*, the paper used 11 publicly available social graphs as user data, anonymized with 12 existing graph anonymization algorithms [60]. Six adversary types were constructed based on six de-anonymization algorithms. Adversary strength was defined by adjusting the amount of prior knowledge available to the adversary: adversaries were given an auxiliary graph with 60–95% overlap with the anonymized graph, and 5–100 seed nodes with known mappings between the anonymized and auxiliary graphs. The paper evaluated 26 privacy metrics in five different output categories, considering 792 scenarios in total.

For *vehicular network privacy*, the user data was given by several real-world traffic traces, and the adversary used a state-of-the-art probabilistic tracking algorithm to track vehicles [59]. The adversary strength was adjusted based on the level of process noise (random motion in the system between observations) and measurement noise (uncertainty in measurement) that the adversary was subject to. The paper evaluated 41 privacy metrics from six different output categories in 15 scenarios.

5.2 Implementation

We used five Python libraries that provide implementations of evolutionary optimization algorithms. For all four single-objective optimization algorithms – CMA-ES, DE, PSO, and Subplex, we relied on their implementations in Pagmo2 [8]. For the multi-objective optimization algorithms, we used Pagmo2 2.15.0 [8] (MOEA/D, NSGA-II), pymoo 0.3.0 [9] (NSGA-II, NSGA-III), Platypus 1.0.2 [18] (CMA-ES, EpsMOEA, GDE3, IBEA, MOEA/D, NSGA-II, NSGA-III, OMOPSO, SMPSO, SPEA2), evoalgos 1.0 [56] (CMSA-ES), and DEAP 1.2.2 [16] (CMA-ES). Note that we used implementations from different libraries for some algorithms so that we could spot whether implementation differences had an influence on the optimization performance for our problem. We compare their results in Section B.4. Table 2 gives an overview of all algorithms, which implementations we use for each algorithm, and how we label each algorithm in the figures.

Our encoding of individuals relies on the fact that we are interested in finding a metrics suite defined by its set of weights w_k that optimizes the properties of the metrics suite. We therefore represent each individual as an array with $|V|$ elements, where V is the set of metrics available for each dataset. Each element in the array can take values in $[-1, 1]$. Even though the weights w_k need to be in $[0, 1]$ to define a valid metrics suite, having negative values in the individual arrays allows us to de-select metrics from metrics suites. This is important because the optimization algorithms choose values for individuals randomly. If individuals could take only values in $[0, 1]$, the optimization would de-select metrics only very infrequently and thus make it hard to achieve objective M_5 (number of metrics). The values of individual weights are clipped to $[0, 1]$ before evaluating the objective functions.

We implemented our five objectives as evaluation functions in Python, closely following their definitions in Section 3. The datasets provided metric values for each scenario and adversary strength level, which we use to compute monotonicity, as well as values for evenness and shared value range for each scenario.

5.3 Settings for optimization algorithms

We configured the settings for each optimization algorithm so that the algorithms used roughly comparable options, but also so that the settings were aligned with the best recommendations from the literature. To set the population size and number of function evaluations for both single and multi-objective formulations, we follow [10]. We set the population size to $N = 212$ for NSGA-III

Table 2. List of algorithms used in our study, with an indication whether the algorithm is single- or multi-objective, the open source library or libraries we used, and the label used in text and figures.

SO/MO	Algorithm	Library	Label
SO	Covariance matrix adaptation evolutionary strategy	Pagmo2	CMA-ES
	Differential evolution	Pagmo2	DE
	Particle swarm optimization	Pagmo2	PSO
	Subplex	Pagmo2	SBPLX
MO	Multi-objective covariance matrix adaptation	DEAP	CMA-ES-deap
		Platypus	CMA-ES-plat
	Covariance matrix self-adaptation evolution strategy	evoalgos	CMSA-ES-evo
	EpsMOEA	Platypus	EpsMOEA-plat
	Generalized Differential Evolution 3	Platypus	GDE-plat
	Indicator-Based Evolutionary Algorithm	Platypus	IBEA-plat
	Multi-objective Evolutionary Algorithm Based on Decomposition	Pagmo2	MOEA/D-gmo
		Platypus	MOEA/D-plat
	Non-dominated Sorting Genetic Algorithm II	Pagmo2	NSGA-II-gmo
		pymoo	NSGA-II-moo
		Platypus	NSGA-II-plat
	Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting	pymoo	NSGA-III-moo
		Platypus	NSGA-III-plat
	Optimized Multi-Objective Optimization PSO	Platypus	OMOPSO-plat
	Speed-constrained Multi-objective PSO	Platypus	SMPSO-plat
	Strength Pareto Evolutionary Algorithm 2	Platypus	SPEA2-plat

and $N = 210$ for all other algorithms and set the number of reference points for multi-objective algorithms to $|R| = 210$. The required number of function evaluations increases with the number of objective functions. We again follow [10] and set the number of function evaluations to $F = 50,400$ obtained as the product of population size $N = 210$ and number of generations $G = 240$.

For a fair comparison, the same mutation and recombination operators should be used for all algorithms. Here, we follow the settings in [10, 58] and use polynomial mutation with mutation probability $p_m = 1/n$, where n is number of decision variables, and mutation distribution index $\eta_m = 20$. We use simulated binary crossover with recombination probability $p_c = 1.0$ and distribution index $\eta_c = 20$. Algorithm specific parameters are set to their original recommendations. For instance, control parameters for GDE3 are set to $CR = 0.2$ and $F = 0.2$, and control parameter in NSGA-III is set to $T = 10$.

We initially performed 20 replications for each MOEA. Then, we computed the values for the three most important performance indicators (hypervolume, generational distance, inverted generational distance) and computed the relative error $\gamma(x)$ for each algorithm/dataset combination based on a 95% confidence interval, i.e. $\gamma(x) = \frac{h(x)}{|\bar{x}|}$ where $h(x)$ is the confidence interval half-width for confidence level 0.95 and x are observed values of the performance indicators for all replications for each combination of algorithm and dataset [31]. We then added replications until the relative error was below 5% for each algorithm/dataset combination, resulting in up to 90 replications in some cases.

5.4 Performance of optimization algorithms

Based on a detailed analysis of the performance of optimization algorithms (presented in appendix B), we excluded algorithms that have shown clear weaknesses in one or more indicators. Figure 1 shows the pairwise C metric for the remaining six implementations, all from the Platypus package. Based on this direct comparison, we can see that solutions from CMA-ES and OMOPSO are

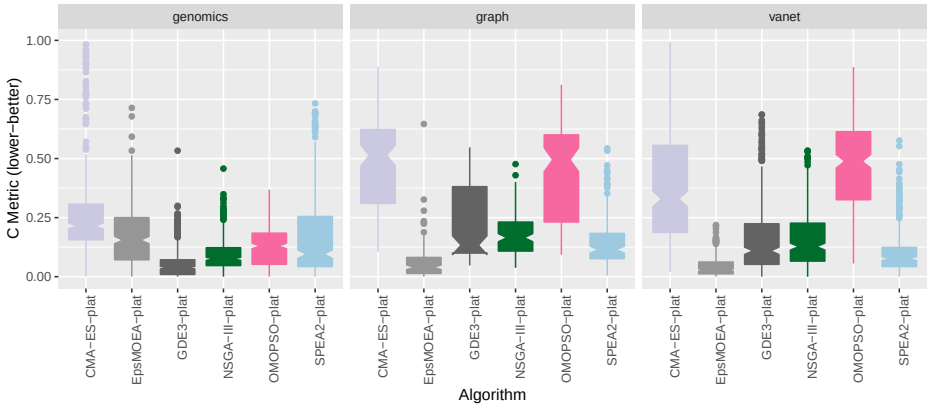
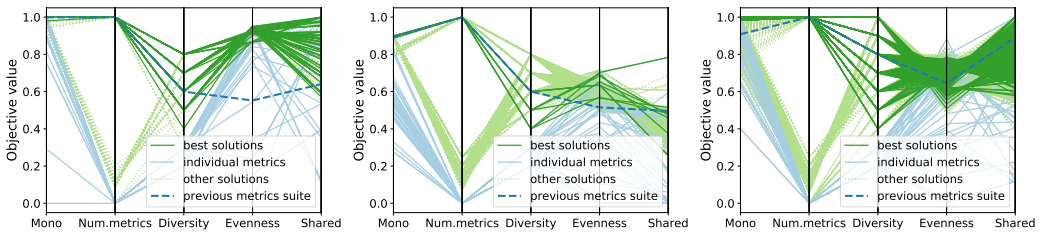


Fig. 1. C indicator for the six best algorithms from the above analysis.

dominated more often than solutions from the other algorithms. EpsMOEA and GDE3 are very good for some of the application domains but worse for others. We therefore conclude that NSGA-III and SPEA2 are the best overall algorithms for our problem. We hypothesize that NSGA-III performs well because it is designed for many-objective problems, and both NSGA-III and SPEA2 use an external archive to keep nondominated solutions which helps retain good solutions found early during the optimization.

6 RESULTS: PERFORMANCE OF METRICS SUITES

We now move on to analyze the metrics suites themselves, i.e. the solutions generated by the multi-objective optimization algorithms. Because we aim to find good metrics suites for each of the three application domains, for this analysis we are only interested in the characteristics of the metrics suites, but not in the algorithm that generated them. Therefore, we combine all solutions generated by all algorithms and narrow them down to select only the non-dominated solutions. In total, the optimization resulted in 47,679 non-dominated metrics suites for genomics privacy, 14,853 for vanet privacy, and 4,126 for graph privacy.



(a) Genomics (b) Graph (c) Vanet

Fig. 2. Parallel coordinates plots showing the objective values for all non-dominated solutions. Solutions with monotonicity better than 99% of the best monotonicity and between 4 and 8 metrics in each non-dominated metrics suite are highlighted in dark green.

6.1 Metrics suites vs. single metrics

Figure 2 shows the objective values for all non-dominated solutions in parallel coordinates plots [37]. To select a specific metrics suite from these solutions, the best metrics suites satisfy additional criteria: first, their monotonicity should be high, e.g. above 99% of the best monotonicity for a given dataset, and second, the number of metrics in the suite should be manageable, e.g. between 4 and 8. We have colored in dark green all solutions that fulfill these constraints.

Figure 2 also compares the objective values for the metrics suites (as given by the non-dominated solutions) to the objective values for all metrics individually (by light blue lines), and the objective values for metrics suites previously proposed in [53, 59, 60] (by blue dashed lines). It is clear that the individual metrics are worse especially on the number of metrics and diversity objectives, but many individual metrics are also worse on monotonicity, evenness, and shared value range. For all three datasets, the previously proposed metrics suites perform well on the number of metrics objective, but fall short on most of the other objectives. Even though the optimization generated some metrics suites that are worse on single objectives (e.g., lower monotonicity than the individual metric with the highest monotonicity), it did not generate solutions that are worse on all objectives. As a result, the metrics suites computed by solving our optimization problem perform better than the previously proposed metrics suites and the individual metrics.

6.2 Number of metrics in a suite

In this section, we answer two questions: first, how does the quality of a metrics suite depend on the number of metrics in it (is it worth the additional effort to use an eight-metrics suite rather than a five-metrics suite?), and second, how does the presence of the number-of-metrics objective influence the optimization process (does the optimization result in less-monotonic metrics suites than it would otherwise?).

6.2.1 Quality of metrics suites. Figure 3 shows how the objective values of the four other objectives evolve when the number of metrics in a metrics suite increases. The three datasets are indicated by different colors, and for each dataset we present two cases: metrics suites with a monotonicity of at least 99% of the best monotonicity for the dataset (solid lines), and all other metrics suites (dashed lines). Some lines in the plot are shorter than others because we removed all *dominated* solutions from the set of metrics suites. As a result, for example, the graph dataset does not have solutions with one or two metrics in a suite.

For monotonicity (see Figure 3a), we can see that for the genomics and vanet datasets, the best (100%) monotonicity can be achieved with as few as 2 metrics in a metrics suite. For the graph dataset, the best monotonicity (90.51%) is achieved with 16 metrics. With 90.28% monotonicity, 6-metrics suites are already very close to the best value.

As expected, the diversity objective (Figure 3b), indicating how diverse in terms of inputs and outputs a metrics suite is, increases with the number of metrics in a suite. This increase is mostly independent from the monotonicity of a metrics suite, i.e. the most monotonic suites and the other suites have similar values for diversity.

Figure 3c shows that the evenness objective decreases with the number of metrics. This indicates that a trade-off has to be made between the number of metrics and evenness. Interestingly, for the vanet dataset, and for metrics suites with a small number of metrics, those suites with the highest monotonicity have lower scores on evenness than the less-monotonic suites.

Similar to evenness, the shared value range objective (Figure 3d) also decreases with the number of metrics. We note that metrics suites with the best monotonicity consistently have a lower shared value range for the graph and genomics datasets compared to less-monotonic suites.

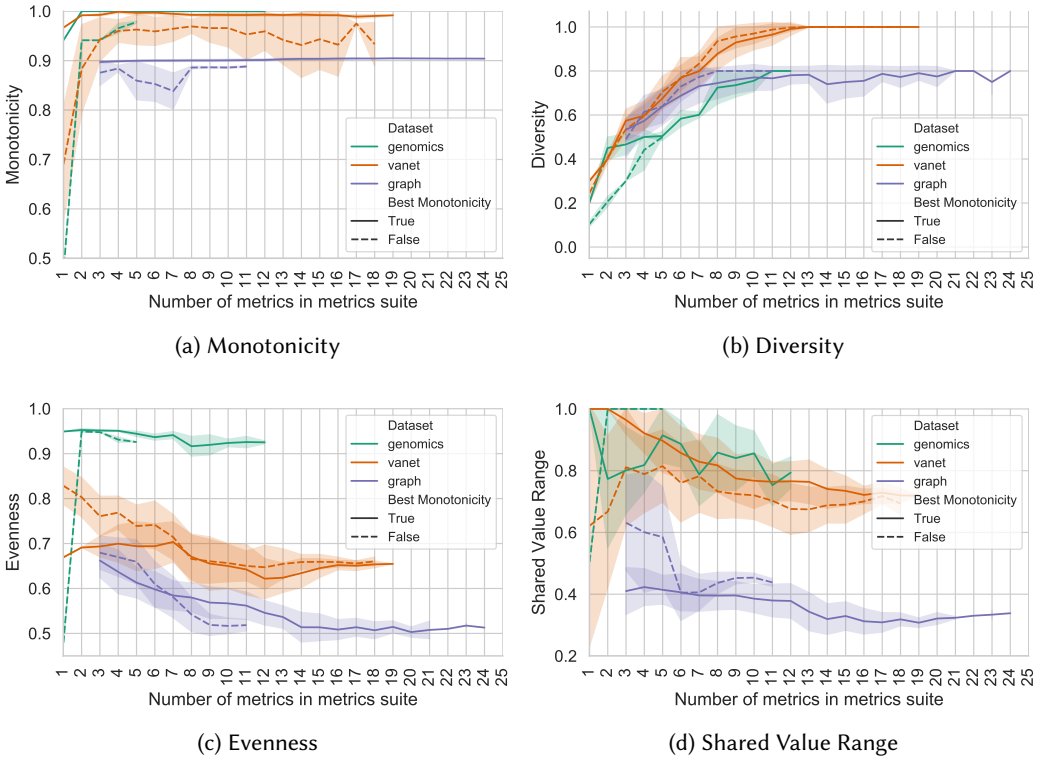


Fig. 3. Monotonicity, Diversity, Evenness, and Shared Value Range depending on the number of metrics in the metrics suite. Solid lines indicate metrics suites whose monotonicity is among the top 1%, whereas dashed lines indicate all other metrics suites. Shaded areas indicate the standard deviation, computed across the objective values for all metrics suites with the given number of metrics.

The fact that both evenness and shared value range decrease with a higher number of metrics in a suite across all datasets indicates that shorter metrics suites can be advantageous. This represents a trade-off between evenness and shared value range on the one hand and the diversity of the metrics suite on the other.

Overall, selecting the best size for a metrics suite depends on the dataset and the desired value for the diversity objective. For most datasets, we see a sweet spot at 5 or 6 metrics. Six-metrics suites achieve the close to the best monotonicity, diversity values of 0.6 or higher, and reasonable values for evenness and shared value range for all datasets.

6.2.2 Quality of optimization result. To evaluate the usefulness of the number-of-metrics objective, we ran experiments without this objective, i.e. optimizing only for the four other objectives. The resulting metrics suites are 10-30% longer on average and the average monotonicity changes by -2% to +6%. The best monotonicity achieved by the metrics suites does not change, but the shortest suite that achieves the best monotonicity is up to 37% longer.

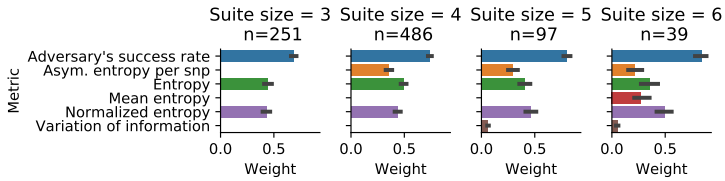
This result can be explained by how the optimization uses objective functions and the non-dominance criterion. The optimization does not exclude metrics suites that are longer than our "ideal" range. For example, Figure 12 shows metrics suites with up to 24 metrics. Instead, the value

of the number-of-metrics objective function decreases with the length of the suite, which guides the optimizer towards shorter suites. This illustrates the non-dominance criterion: if a metrics suite has better monotonicity but worse number-of-metrics than another suite, neither suite is dominated and both are kept in the solution set. The decision which non-dominated metrics suite should be selected is then up to the researcher's priorities.

With all other objective values equal, we can say that including the number-of-metrics objective results in shorter metrics suites, which are generally preferable.

6.3 Metrics suites for genomics privacy

We will now present the composition of specific metrics suites for each dataset. To allow researchers to select custom metrics suites depending on their needs, we make the full dataset of non-dominated metrics suites, including their composition and objective values, available at [55].



(a) Average weight for each metric included in the best metrics suites for the genomics dataset, by size of the suite. *n* indicates how many solutions, or metrics suites, of each size were found by the optimization.

Category	Number of metrics/suite	6	5	4	3	8 (prior)
Success	Adversary's success rate	0.94	0.74	0.80	0.47	1.0
Uncertainty	Asym. entropy per snp	0.05	0.005	0.14		
Uncertainty	Entropy	0.64	0.34	0.17	0.70	1.0
Uncertainty	Mean entropy	0.69				
Uncertainty	Normalized entropy	0.46	0.64	0.14	0.71	
Information gain	Variation of information	0.009	0.005			
Information gain	Amount of leaked information					1.0
Error	Expected error					1.0
Uncertainty	Genomic privacy					1.0
Information gain	Information surprisal					1.0
Error	Mean squared error					1.0
Information gain	Relative entropy					1.0
	Monotonicity	1.0	1.0	1.0	1.0	0.998
	Evenness	0.93	0.92	0.94	0.94	0.55
	Shared value range	0.97	0.99	0.99	0.99	0.64
	Diversity	0.6	0.6	0.5	0.4	0.6

(b) Weights and objective values for example metrics suites for genomics dataset. Each column is one metrics suite.

Fig. 4. Metrics suite characteristics for genomic privacy.

Figure 4a shows the average weight for each metric included in the most-monotonic metrics suites for genomics, depending on the size of the suite. The figure also indicates how many metrics suites were found in each case. We can see that at each increase in the size of the metrics suite one

metric is added and all previous metrics are retained. We can also see that the adversary's success rate is the most highly weighted single metric, and that larger metrics suites mainly add different versions of the entropy metric, such as normalized entropy, mean entropy, and asymmetric entropy.

We selected four example metrics suites and show their weights and objective values in Figure 4b. Each weight is shown with a value range of $[0, 1]$ to allow easier comparison of their relative importance, regardless of the size of the metrics suite. The optimization uses normalized metric weights so that all weights in a suite add up to 1. We note that the weights of specific metrics suites, as shown in Figure 4b, do not necessarily correspond to the *average* weights over all metrics suites, as shown in Figure 4a. We can see that all metrics suites have 100% monotonicity. As expected from the discussion in the previous section, larger metrics suites have slightly smaller values for evenness and shared value range, but higher values for diversity. The comparison with a previously suggested metrics suite [53] in the rightmost column shows that the optimized metrics suites perform equally well on monotonicity, but much better on evenness and shared value range with a smaller number of metrics.

6.4 Metrics suites for graph privacy

Compared with the genomics dataset, the selection of metrics in the top-performing metrics suites for the graph dataset (Figure 5a) is much more diverse. For example, the 222 non-dominated 5-metrics suites use a total of 18 different metrics. The adversary's success rate (including its variant, the overall success rate) is still the most highly weighted single metric.

We show four example metrics suites for graph privacy in Figure 5b. In contrast to genomics, monotonicity is increasing with the size of the metrics suite. Monotonicity of the best individual metric for graph privacy is 82.3% (amount of leaked information) [60]. The example 6-metrics suite in Figure 5b increases this monotonicity to 90.3% (+ 8%), which illustrates the benefit of using a well-optimized metrics suite. Compared to the metrics suite from prior work [60] (rightmost column), the optimized 6-metrics suite has higher monotonicity, evenness, shared value range, and diversity.

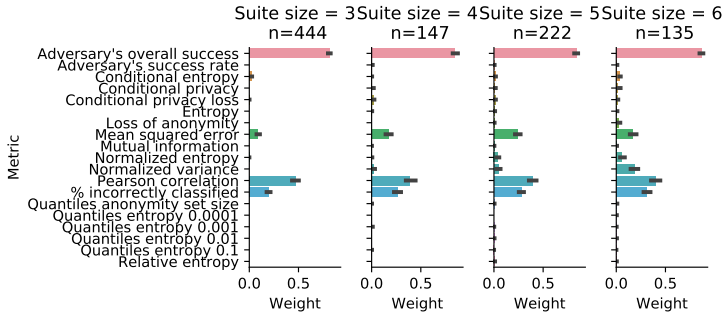
6.5 Metrics suites for vanet privacy

Finally, Figure 6a shows the average weights for non-dominated vanet privacy metrics suites. In contrast to genomics and graph privacy metrics suites, the adversary's success rate is only a minor component for vanet privacy metrics suites. Instead, normalized entropy is the most highly weighted individual metric.

The example metrics suites in Figure 6b show that monotonicity is close or equal to 100% in all cases. As expected, the diversity objective increases with the size of the metrics suite. The values for evenness and shared value range vary and depend strongly on the specific metrics suite that is selected. This illustrates that a trade-off between diversity, evenness, and shared value range has to be made depending on the specific requirements for privacy measurement. Compared to a previously suggested combination of metrics [59], the optimized metrics suites have much higher monotonicity (+8%). The optimized 6-metrics suite performs similar to prior work on evenness, shared value range, and diversity, while the other optimized suites perform better on one objective but worse on the others, showing the trade-off between evenness, shared value range, and diversity.

7 DISCUSSION

Metrics suites allow to rank systems by their privacy level, for example systems that differ in the privacy-enhancing technology they use. Because these rankings incorporate different aspects of privacy through the different privacy metrics that make up the metrics suite, they offer a more comprehensive view than rankings based on individual metrics alone.



(a) Average weight for each metric included in the best metrics suites for the graph dataset, by size of the suite. *n* indicates how many solutions, or metrics suites, of each size were found by the optimization.

Category	Number of metrics/suite	6	5	4	3	7 (prior)
Success	Adversary's overall success	0.96	0.87	0.93	0.97	1.0
Success	Adversary's success rate					1.0
Uncertainty	Conditional privacy	0.19				
Information gain	Pearson correlation	0.23	0.24	0.23	0.95	1.0
Error	% incorrectly classified	0.53	0.40	0.50		
Information gain	Loss of anonymity	0.22				
Error	Mean squared error		0.49		0.47	
Uncertainty	Normalized entropy		0.04	0.2		
Similarity	Normalized variance	0.52				1.0
Error	Absolute error					1.0
Information gain	Amount of leaked information					1.0
Error	Incorrectness					1.0
	Monotonicity	0.903	0.902	0.90	0.896	0.892
	Evenness	0.62	0.60	0.60	0.68	0.44
	Shared value range	0.27	0.43	0.43	0.47	0.17
	Diversity	0.8	0.6	0.6	0.5	0.6

(b) Weights and objective values for example metrics suites for graph dataset.

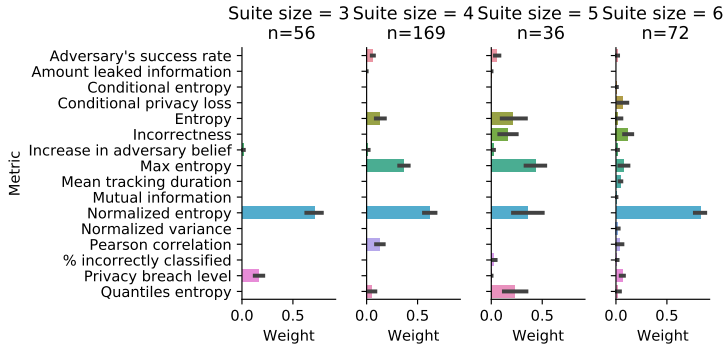
Fig. 5. Metrics suite characteristics for graph privacy.

The benefits of using *optimization* to determine the selection and weighting of metrics in a suite are that optimization allows researchers to select a metrics suite that fits their needs when multiple objectives cannot be satisfied simultaneously (e.g. evenness and shared value range), and that optimization can give confidence that the selected metrics suite is of high quality in terms of the optimization objectives.

In this section, we first show how metrics suites can be applied in practice, and then discuss some limitations of our approach.

7.1 Application example: ranking of graph anonymization algorithms

To illustrate the benefits of optimized metrics suites in a practical example, we now use the graph privacy dataset to answer the question, *which graph anonymization algorithm performs best when tested against different adversaries and on different graphs?* We will compare how the answer to this question differs when individual metrics are used as opposed to metrics suites.



(a) Average weight for each metric included in the best metrics suites for the vanet dataset, by size of the suite. *n* indicates how many solutions, or metrics suites, of each size were found by the optimization.

Category	Number of metrics/suite	6	5	4	3	5 (prior)
Success	Adversary's success rate		0.02	0.008		
Error	Incorrectness	0.84				1.0
Information gain	Increase in adversary belief	0.008	0.01	0.01	0.02	
Time	Mean tracking duration	0.14				
Uncertainty	Normalized entropy	0.78	0.76	0.59	0.88	1.0
Information gain	Pearson correlation	0.46		0.53		
Error	% incorrectly classified		0.02			
Success	Privacy breach level	0.61			0.24	1.0
Information gain	Amount of leaked information		0.02			
Information gain	Conditional privacy loss					1.0
Time	Time to confusion					1.0
	Monotonicity	0.99	0.99	1.0	0.99	0.908
	Evenness	0.69	0.77	0.72	0.68	0.65
	Shared value range	0.89	0.68	0.89	0.99	0.89
	Diversity	0.8	0.7	0.6	0.5	0.8

(b) Weights and objective values for example metrics suites for vanet dataset.

Fig. 6. Metrics suite characteristics for vanet privacy.

The graph privacy dataset provides sufficient data to answer this question: the experiment evaluated 12 state-of-the-art anonymization algorithms against six de-anonymization algorithms on 11 different graphs [60]. The dataset uses several different privacy metrics that we can use to rank the anonymization algorithms. The five left-hand columns in Table 3 show the ranking generated by five individual privacy metrics. It is clear that each metric generates a different ranking, and there is little agreement on the relative merits of each anonymization algorithm. In contrast, the five columns on the right show rankings generated by the metrics suites defined in Figure 5b[1]. Here it is clear that the metrics suites are largely consistent in the rankings they generate, and in particular

[1]We note that the prior metrics suite from [60] was already quite effective at ranking algorithms. This is because the metrics suite in that paper was hand-optimized by tuning the selection and weighting of metrics to achieve high monotonicity. Our results show that the optimization method presented in this paper can improve the metrics suite compared to hand-optimization, without the effort of manual tuning.

all metrics suites agree on the best and worst algorithm, and on the three algorithms that make up the top-3. In addition, the three algorithms based on differential privacy are ranked highest, which is consistent with expectation based on differential privacy literature, and the baseline algorithm ID removal, which is known to provide little to no privacy protection, is ranked near the bottom[2]. We also observe that the three right-most metrics suites (consisting of 4, 5, and 6 metrics), which have higher monotonicity than the other metrics suites, are most consistent in their rankings. The last row of Table 3 shows the Spearman rank-correlation coefficient comparing the rankings of the metric/metrics suite in each column with the 6-metrics suite. We can see that only one of the individual metrics, the adversary's overall success rate, is highly correlated with the best metrics suites in this example. However, while the adversary's overall success rate generates a consistent ranking of anonymization algorithms, it does not perform well on the evenness and shared value range criteria which are needed for more detailed analyses of privacy levels. By combining different metrics, the metrics suites can compensate for this weakness.

Based on the rankings generated by, for example, the 6-metrics suite, we can conclude that Weighted privacy integrated queries based DP is the graph anonymization algorithm that performs best, followed by Hierarchical random graph based DP. This application example shows how metrics suites can help ensure a consistent ranking of systems or algorithms by the privacy level they provide. Our optimization-based approach ensures that selection and weighting of metrics is consistent and objective, and does not require luck or subjective judgments. Individual metrics can still be interpreted separately if needed, e.g., for comparisons of individuals within scenarios or for ease of comparison with prior work.

7.2 Limitations of our approach

Although we have shown that our approach improves privacy measurement by optimizing the selection and weighting of privacy metrics, it has a number of limitations. Our approach does not attempt to unify privacy measurement across domains. Rather, our approach can be used to improve the measurement of privacy in individual, well-defined scenarios. Even though privacy metrics suites may differ between domains, we argue that it is important to measure multiple aspects of privacy, and to measure it in a consistent way. Our approach to use optimization to determine the composition of metrics suites can help address these issues. Drawbacks of our approach are that it requires additional work before privacy can be measured, that transferability of its results to new domains is limited, and that it does not resolve the issue of having to know which privacy metrics are effective and applicable in a specific application. These are discussed in more detail below.

7.2.1 Steps for practical application. To apply our approach in the context of a research project to evaluate new PETs or existing PETs in a new scenario, the evaluation workflow needs four additional steps. The prerequisite for these steps is that the scenario has been defined, including considerations regarding what data needs to be protected and what the adversary's capabilities are.

The additional steps are then: First, to define a sequence of adversary strength levels. In our experience, this is straightforward after the scenario has been defined. Second, to collect a list of candidate privacy metrics that may be suitable in the scenario. Third, to evaluate the candidate privacy metrics for each adversary strength level. This data is needed as input for the optimization. Fourth, to perform the optimization using one or more of the optimization algorithms described in this paper[3] and choose among the optimized metrics suites. The metrics suite can be selected

[2]Note that differential privacy *as a metric* was omitted from the three datasets that we used, but differential privacy *as a mechanism* is present in the graph dataset. The reason that differential privacy as a metric was omitted is that it is not possible to compute an ϵ score for a scenario that does not use differential privacy.

[3]Solutions from multiple algorithms can be combined by retaining non-dominated solutions, as we have done in Section 7.

Table 3. Ranking of 12 graph anonymization algorithms by their privacy level. Each cell indicates the algorithm's rank when the ranking was generated using the column's metric or metrics suite. The underlying experiment evaluated each anonymization algorithm on 11 graphs and against six different de-anonymization algorithms [60]. The individual metrics have been chosen because they make up the 3-, 4-, and 5-metrics suites (see Figure 5b).

Anonymizer	Adversary's overall success	Pearson correlation	Mean squared error	% incorrectly classified	Normalized entropy	Prior metrics suite [60]	3-metrics suite	4-metrics suite	5-metrics suite	6-metrics suite
Weighted privacy integrated queries based DP	1	2	10	6	10	1	1	1	1	1
Hierarchical random graph based DP	2	4	9	1	1	2	3	2	2	2
dk-series based differential privacy (DP)	3	3	2	11	11	3	2	3	3	3
Random walk	4	1	1	3	5	4	4	4	4	4
Univariate micro-aggregation (UMGA)	5	9	11	2	2	7	9	5	5	5
k-Degree anonymity	6	6	4	7	8	5	5	6	6	6
Bounded t-means	8	5	3	5	9	6	6	7	7	7
Graph k-anonymzation	7	10	8	4	3	10	10	8	9	8
Switch	9	7	5	9	7	8	7	9	8	9
ID removal	10	8	6	10	4	9	8	10	10	10
k-Symmetry anonymity	11	11	7	8	6	11	11	11	11	11
Spearman rank-correlation with 6-metrics suite	0.99	0.79	-0.11	0.41	-0.15	0.95	0.85	1.0	0.99	1.0

to fit the scenario. For example, if within-scenario comparisons are needed, a metrics suite with high evenness is most suitable, and a metrics suite with high shared value range is best if between-scenario comparisons are needed. Finally, the chosen metrics suite can be used to evaluate privacy levels.

These steps show that the improvements in privacy measurement that can be achieved with optimized metrics suites comes at a price. Future work may be able to focus on simplifying this approach to make it easier to apply.

7.2.2 Efficacy of existing privacy metrics. Our approach relies on a list of effective privacy metrics from which to construct metrics suites. In this paper, we have selected datasets from prior work that contained a wide range of privacy metrics, not all of which may be conceptually sound choices in every application domain.

In a practical application, however, is preferable to remove privacy metrics from the list which have known weaknesses in the intended application scenario, and to ensure that the metrics on the list capture vulnerabilities to specific attacks if needed. For example, it has been shown that max-entropy and anonymity set size overestimate privacy because they do not take into account that an adversary may consider some users more likely than others [12]. However, this weakness has not stopped the metric from being used in many research papers. Overall, we argue that it is important to consider strength and weaknesses of privacy metrics before applying our optimization-based method, but since our method leads to a combination of different metrics, it has the potential of mitigating the weaknesses of individual metrics to some extent.

7.2.3 Availability of data. The optimization relies on input data that represent the privacy levels measured by different privacy metrics, arranged in sequences for which it is known how privacy levels *should* evolve, e.g. a sequence can be given by an ordered sequence of adversary strength levels where a stronger adversary is expected to result in lower privacy levels. In this paper, we have used empirical datasets from prior work. However, if no existing datasets are available, it can be difficult and time-consuming to generate this data because appropriate adversary strength levels need to be defined and values for privacy metrics need to be computed, possibly for many combinations of user and adversary behavior.

7.2.4 Transfer of metrics suites to new settings. We have computed optimized metrics suites for three application domains: genomic privacy, graph privacy, and vehicular communications privacy. For two of these domains (genomic and graph privacy), the datasets we used included data for different adversary models. Based on the observed differences between metrics suites, we can judge to what extent optimized metrics suites can be used in other settings.

Prior work showed that the strength of individual metrics is largely consistent within application domains (e.g., genomics [53], vehicular communications [59], and social graphs [60]) and across algorithms the adversary uses [53, 60]. However, in combination these results show that different metrics are strong in different application domains. In addition, these papers did not study whether metric strength is consistent across different adversary goals, such as inference of a user's location vs. correct tracking of a user's trajectory.

Combining our results with prior work, we conclude that metric strength is not necessarily consistent between application domains, and may not be consistent for different adversary goals in the same domain. As a result, we believe that new optimization of metrics suites is needed for new application domains, new adversary goals, and newly introduced metrics.

8 CONCLUSION

In this paper, we have used multi-objective evolutionary optimization to generate privacy metrics suites that simultaneously fulfill five objectives: monotonicity, diversity, evenness, shared value range, and number of metrics. We evaluated which optimization algorithm performs best on our problem and find that NSGA-III and SPEA2 generate the best solution sets across our application domains.

Regarding the quality of privacy measurement, we find that metrics suites generated by the optimization increase the monotonicity of privacy measurement and at the same time ensure that the metrics suites exhibit other desirable properties such as diversity, evenness, shared value range, and number of metrics. Our optimization generates specific metrics suites, i.e. selections of privacy metrics and their weights, that researchers can use to evaluate new privacy technologies. We have shown in an application example that optimized metrics suites rank privacy technologies more consistently than individual metrics. The optimization thus removes uncertainty from the selection of privacy metrics.

REFERENCES

[1] James Alexander and Jonathan Smith. 2003. Engineering Privacy in Public: Confounding Face Recognition. In *Proc. 3rd Int. Workshop on Privacy Enhancing Technologies (PET 2003) (LNCS 2760)*. Springer, Dresden, Germany, 88–106.

[2] Erman Ayday, Jean Louis Raisaro, Jean-Pierre Hubaux, and Jacques Rougemont. 2013. Protecting and Evaluating Genomic Privacy in Medical Tests and Personalized Medicine. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society (WPES '13)*. ACM, New York, NY, USA, 95–106. https://doi.org/10.1145/2517840.2517843

[3] Thomas Bäck. 2000. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Taylor & Francis Ltd.

[4] Thomas Bäck, D. B. Fogel, and Z. Michalewicz. 2000. *Evolutionary Computation 1: Basic Algorithms and Operators* (1st ed.). CRC Press.

[5] Vitor Basto-Fernandes, Iryna Yevseyeva, André Deutz, and Michael Emmerich. 2017. A Survey of Diversity Oriented Optimization: Problems, Indicators, and Algorithms. In *EVOLVE – A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation VII*. Springer International Publishing, Cham, 3–23. https://doi.org/10.1007/978-3-319-49325-1_1

[6] Elisa Bertino, Dan Lin, and Wei Jiang. 2008. A Survey of Quantification of Privacy Preserving Data Mining Algorithms. In *Privacy-Preserving Data Mining: Models and Algorithms*. Number 34 in Advances in Database Systems. Springer, Chapter 8, 183–205.

[7] Hans-Georg Beyer and Bernhard Sendhoff. 2008. Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy –. In *Parallel Problem Solving from Nature – PPSN X (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, 123–132.

[8] Francesco Biscani, Dario Izzo, Wenzel Jakob, Marcus Märtens, Alessio Mereta, Cord Kaldemeyer, Sergey Lyskov, Sylvain Corlay, Benjamin Pritchard, Kishan Manani, Johan Mabille, Tomasz Miąsko, Axel Huebl, jakirkham, hulucc, polygon, Zihao Fu, The Gitter Badger, Merlin Nimier-David, Luka Čehovin Zajc, Jonas Adler, John Travers, Jeongseok Lee, Jakob Jordan, Ivan Smirnov, Huu Nguyen, Felipe Lema, Erik O'Leary, and Andrea Mambrini. 2019. Esa/Pagmo2: Pagmo 2.10. https://zenodo.org/record/2529931/export/hx. https://doi.org/10.5281/zenodo.2529931

[9] Julian Blank and Kalyanmoy Deb. 2019. Pymoo - Multi-Objective Optimization in Python. https://pymoo.org.

[10] Kalyanmoy Deb and Himanshu Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (2014), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197. https://doi.org/10.1109/4235.996017

[12] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2003. Towards Measuring Anonymity. In *Privacy Enhancing Technologies*, Roger Dingledine and Paul Syverson (Eds.). Number 2482 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 54–68.

[13] Claudia Diaz, Carmela Troncoso, and George Danezis. 2007. Does Additional Information Always Reduce Anonymity?. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES '07)*. ACM, Alexandria, VA, USA, 72–75. https://doi.org/10.1145/1314333.1314347

[14] David Eckhoff and Christoph Sommer. 2018. Readjusting the Privacy Goals in Vehicular Ad-Hoc Networks: A Safety-Preserving Solution Using Non-Overlapping Time-Slotted Pseudonym Pools. *Computer Communications* 122 (June 2018), 118–128. https://doi.org/10.1016/j.comcom.2018.03.006

[15] Michael Emmerich, Nicola Beume, and Boris Naujoks. 2005. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In *Evolutionary Multi-Criterion Optimization*. Springer Berlin Heidelberg, 62–76.

[16] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research* 13 (2012), 2171–2175. Issue Jul.

[17] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading: MA.

[18] David Hadka. 2019. A Free and Open Source Python Library for Multiobjective Optimization: Project-Platypus/Platypus. https://github.com/Project-Platypus/Platypus.

[19] M. P. Hansen and A. Jaszkiewicz. 1998. *Evaluating the Quality of Approximations to the Non-Dominated Set*. Technical Report IMM Technical Report IMMREP1998-7. Technical University of Denmark.

[20] N. Hansen and A. Ostermeier. 1996. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*. 312–317. https://doi.org/10.1109/ICEC.1996.542381

[21] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. https://doi.org/10.1162/106365601750190398

[22] Daojing He, S. Chan, and M. Guizani. 2015. Privacy and Incentive Mechanisms in People-Centric Sensing Networks. *IEEE Communications Magazine* 53, 10 (Oct. 2015), 200–206. https://doi.org/10.1109/MCOM.2015.7295484

[23] Christian Igel, Nikolaus Hansen, and Stefan Roth. 2007. Covariance Matrix Adaptation for Multi-Objective Optimization. *Evolutionary Computation* 15, 1 (2007), 1–28. https://doi.org/10.1162/evco.2007.15.1.1

[24] Himanshu Jain and Kalyanmoy Deb. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation* 18, 4 (Aug. 2014), 602–622. https://doi.org/10.1109/TEVC.2013.2281534

[25] Georgios Kalogridis, Costas Efthymiou, Stojan Z. Denic, Tim A. Lewis, and Rafael Cepeda. 2010. Privacy for Smart Meters: Towards Undetectable Appliance Load Signatures. In *First International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, Gaithersburg, MD, USA, 232–237.

[26] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. Multidimensional Knapsack Problems. In *Knapsack Problems*. Springer Berlin Heidelberg, 235–283. https://doi.org/10.1007/978-3-540-24777-7_9

[27] J. Kennedy and R. Eberhart. 1995. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[28] S. Kukkonen and J. Lampinen. 2005. GDE3: The Third Evolution Step of Generalized Differential Evolution. In *2005 IEEE Congress on Evolutionary Computation* (Edinburgh, Scotland, UK), Vol. 1. IEEE, 443–450. https://doi.org/10.1109/CEC.2005.1554717

[29] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. 2002. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation* 10, 3 (Sept. 2002), 263–282. https://doi.org/10.1162/106365602760234108

[30] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. 2001. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* TIK-Report 103. Eidgenössische Technische Hochschule Zürich (ETH).

[31] Averill M. Law and W. David Kelton. 2000. *Simulation Modelling and Analysis* (3rd edition ed.). McGraw-Hill Education, Boston.

[32] Longmei Li, Iryna Yevseyeva, Vitor Basto-Fernandes, Heike Trautmann, Ning Jing, and Michael Emmerich. 2017. Building and Using an Ontology of Preference-Based Multiobjective Evolutionary Algorithms. In *Evolutionary Multi-Criterion Optimization*. Vol. 10173. Springer International Publishing, Cham, 406–421. https://doi.org/10.1007/978-3-319-54157-0_28

[33] Miqing Li and Xin Yao. 2019. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *ACM Comput. Surv.* 52, 2 (March 2019), 26:1–26:38. https://doi.org/10.1145/3300148

[34] Silvano Martello, David Pisinger, and Paolo Toth. 2000. New Trends in Exact Algorithms for the 0–1 Knapsack Problem. *European Journal of Operational Research* 123, 2 (2000), 325–332. https://doi.org/10.1016/S0377-2217(99)00260-X

[35] Silvano Martello and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Inc.

[36] Kaisa Miettinen. 1998. *Nonlinear Multiobjective Optimization.* Springer Science & Business Media.

[37] Rida E. Moustafa. 2011. Parallel Coordinate and Parallel Coordinate Density Plots. *Wiley Interdisciplinary Reviews: Computational Statistics* 3, 2 (2011), 134–148. https://doi.org/10.1002/wics.145

[38] Arvind Narayanan and Vitaly Shmatikov. 2009. De-Anonymizing Social Networks. In *IEEE Symposium on Security and Privacy*. IEEE, Oakland, CA, USA, 173–187. https://doi.org/10.1109/SP.2009.22

[39] A.J. Nebro, J.J. Durillo, J. Garcia-Nieto, C.A. Coello Coello, F. Luna, and E. Alba. 2009. SMPSO: A New PSO-Based Metaheuristic for Multi-Objective Optimization. In *2009 IEEE Symposium on Computational Intelligence in Milti-Criteria Decision-Making* (Nashville, TN, USA). IEEE, 66–73. https://doi.org/10.1109/MCDM.2009.4938830

[40] J. A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7, 4 (1965), 308–313. https://doi.org/10.1093/comjnl/7.4.308

[41] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. 2017. Back to the Drawing Board: Revisiting the Design of Optimal Location Privacy-Preserving Mechanisms. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, Dallas, Texas, USA, 1959–1972. https://doi.org/10.1145/3133956.3134004

[42] Thomas Harvey Rowan. 1990. *Functional Stability Analysis Of Numerical Algorithms.* Ph.D. Dissertation. University of Texas at Austin.

[43] Jason R. Schott. 1995. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization.* Technical Report AFIT/CI/CIA-95-039. Air Force Inst of Tech Wright-Patterson AFB OH.

[44] Richard M. Shiffrin and Robert M. Nosofsky. 1994. Seven plus or Minus Two: A Commentary on Capacity Limitations. *Psychological Review* 101, 2 (1994), 357–361. https://doi.org/10.1037/0033-295X.101.2.357

[45] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying Location Privacy. In *IEEE Symposium on Security and Privacy (S&P)* (Oakland, CA, USA). IEEE, 247–262. https://doi.org/10.1109/SP.2011.18

[46] Margarita Reyes Sierra and Carlos A. Coello Coello. 2005. Improving PSO-Based Multi-Objective Optimization Using Crowding, Mutation and ∈-Dominance. In *Evolutionary Multi-Criterion Optimization (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, 505–519.

[47] Rainer Storn and Kenneth Price. 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359. https://doi.org/10.1023/A:1008202821328

[48] Paul Syverson. 2013. Why I'm Not an Entropist. In *Proc. 17th Int. Workshop on Security Protocols (LNCS 7028)*. Springer, Cambridge, UK, 213–230.

[49] El-Ghazali Talbi. 2009. *Metaheuristics: From Design to Implementation.* John Wiley & Sons.

[50] Chris Tofallis. 2014. Add or Multiply? A Tutorial on Ranking and Choosing with Multiple Criteria. *INFORMS Transactions on Education* 14, 3 (2014), 109–119. https://doi.org/10.1287/ited.2013.0124

[51] Evangelos Triantaphyllou. 2000. *Multi-Criteria Decision Making Methods: A Comparative Study.* Springer US.

[52] David A Van Veldhuizen and Gary B Lamont. 1998. Evolutionary Computation and Convergence to a Pareto Front. In *Late Breaking Papers at the Genetic Programming 1998 Conference*. 221–228.

[53] Isabel Wagner. 2017. Evaluating the Strength of Genomic Privacy Metrics. *ACM Trans. Priv. Secur.* 20, 1 (Jan. 2017), 2:1–2:34. https://doi.org/10.1145/3020003
[54] Isabel Wagner and David Eckhoff. 2018. Technical Privacy Metrics: A Systematic Survey. *ACM Comput. Surv.* 51, 3 (2018), 57:1–57:38. https://doi.org/10.1145/3168389
[55] Isabel Wagner and Iryna Yevseyeva. 2020. Privacy Metrics Suites for Genomic Privacy, Vehicular Communications Privacy, and Graph Privacy (Version 1.0.0) [Data Set]. https://doi.org/10.5281/zenodo.3350563.
[56] Simon Wessing. 2017. Evoalgos: Modular Evolutionary Algorithms. https://ls11-www.cs.tu-dortmund.de/people/swessing/evoalgos/doc/.
[57] Iryna Yevseyeva, Andreia P. Guerreiro, Michael T. M. Emmerich, and Carlos M. Fonseca. 2014. A Portfolio Optimization Approach to Selection in Multiobjective Evolutionary Algorithms. In *Parallel Problem Solving from Nature – PPSN XIII*. Vol. 8672. Springer International Publishing, Cham, 672–681. https://doi.org/10.1007/978-3-319-10762-2_66
[58] Q. Zhang and H. Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (Dec. 2007), 712–731. https://doi.org/10.1109/TEVC.2007.892759
[59] Y. Zhao and I. Wagner. 2019. On the Strength of Privacy Metrics for Vehicular Communication. *IEEE Transactions on Mobile Computing* 18, 2 (Feb. 2019), 390–403. https://doi.org/10.1109/TMC.2018.2830359
[60] Yuchen Zhao and Isabel Wagner. 2020. Using Metrics Suites to Improve the Measurement of Privacy in Graphs. *IEEE Transactions on Dependable and Secure Computing* (2020). https://doi.org/10.1109/TDSC.2020.2980271
[61] Eckart Zitzler and Simon Künzli. 2004. Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature - PPSN VIII*. Vol. 3242. Springer Berlin Heidelberg, 832–842.
[62] Eckart Zitzler and Lothar Thiele. 1998. Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study. In *Parallel Problem Solving from Nature — PPSN V*. Vol. 1498. Springer Berlin Heidelberg, 292–301. https://doi.org/10.1007/BFb0056872
[63] E. Zitzler and L. Thiele. Nov./1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (Nov./1999), 257–271. https://doi.org/10.1109/4235.797969

A SOLUTION ALGORITHMS

A.1 Single-objective optimization

For optimizing only monotonicity as the most important objective (Equation 2), we use state-of-the-art single-objective EAs and some other metaheuristics: Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [20], Differential Evolution (DE) [47], Particle Swarm Optimization (PSO) [27], and Subplex [42].

A.1.1 Covariance Matrix Adaptation Evolutionary Strategy. Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) is an evolutionary strategy developed for real-valued optimization [21]. At each iteration of the algorithm a vector of individuals is generated according to the multi-variate normal distribution, which follows the maximum entropy principle. Mutation operator uses covariance matrix adaptation (CMA), which learns all pairwise dependencies between variables and enables step size control according to the local curvature and scaling of the objective function. CMA increases the likelihood of previously successful steps speeding up convergence yet preventing premature convergence by controlling the step size.

A.1.2 Differential Evolution. Differential Evolution (DE) [47] uses self-adapting mutation, which changes depending on the objective space surface and adapts to the current population, similarly to CMA-ES [20] and inspired by the simplex search method [40]. The scaled difference between two parents is added to the third one to create an offspring. Mutation happens on selected elements of the parental individual if the mutation probability is below a threshold. The algorithm execution is controlled by a scaling factor, which controls the convergence rate, and a crossover parameter, which directs the rotational invariance search.

A.1.3 Particle Swarm Optimization. Particle Swarm Optimization (PSO) [27] is a metaheuristic that instead of imitating an evolutionary process, simulates the social behaviour of swarms, e.g. birds flocking or fish schooling, where individuals/particles *move* rather than evolve as in EAs. Each

individual of the population is called *particle* and the population is called *swarm*. At each iteration each particle is updated taking into account its parent and velocity, which considers information about the best position that the particle has seen so far (personal best) and the best particle in a certain neighbourhood (global best) formed by the swarm members with which the particle can interact.

A.1.4 Subplex. The final method we use for single objective optimization is *Subplex* [42]. It is the only algorithm in this paper that is neither an evolutionary method nor metaheuristic. Instead, it is based on the Nelder-Mead simplex search method [40]. The Subplex method expands the Nelder-Mead method for problems with more than five dimensions by decomposing the problem into sub-spaces with step size changing depending on the size of the problem. The Subplex method was particularly designed to work with noisy multi-modal many-dimensional functions where the Nelder-Mead method does not perform well, however, it can be used as a general purpose unconstrained optimization method.

A.2 Multi-objective and many-objective optimization

Our optimization problem as formulated in Equation (7) can be considered a many-objective optimization problem because the number of objectives exceeds three. Traditional multi-objective algorithms struggle to solve many-objective problems because of the increased dimension of the search space and the large number of non-dominated solutions obtained. The first difficulty also leads to the diversity evaluation becoming computationally expensive and the recombination operator becoming inefficient. To address the diversity issue many diversity-oriented algorithms have been developed [5].

In this paper, we use multi-objective metaheuristics and a number of state-of-the-art multi-objective evolutionary algorithms (MOEAs) that have been developed specifically to deal with many-objective problems.

A.2.1 NSGA-II and SPEA2. Non-dominated Sorting Genetic Algorithm II (NSGA-II) [11] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [30] belong to the group of elitist Pareto-based state-of-the-art MOEAs. They preserve non-dominated solutions at each iteration either as members of the next generation, like in NSGA-II, or in a separate archive, like in SPEA2.

In NSGA-II, the population is sorted into a set of Pareto fronts based on the non-dominance relation and for the last front(s) the individuals in the most crowded regions are discarded. NSGA-II is often used as a reference benchmark to evaluate newly developed MOEAs. SPEA2 maintains an archive of solutions in addition to the current population. The strength of an individual in the archive is based on the number of members of the population that the individual dominates or equals, while the strength of an individual in the population is the sum of strengths of all individuals in the archive that dominate or equal the individual. The archive size is truncated based on a clustering technique which always preserves outer solutions.

A.2.2 EpsMOEA. EpsMOEA [29] is another elitist algorithm based on ϵ-dominance, i.e. the non-dominance relation is checked with respect to a relaxation interval ϵ within which two solutions are considered to be non-dominated with respect to each other. In EpsMOEA, the objective space is divided into a set of boxes of ϵ size each, and the archive consists of a set of non-dominated boxes with one individual per box instead of a set of non-dominated solutions. An individual is accepted into the archive if it is located in a box non-dominated by any other box, and boxes dominated by a new archive member are removed from the archive.

A.2.3 IBEA. In general, any performance measure (indicator) can be used in an Indicator-Based Evolutionary Algorithm (IBEA) [61] to select the next generations of individuals and in this way to

guide the selection and search process towards a population performing best on the chosen indicator. Most commonly, indicators compliant with Pareto non-dominance and satisfying monotonicity property are used, such as hypervolume and epsilon indicators.

S-metric Selection Evolutionary Multi-objective Algorithm (SMS-EMOA) [15], which at each iteration removes the individual with the worst hypervolume contribution to the overall hypervolume of population, is based on a similar idea to IBEA. More recently, Portfolio Optimization Selection Evolutionary Algorithm (POSEA) was proposed [57], which selects a portfolio of the most diverse individuals of best quality by evaluating Sharpe ratio indicator.

The performance indicator's specific features should be considered when choosing an indicator to guide the search, and in this way they may reflect the decision maker's preferences. In this paper, we use IBEA with the hypervolume indicator, which has been shown to select extreme points and concentrate search around the knee region of the Pareto front [15].

A.2.4 MOEA/D and NSGA-III. Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [58] and Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach (NSGA-III) [10], [24] were developed specifically to handle many objectives. Both MOEA/D and NSGA III use reference points to guide the search towards a limited number of directions in the search space.

MOEA/D decomposes the initial problem into a number of scalar optimization sub-problems and optimizes them simultaneously. Each sub-problem is associated with a weight vector and each individual in the population is associated with a sub-problem randomly. Mating is arranged between neighbouring sub-populations, resulting in an offspring which is associated with one or more sub-problems. Association is based on metrics typical for classic multi-objective optimization with scalarizing functions, such as the Tchebycheff metric.

NSGA-III modifies NSGA-II in its diversity maintenance mechanism. Instead of eliminating individuals in the most crowded part of the last front, NSGA-III uses a set of reference points to cluster points on the last front and then select some points from each cluster. The reference points are either provided by the user or generated in a predefined way, e.g. by placing points evenly on a normalized hyper-plane [10]. Each member of the last front is associated with its closest reference point and the next population is filled from the last front members by adding one random individual from each reference point association at a time.

A.2.5 Multi-Objective Covariance Matrix Adaptation. Multi-Objective Covariance Matrix Adaptation Evolutionary Strategy (MO-CMA-ES) [23] extends the single-objective CMA-ES to the multi-objective case using nondominated sorting either with crowding distance, used in a similar way to NSGA-II [11], or with hypervolume. Covariance matrix self-adaptation evolution strategy (CMSA-ES) [7] is a variation of CMA-ES that replaces cumulative step-size adaptation with self-adaptation.

A.2.6 Multi-Objective Differential Evolution. Generalized Differential Evolution 3 (GDE3) [28] is a multi-objective version of the Differential Evolution algorithm. GDE3 considers both constraint handling and preservation of non-dominating solutions (elitism) in the next iterations of the algorithm. Out of the offspring and parent solutions the selection is made based on the feasibility w.r.t. constraints and non-dominance w.r.t. each other. If both solutions are infeasible, the one which non-dominates the other one in constraint violation space is selected. Out of feasible and infeasible solutions, feasible is selected. In case both solutions are non-dominating w.r.t. each other, they both pass to the next generation, which might lead to increased population size. In the last case, NSGA-II principles of non-dominated sorting and crowding distance checking are applied to reduce the size to the needed one.

A.2.7 Particle Swarm Optimization. Particle swarm optimization (PSO) metaheuristic was also extended to solving multi-objective optimization problems. Two recent versions of PSO for multi-objective optimization are Optimized Multi-Objective Optimization PSO (OMOPSO) and Speed-constrained Multi-objective PSO (SMPSO). OMOPSO [46] uses an external archive to keep the best solutions based on NSGA-II's crowding distance, with the size of the archive truncated by ϵ-dominance. SMPSO [39] extends OMOPSO by using a velocity construction technique to control swarm explosion.

A.3 Performance Measures for Multi-objective Algorithms

Evaluating the performance of MOEAs is complex because of the need to compare solution sets, not just single solutions. The quality of solution sets, or Pareto front approximations, depends on multiple aspects [33], such as how close solutions are to the real Pareto front (*convergence*); how widely the solutions are distributed along the Pareto front approximation (*spread*); how uniformly or evenly solutions are distributed on the Pareto front approximation (*uniformity*); and how many solutions are on the Pareto front approximation (*cardinality*). Some quality indicators integrate several of these aspects into a single value.

Unary quality indicators evaluate the quality of the Pareto front approximation for a single solution set, and *binary* quality indicators compare pairs of solution sets, either two Pareto front approximations or one approximation and the true Pareto front. In this work, we selected one binary and four unary indicators that cover all four quality aspects [33]: C indicator, Hypervolume, Generational Distance, Inverted Generational Distance, and Spacing.

Hypervolume (HV) computes the objective space volume covered by the solutions belonging to the Pareto front approximation set. Hypervolume is probably the most often used unary quality indicator because it covers all four aspects of solution set quality: convergence, uniformity, spread and cardinality, and because of its important *Pareto compliance* property. Pareto compliant indicators preserve the order of sets induced by the non-dominance relation: if one set is "at least as good as" the other set, its hypervolume is bigger.

Generational Distance (GD) [52] is a distance-based indicator for convergence. For each solution, it computes its distance to the Pareto front as the Euclidean distance to the closest solution on the true Pareto front. GD is then given by the quadratic mean of these distances. *Inverted Generational Distance* (IGD) is similar to GD, but measures the distance from the Pareto front to each solution and uses the arithmetic mean in its computation. Lower values are preferable for both GD and IGD.

Spacing (SP) measures the uniformity of the distribution of solutions along the Pareto front approximation as the variation of the distances between solutions [43]. The smaller the value of $SP(A)$ the better uniformity of the set is, that is, the value $SP(A) = 0$ means equal distance between all solutions of the set. Hence, lower values of this indicator are preferred. However, to fully represent diversity of the set other quality indicators need to be used.

The binary *C Indicator* [62] checks the non-dominance relation between all pairs of solutions in two solution sets and calculates the proportion of solutions in the set B that are weakly dominated by at least one solution in the set A. The C indicator is not symmetric, i.e. the value of $C(A, B)$ is different from $C(B, A)$. Its value range is in $[0, 1]$, with $C(A, B) = 1$ indicating that A weakly dominates B, i.e. all solutions in B are weakly dominated by at least one solution in A. Hence, higher values indicate a better solution set. The limitation of the C indicator is that it does not express how much one set is better than the other set.

B RESULTS: PERFORMANCE OF OPTIMIZATION ALGORITHMS

In this section we report on the performance of both single-objective and multi-objective optimization algorithms using a range of performance indicators.

B.1 Single-objective optimization

Even though we have defined our problem as a 5-objective optimization problem – with the goal to find well-rounded metrics suites that perform well on all five objectives – we have also argued that the monotonicity objective is the most important of the five and should take precedence over the other objectives. Therefore, we first use single-objective optimization to estimate what the best monotonicity values are for each of the three datasets. To do this, we applied four single-objective optimization algorithms – CMA-ES, DE, PSO, and Subplex – to maximize the monotonicity objective M_1 (Equation 2) without considering the other objectives. We can then use these monotonicity values as a baseline to evaluate the solutions found by multi-objective optimization algorithms.

Table 4 shows the best monotonicity values achieved by each of the algorithms for each of the datasets. We represent monotonicity as the portion of adversary strength levels that were ranked correctly by WPM, resulting in a range of [0, 1]. The table shows that for the genomics and vanet datasets, most of the algorithms are able to find perfect metrics suites that rank all adversary strength levels correctly. For the graph dataset, the best monotonicity values are just above 90%, with particle swarm optimization finding the best metrics suite with 90.5% monotonicity.

Table 4 also shows how many metrics the shortest metrics suite consisted of that achieved the best monotonicity. We can see that the metrics suites are undesirably large, with more than ten metrics in most metrics suites. While these metrics suites may have excellent monotonicity properties, they are likely to be impractical for most applications, due to the overheads of having to implement, analyze and interpret the large number of metrics. This shows that multi-objective optimization is indeed needed for the problem of finding metrics suites.

Table 4. Results for single-objective optimization algorithms

Dataset	Algorithm	Best monotonicity	Smallest number of metrics
Genomics	CMA-ES	1.0	7
Genomics	DE	1.0	7
Genomics	PSO	1.0	9
Genomics	SBPLX	1.0	6
Graph	CMA-ES	0.901	9
Graph	DE	0.903	13
Graph	PSO	0.905	14
Graph	SBPLX	0.901	19
Vanet	CMA-ES	1.0	11
Vanet	DE	1.0	11
Vanet	PSO	1.0	10
Vanet	SBPLX	0.992	14

B.2 MO optimization: number of function evaluations

Before we evaluate the performance of the multi-objective optimization algorithms, we need to verify that the number of 50,400 function evaluations we selected for the optimization process is sufficient for the algorithms to generate good solutions. To do so, we performed initial runs of the optimization with between 75 and 240 generations. Figure 7 shows the hypervolume indicator for each dataset after each generation. We can see that most algorithms converged after 50–100 generations, with the exception of CMA-ES-deap, which converged only after about 350 generations for the graph and vanet datasets (not shown). This means that our choice of 50,400 function evaluations is sufficiently large to allow most optimization algorithms to converge.

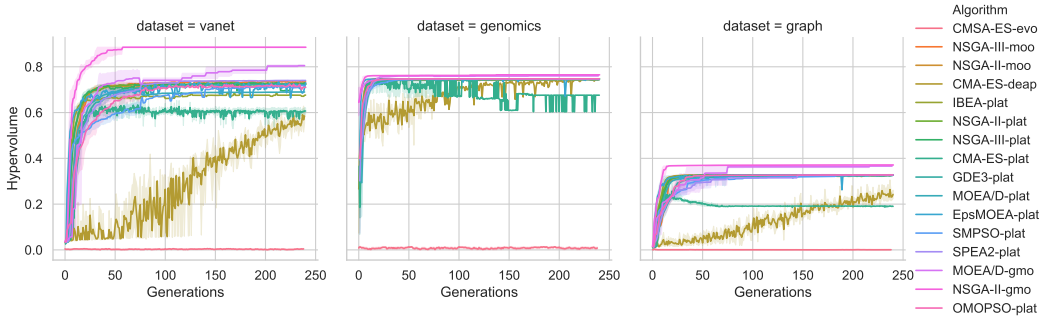


Fig. 7. Convergence of the hypervolume indicator during the optimization process.

B.3 MO optimization: performance

We now evaluate the performance of multi-objective algorithms according to four established performance indicators: hypervolume, generational distance, inverted generational distance, and spacing as described in Section A.3. Our aim is to determine which algorithms are particularly suitable for our problem, and which algorithms are not, by excluding the algorithms that perform worst on each performance indicator.

The plots in this section are color-coded to indicate semantic similarities between algorithms: we chose different shades of red for MOEA/D implementations, shades of blue for NSGA-II and SPEA2, greens for NSGA-III, purples for covariance matrix based algorithms, pinks for particle swarm algorithms, and greys for algorithms that are not similar to any of the others (EpsMOEA, GDE3, IBEA).

B.3.1 Hypervolume (HV). Hypervolume is a general-purpose indicator that evaluates the convergence, spread, and cardinality of solution sets. As Figure 8 shows, most algorithms perform well on this indicator. However, there are some exceptions. The worst algorithm is CMSA-ES, which has a hypervolume of 0 in almost all cases. This is because CMSA-ES generates only a single unique solution and thus has a low cardinality. The two implementations from the pymoo package, NSGA-II and NSGA-III, as well as IBEA, also perform noticeably worse than the other algorithms. Both implementations of CMA-ES, although below the best in the graph and vanet datasets, perform very well in the genomics dataset.

B.3.2 Generational Distance (GD). The generational distance is a specialized indicator that shows how well solution sets converge to the Pareto front. As Figure 9 shows, most algorithms that performed worst on hypervolume are also among the worst performers on generational distance. In addition, the particle swarm algorithms, SMPSO in particular, are worse than the others. OMOPSO and EpsMOEA perform badly for the genomics and graph datasets, but very well for the vanet dataset. IBEA, on the other hand, performs quite well on all datasets.

B.3.3 Inverted generational distance (IGD). Inverted generational distance indicates both the spread and the convergence properties of solution sets. Compared to the results for hypervolume in Figure 8, IGD in Figure 10 shows bigger differences in performance between the 16 algorithms. The reason for this is that HV includes the cardinality of solution sets, which in our case pushes most algorithms towards high HV values. Because IGD does not consider cardinality, the spread and convergence properties are more pronounced. This allows us to identify additional algorithms that clearly perform worse than others. The MOEA/D implementation from the pagmo2 package, CMA-ES

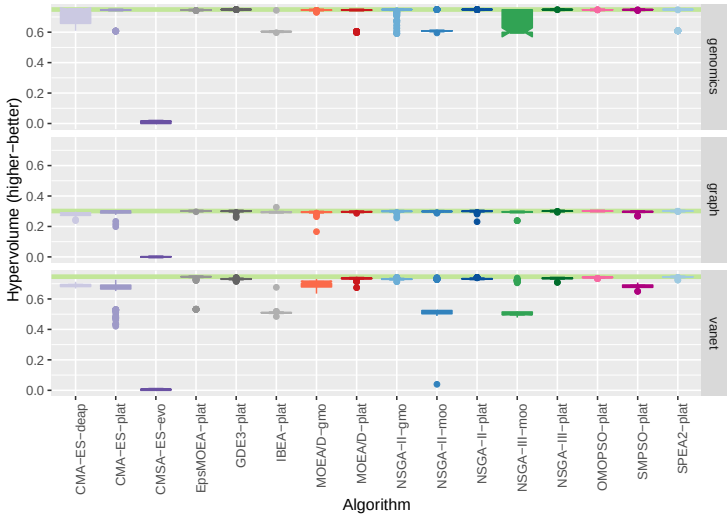


Fig. 8. Hypervolume for all algorithms and datasets. The light-green horizontal line indicates the best median hypervolume achieved by any of the algorithms. Higher values indicate better performance.

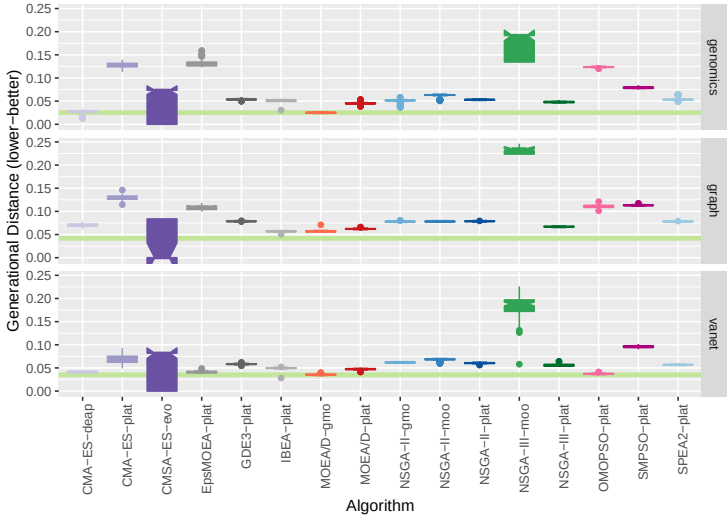


Fig. 9. Generational Distance for all algorithms and datasets. Lower values indicate better performance.

(deap package), EpsMOEA, IBEA, and the MOEA/D implementation from Platypus all perform worse in at least two of three datasets.

The NSGA-II implementations from the pagmo2 and Platypus packages belong to the best performers on all indicators so far. This is a surprising result because NSGA-II is typically said to be unsuitable for problems with more than three objectives.

B.3.4 Spacing (SP). Spacing indicates how uniformly solutions are distributed in the solution space. Importantly, it neither takes into account the size of the solution space, nor how far solutions

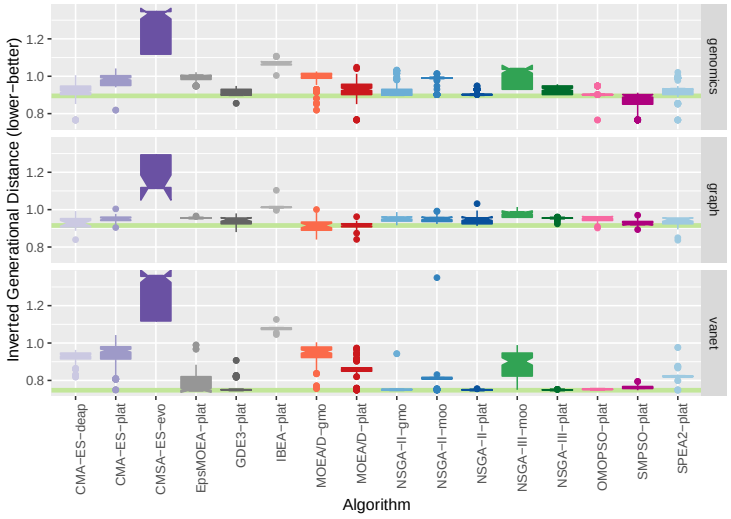


Fig. 10. Inverted Generational Distance. Lower values indicate better performance.

are from the Pareto front. This means that algorithms whose solutions are spread uniformly on a very small portion of the solution space would perform very well, but may not be the best choice of algorithm because they do not cover the entire Pareto front. As such, we treat spacing as a supplementary indicator.

Most algorithms in our experiment achieve roughly comparable spacings. The only exception is CMSA-ES which clearly achieves the best spacing (see Figure 11). However, further analysis shows that this is because CMSA-ES produces only one unique non-dominated solution. The range of the solution space is thus limited to one solution, and this solution is perfectly spaced. This illustrates that spacing alone cannot adequately describe the performance of optimization algorithms.

B.4 Differences between MO implementations

Our evaluation includes algorithms that are semantically similar, i.e. that are based on similar ideas, as well as different implementations of the same algorithms. We now compare these groups of algorithms and implementations to analyze further which algorithms are most suitable for our problem.

We use the C indicator, i.e. the portion of solutions generated by one algorithm that are dominated by at least one solution generated by the other algorithm, for pairwise comparison of algorithms. In contrast to the indicators above, set coverage thus allows to compare implementations directly with each other.

B.4.1 NSGA-II and SPEA2. Conceptually, NSGA-II and SPEA2 are very similar. We compare three implementations of NSGA-II and one implementation of SPEA2. The hypervolume indicator (see Figure 8) showed that the NSGA-II implementation from the pymoo package performed clearly worse than the other implementations. These other implementations performed at a comparable level, at least in their medians across all replications.

Figure 12 shows the pairwise C indicator, aggregated to compare the four implementations with each other. The scores indicate how frequently an algorithm generates solutions that are dominated by solutions from the other algorithms. Lower scores thus indicate better algorithms. Figure 12

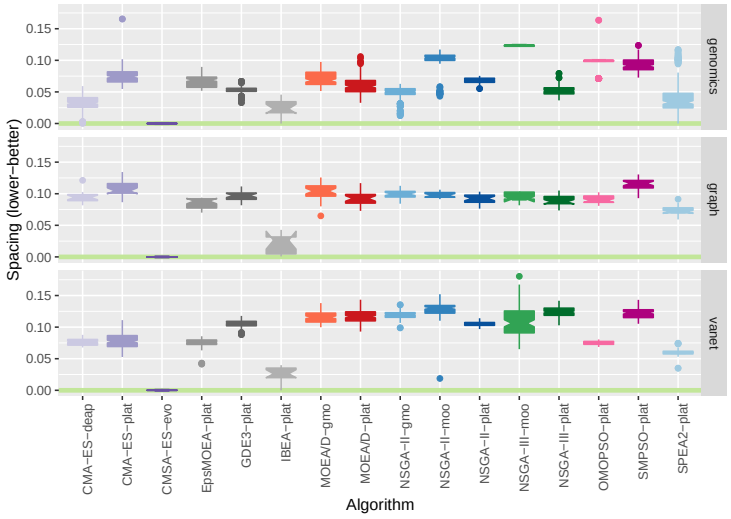


Fig. 11. Spacing. Lower values indicate better performance.

confirms that the pymoo implementation performs worst across all datasets and shows that the performance of the other three implementations depends on the dataset. Overall, the NSGA-II and SPEA2 implementations from Platypus have a slight advantage over the others.

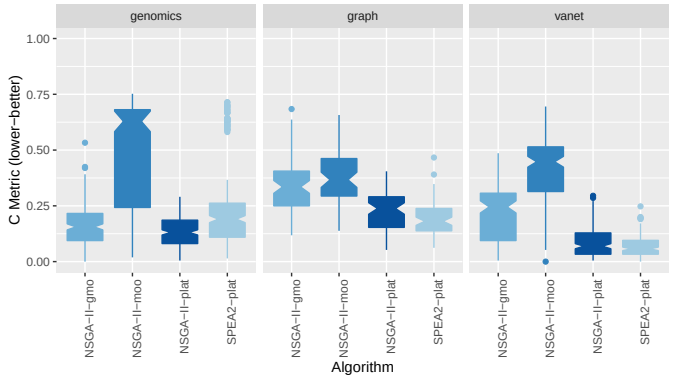


Fig. 12. C indicator for NSGA-II and SPEA2. The y axis indicates the portion of solutions that are dominated by at least one solution from another algorithm in the comparison. Lower values thus indicate more non-dominated solutions, i.e. a preferable algorithm.

B.4.2 NSGA-III and MOEA/D. NSGA-III and MOEA/D are algorithms that have been reported to perform best for many-objective problems. We compare two implementations for each algorithm. The hypervolume indicator (see Figure 8) showed that the NSGA-III implementation from the pymoo package performed worst, but did not show a large difference when compared to the other three implementations. The C indicator in Figure 13 shows that in fact, both MOEA/D implementations generate more dominated solutions, and thus perform worse, than the NSGA-III implementations. The NSGA-III implementation from Platypus performs clearly best in all three datasets.

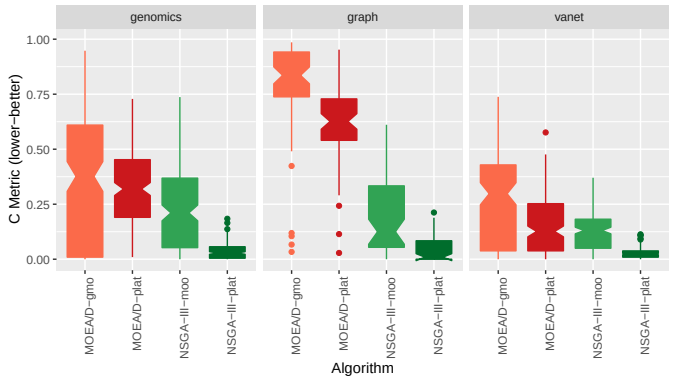
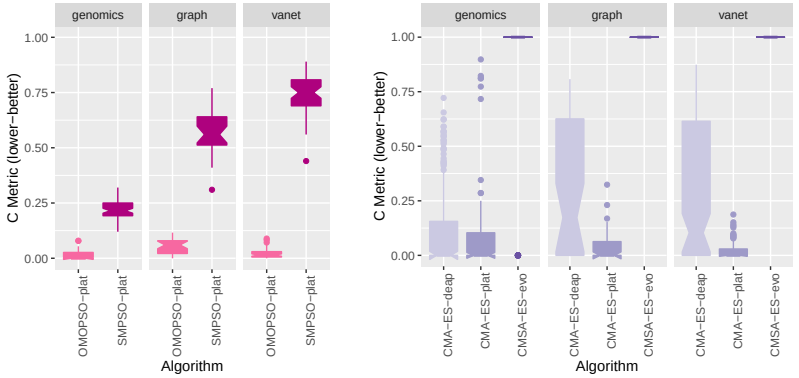


Fig. 13. C indicator for NSGA-III and MOEA/D.



(a) C indicator for OMOPSO and SMPSO.

(b) C indicator for multiobjective CMA-ES and CMSA-ES.

Fig. 14. C indicator for (a) particle swarm and (b) covariance matrix based algorithms

B.4.3 Particle Swarm Algorithms. The hypervolume indicator in Figure 8 did not show a large difference between the two particle swarm algorithms OMOPSO and SMPSO. However, the pairwise C indicator (see Figure 14a) clearly shows that solutions generated by SMPSO are often dominated by OMOPSO solutions, across all three datasets. This is a surprising finding, because SMPSO is the newer algorithm that was developed as an improvement of OMOPSO.

B.4.4 Covariance Matrix Algorithms. We compared three implementations of covariance matrix based algorithms, two of them implementing the CMA-ES algorithm (from Platypus and DEAP), and one implementing CMSA-ES. The unary indicators showed that CMSA-ES performs worst on our problem, and that the DEAP implementation seems slightly better than the Platypus implementation of CMA-ES.

The C indicator in Figure 14b confirms that solutions generated by CMSA-ES are always dominated by solutions from the other implementations. However, the figure also indicates that solutions from the DEAP implementation are dominated more often than those generated by the Platypus implementation, contradicting the findings from the unary indicators.