# Fast Pareto Optimization Using Sliding Window Selection for Problems with Deterministic and Stochastic Constraints

A Preprint

Frank Neumann
Optimisation and Logistics
School of Computer Science and Information Technology
Adelaide University
Adelaide, Australia

Carsten Witt
Algorithms, Logic and Graphs
DTU Compute
Technical University of Denmark
2800 Kgs. Lyngby Denmark

January 7, 2026

## Abstract

Submodular optimization problems play a key role in artificial intelligence as they allow to capture many important problems in machine learning, data science, and social networks. Pareto optimization using evolutionary multi-objective algorithms such as GSEMO (also called POMC) has been widely applied to solve constrained submodular optimization problems. A crucial factor determining the runtime of the used evolutionary algorithms to obtain good approximations is the population size of the algorithms which usually grows with the number of trade-offs that the algorithms encounter. In this paper, we introduce a sliding window speed up technique for recently introduced algorithms. We first examine the setting of deterministic constraints for which bi-objective formulations have been proposed in the literature. We prove that our technique eliminates the population size as a crucial factor negatively impacting the runtime bounds of the classical GSEMO algorithm and achieves the same theoretical performance guarantees as previous approaches within less computation time. Our experimental investigations for the classical maximum coverage problem confirm that our sliding window technique clearly leads to better results for a wide range of instances and constraint settings. After we have shown that the sliding approach leads to significant improvements for bi-objective formulations, we examine how to speed up a recently introduced 3-objective formulation for stochastic constraints. We show through theoretical and experimental investigations that the sliding window approach also leads to significant improvements for such 3-objective formulations as it allows for a more tailored parent selection that matches the optimization progress of the algorithm.

## 1 Introduction

General heuristic search methods (Zhang et al., 2001; Korf, 2010), evolutionary algorithms (Eiben and Smith, 2015), and constrained programming (Rossi et al., 2006) provide methods for solving a wide range of complex optimization and search problems in areas such as planning, scheduling, robotics and games. An important area where greedy heuristic search methods and evolutionary algorithms have been applied is the area of submodular optimization (Krause and Golovin, 2014; Qian et al., 2019; Gu et al., 2023). The research in this domain captures a wide range of important problems arising in artificial intelligence such as influence maximization in social networks (Kempe et al., 2015) or regression problems in machine learning (Das and Kempe, 2011). These and many other real-world optimization problems face diminishing returns when adding additional elements to a solution and can be formulated in terms of a submodular function (Nemhauser and Wolsey, 1978; Krause and Golovin, 2014). Other important problems that can be stated in terms of a submodular function include classical NP-hard combinatorial optimization problems such as the computation of a maximum coverage (Khuller et al., 1999) or maximum cut (Goemans and Williamson, 1995) in graphs which are often underlying problems to be solved as part of complex applications.

Evolutionary algorithms have been widely applied to tackle complex optimization problems from the areas of artificial intelligence and operations research. One of their major success stories is in the area of multi-objective optimization. As evolutionary algorithms use a population to solve a given problem, such a population can be used to evolve a set of solutions that represents trade-offs according to a given set of (conflicting) objective functions. Evolutionary multi-objective algorithms have also been widely applied to constrained single-objective optimization problems. Here, the constraint is treated as an additional objective. Formulating a constrained single-objective optimization problem as a multi-objective problem allows evolutionary algorithms a different way of tackling the given problem Friedrich and Neumann (2015); Knowles et al. (2001). In particular, this enables evolutionary multi-objective algorithms to mimic a greedy behavior which is highly beneficial for tackling constrained submodular optimization problems.
Classical approximation algorithms for monotone submodular optimization problems under different types of constraints rely on greedy approaches which select elements with the largest benefit/cost gain according to the gain with respect to the given submodular function and the additional cost with respect to the given constraint (Zhang and Vorobeychik, 2016; Krause and Golovin, 2014). During the last years, evolutionary multi-objective algorithms such as GSEMO (also called POMC in the artificial intelligence literature) have been shown to provide the same theoretical worst case performance guarantees on solution quality as the greedy approaches while clearly outperforming classical greedy approximation algorithms in practice (Friedrich and Neumann, 2015; Qian et al., 2015, 2017; Neumann and Neumann, 2025). Results are obtained by means of rigorous runtime analysis (see (Neumann and Witt, 2010; Doerr and Neumann, 2020) for comprehensive overviews) which is a major tool for the analysis of evolutionary algorithms. In particular, this tool gives rigorously proven performance guarantees for the algorithms, including guidelines for high-performing parameter choices.
Solving constrained single-objective problems through multiobjectivization (Knowles et al., 2001) has been shown to be highly successful for a wide range of problems (Ma et al., 2023). In the context of submodular optimization, the analyzed approaches use a 2-objective formulation of the problem where the first objective is to maximize the given submodular function and the second objective is to minimize the cost of the given constraint. The two objectives are optimized simultaneously using variants of the GSEMO algorithm (Laumanns et al., 2004; Giel, 2003) and the algorithm outputs the feasible solution with the highest function value when terminating. Before GSEMO has been analyzed for submodular problems, similar multi-objective setups have already been shown to be provably successful for other constrained single-objective optimization problems such as minimum spanning trees (Neumann and Wegener, 2006) and vertex and set covering problems (Friedrich et al., 2010; Kratsch and Neumann, 2013) through different types of runtime analyses.
A crucial factor which significantly influences the runtime of these Pareto optimization approaches using GSEMO is the size of the population that the algorithm encounters. This is in particular the case for problems where both objectives can potentially attain exponentially many different function values. In the context of submodular optimization this is the case iff the function $f$ to be optimized as well as the constraint allows exponentially many trade-offs. In order to produce a new solution in the algorithm, a solution chosen uniformly at random from the current population is mutated to produce an offspring. A large population size implies that often time is wasted by not selecting the "right" individual for mutation. This effect becomes even more predominant when problems with more than 2 objectives are considered. In the context of problems with stochastic constraints, it has recently been shown that 3-objective formulations where the given constraint is relaxed into a third objective lead to better performance than 2-objective formulations that optimize the expected value and variance of the given stochastic components under the given constraint (Neumann and Witt, 2023a, 2025). The experimental investigations for the chance constrained dominating set problem carried out in Neumann and Witt (2023a) show that the 3-objective approach is beneficial and outperforms the bi-objective one introduced in Neumann and Witt (2025) for medium size instances of the problem. However, it has difficulties in computing even a feasible solution for larger graphs. In order to deal with large problem instances, we design a new 3-objective Pareto optimization approach based on the sliding window technique.

### 1.1 Our contribution

In this article, we present a sliding window approach for fast Pareto optimization called SW-GSEMO that tries to select the "right" parent for mutation at any given point in time. We should note that our sliding window approaches differ significantly from other parent selection mechanisms explored in the literature (Laumanns et al., 2004; Osuna et al., 2020; Bian and Qian, 2022; Antipov et al., 2024). These mechanism have been designed and explored in the context of standard multi-objective formulations where the goal is to cover the whole Pareto front.
Our approach is based on the fact that several previous runtime analyses of GSEMO for Pareto optimization (e. g., Friedrich and Neumann (2015); Qian et al. (2017)) consider improving steps of a certain kind only.

More precisely, they follow an inductive approach, e. g., where high-quality solutions having a cost value of at most $i$ in the constraint are mutated to high-quality solutions of constraint cost value at most $i+1$, which we call a success. (More general sequences of larger increase per success are possible and will be considered in this paper.) Steps choosing individuals of other constraint cost values do not have any benefit for the analysis. Assuming that the successes increasing the quality and cost constraint value have the same success probability (or at least the same lower bound on it), the analysis essentially considers phases of uniform length where each phase must be concluded by a success for the next phase to start. More precisely, the runtime analysis typically allocates a window of $t^*$ steps, for a certain number $t^*$ depending on the success probability, and proves that a success is for within an expected number of $t^*$ steps or is even highly likely in such a phase. Nevertheless, although only steps choosing the best individual of constraint value at most $i$ are relevant for the phase, the classical GSEMO selects the individual to be mutated uniformly a random from the population, leading to the "wasted" steps as mentioned above.

Our sliding window approach replaces the uniform selection with a time-dependent window for selecting individuals based on their constraint cost value. This time-dependent window is of uniform length $T$, which is determined by the parameters of the new algorithm, and chooses high quality individuals with constraint cost value at most $i$ for $T$ steps each. In our analysis, $T$ will be at least $t^*$, i. e., a proven length of the phase allowing a success towards a high-quality solution of the following constraint cost value with high probability. This allows the algorithm to make time-dependent progress and to focus the search on solutions of a "beneficial" cost constraint value, which in the end results with high probability in the same approximation guarantee as the standard Pareto optimization approaches using classical GSEMO. Our analysis points out that our proven runtime bound is independent of the maximum population size that the algorithm attains during its run. This provides significantly better upper bounds for our fast Pareto optimization approach compared to previous Pareto optimization approaches.

Sliding window selection works well with two objectives, one of which represents a constraint value, since usually the population will only contain one individual for each constraint value. Making the sliding window technique work for problems with 3 (or more) objectives requires one to deal with a potentially large number of trade-off solutions even for a small number of constraint values. We design highly effective 3-objective Pareto optimization approaches based on the sliding window technique. Our theoretical investigations using runtime analysis for the chance constrained problem under a uniform constraint show that our approach may lead to a significant speed-up in obtaining all required Pareto optimal solutions. In order to make the approach even more suitable in practice, we introduce additional techniques that do not hinder the theoretical performance guarantees, but provide additional benefits in applications. One important technique is to control the sliding window through an additional parameter $a$. Choosing $a \in ]0,1[$ allows the algorithm to move the sliding window faster at the beginning of the optimization process and slow down when approaching the constraint bound. This allows us to maintain the benefit of the Pareto optimization approach including its theoretical performance guarantees while focusing more on the improvement of already high quality solutions at the end of the optimization run. The second technique that we incorporate is especially important for problems like the dominating set problem where a constraint that is not fulfilled for most of the optimization process needs to be fulfilled at the end. In order to deal with this, we introduce a parameter $t_{frac} \in [0,1]$ which determines the fraction of time our sliding window technique is used. If after $t_{frac} \cdot t_{max}$ steps a feasible solution has not been found yet, then in each step a solution from the population that is closest to feasible is selected in the parent selection step to achieve feasibility within the last $(1-t_{frac}) \cdot t_{max}$ steps.

This article is based on two preliminary conference versions. The bi-objective sliding window approach introduced in Neumann and Witt (2023b) and the 3-objective sliding window approach introduced in Neumann and Witt (2024). Furthermore, the investigations in Neumann and Witt (2023b) have been extended and an additional analysis for the expected approximation ratio based on a given time budget is provided in Section 3.2.

The outline of the article is as follows. In Section 2, we introduce the class of problems we consider in this work. Section 3 presents our new SW-GSEMO algorithm for bi-objective formulations, including a theoretical analysis and an experimental evaluation. Our sliding window approach for 3-objective problems together with its theoretical analysis and experimental investigations is presented in Section 4. Finally, we finish with some concluding remarks.

## 2 Preliminaries

In this section, we describe the problem classes and algorithms relevant to our study. Overall, the aim is to maximize pseudo-boolean fitness/objective functions $f\colon \{0,1\}^n \to \mathbb{R}$ under constraints on the allowed search points. An important class of such functions is given by the so-called submodular functions (Nemhauser and Wolsey, 1978). We formulate it here on bit strings on length $n$; in the literature, also an equivalent

formulation using subsets of a ground set $V = \{v_1, \ldots, v_n\}$ can be found. The notation $x \leq y$ for two bit strings $x, y \in \{0,1\}^n$ means that $x$ is component-wise no larger than $y$, i. e., $x_i \leq y_i$ for all $i \in \{1, \ldots, n\}$.

Definition 1. A function $f\colon \{0,1\}^n \to \mathbb{R}^+$ is called submodular if for all $x, y \in \{0,1\}^n$ where $x \leq y$ and all $i$ where $x_i = y_i = 0$ it holds that

$$f(x \oplus e_i) - f(x) \geq f(y \oplus e_i) - f(y)$$

where $a \oplus e_i$ is the bit string obtained by setting bit $i$ of $a$ to 1 (and not changing the other bits).

We will also consider functions that are monotone (either being submodular at the same time or without being submodular). A function $f\colon \{0,1\}^n \to \mathbb{R}$ is called monotone if for all $x, y \in \{0,1\}^n$ where $x \leq y$ it holds that $f(x) \leq f(y)$, i. e., setting bits to 1 without setting existing 1-bits to 0 will not decrease fitness.

### 2.1 Deterministic cost constraint

Optimizing motonone functions becomes difficult if different constraints are introduced. Formally, there is a cost function $c\colon \{0,1\}^n \to \mathbb{R}$ and a budget $B \in \mathbb{R}$ that the cost has to respect. Then the general optimization problem is defined as follows.

Definition 2 (General Optimization Problem). Given a monotone objective function $f\colon \{0,1\}^n \to \mathbb{R}$, a monotone cost function $c\colon \{0,1\}^n \to \mathbb{R}$ and a budget $B \in \mathbb{R}$, find

$$\arg \max_{x \in \{0,1\}^n} f(x) \text{ such that } c(x) \leq B.$$

Without loss of generality, we assume that the monotone functions $f$ and $c$ are normalized, i.e. $f(0^n) = c(0^n) = 0$ holds. A constraint function $c$ is called uniform if it just counts the number of one-bits in $x$, i. e., $c(x) = \sum_{i=1}^{n} x_i$. It should be noted that the most simple setting of optimizing a monotone submodular function under a uniform constraint is already NP-hard but allows finding a $(1 - 1/e)$-approximation using greedy algorithms (Nemhauser and Wolsey, 1978).

In recent years, several variations of the problem in Definition 2 have been solved using multi-objective evolutionary algorithms like the classical GSEMO algorithm (Friedrich and Neumann, 2015; Qian et al., 2015, 2017; Roostapour et al., 2022). In the following, we describe the basic concepts of multi-objective optimization relevant for our approach.

For the deterministic setting, we consider a multi-objective objective function given as $f_D(x) = (f_1(x), f_2(x))$, where $f_2(x) = c(x)$ and

$$f_1(x) = \begin{cases} -\infty & \text{if } c(x) > B \\ f(x) & \text{otherwise} \end{cases},$$

Let $x, y \in \{0,1\}^n$ be two search points. We say that $x$ (weakly) dominates y ($x \succeq y$) iff $f_1(x) \geq f_1(y)$ and $f_2(x) \leq f_2(y)$ holds. We say $x$ strictly dominates $y$ ($x \succ y$) iff $x \succeq y$ and $(f_1(x) \neq f_1(y) \vee f_2(x) \neq f_2(y))$ holds. The dominance relationship applies in the same way to the objective vectors $(f(x), c(x))$ of the solutions. The set of non-dominated solutions is called the Pareto set and the set of non-dominated objectice vectors is called the Pareto front.

The classical goal in multi-objective optimization is to compute for each Pareto optimal objective vector a corresponding solution. The approaches using Pareto optimization for constrained submodular problems as investigated in, e. g., Friedrich and Neumann (2015); Qian et al. (2017), differ from this goal. Here the multi-objective approach is used to obtain a feasible solution, i. e., a solution for which $c(x) \leq B$ holds, that has the largest possible function value $f(x)$.

### 2.2 Chance Constrained Problems

Chance constrained problems involve stochastic constraints that are impacted by the expected (cost) value as well as its variance. In Neumann and Witt (2025), a chance constrained problem has been considered which involves such stochastic components and has an additional deterministic constraint. Pareto optimization approaches are usually used to tackle constrained single-objective optimization problems by taking the constraint as an additional objective.

We consider the chance constrained problem investigated in Neumann and Witt (2025) and motivate our multi-objective settings by these recent investigations. Given a set of $n$ items $V = \{v_1, \ldots, v_n\}$ with stochastic weights $w_i$, $1 \leq i \leq n$, we want to solve

$$\min W \quad \text{subject to} \quad (Pr(w(x) \leq W) \geq \alpha) \wedge (|x|_1 \geq k) \tag{1}$$

where $w(x) = \sum_{i=1}^{n} w_i x_i$, $x \in \{0,1\}^n$, and $\alpha \in [1/2, 1[$. We assume that the weights of the items are independent of each other and that for each item $v_i$ the weight $w_i$ is distributed according to a normal

distribution $N(\mu_i, \sigma_i^2)$, where $\mu_i \geq 1$ and $\sigma_i \geq 1$, $1 \leq i \leq n$. We denote by $\mu(x) = \sum_{i=1}^{n} \mu_i x_i$ the expected weight and by $v(x) = \sum_{i-1}^{n} \sigma_i^2 x_i$ the variance of the weight of solution $x$.
Based on the properties of the normal distribution explored in Ishii et al. (1981), the problem given in Equation (1) is equivalent to minimizing

$$\hat{w}(x) = \mu(x) + K_\alpha \sqrt{v(x)}, \tag{2}$$

under the condition that $|x|_1 \geq k$ holds (Neumann and Witt, 2025). Here, $K_\alpha$ denotes the $\alpha$-fractile point of the standard normal distribution.
The uniform constraint $|x|_1 \geq k$ requires that each feasible solution has to contain at least $k$ elements. As expected weights and variances are strictly positive, an optimal solution has exactly $k$ elements. Depending on the choice of $\alpha$, the difficulty lies in finding the right trade-off between the expected weight and variance among all solutions with exactly $k$ elements.
It has been shown that the problem given in Equation 1 can be solved by the following bi-objective formulation for any $\alpha \in [1/2, 1[$ (Theorem 3 in Neumann and Witt (2025)). The objective function is given as $f_{2D}(x) = (\hat{\mu}(x), \hat{v}(x))$, where

$$\hat{\mu}(x) = \begin{cases} \sum_{i=1}^{n} \mu_i x_i & |x|_1 \geq k \\ (k - |x|_1) \cdot (1 + \sum_{i=1}^{n} \mu_i) & |x|_1 < k \end{cases}$$

$$\hat{v}(x) = \begin{cases} \sum_{i=1}^{n} \sigma_i^2 x_i & |x|_1 \geq k \\ (k - |x|_1) \cdot (1 + \sum_{i=1}^{n} \sigma_i^2) & |x|_1 < k, \end{cases}$$

both of which are to be minimized. Here, the model allows to obtain trade-offs with respect to $\hat{\mu}$ and $\hat{v}$ resulting in an optimal solution for any $\alpha \in [1/2, 1[$.
Similarly as above, we say that a solution $x$ dominates a solution $y$ ($x \succeq y$) iff $\hat{\mu}(x) \leq \hat{\mu}(y) \wedge \hat{v}(x) \leq \hat{v}(y)$. Furthermore, a solution $x$ strongly dominates a solution $y$ ($x \succ y$) iff $x \succeq y$ and $f_{2D}(x) \neq f_{2D}(y)$. The setup can be generalized by using $c(x) \geq k$ for a constraint function $c(x)$ instead of $|x|_1 \geq k$. In the experimental investigations carried out in Neumann and Witt (2025), $c(x)$ is counting the number of dominated nodes in the dominating set problem in graphs with $n$ nodes. Here, the set of dominated nodes refers to the set of nodes selected by $x$ and their neighbors in the given graph. The constraint $c(x) = n$ is used and enforces that each node in the graph is either selected by $x$ or a neighbor of a selected node which means that the set of nodes selected by $x$ forms a dominating set.
The key idea of the result given in Neumann and Witt (2025) is to show that the algorithm computes the extremal points of the Pareto front of the given problem. As the expected costs and variances are strictly positive, each Pareto optimal solution contains exactly $k$ elements when considering this bi-objective formulation.
We also consider the problem of maximizing a given deterministic objective $c(x)$ under a given chance constraint, i.e

$$\max c(x) \quad \text{subject to} \quad Pr(w(x) \leq B) \geq \alpha. \tag{3}$$

with $w(x) = \sum_{i=1}^{n} w_i x_i$ where each $w_i$ is chosen independently of the other according to a normal distribution $N(\mu_i, \sigma_i^2)$, and $B$ and $\alpha \in [1/2, 1[$ are a given weight bound and reliability probability.
Such a problem formulation includes for example the maximum coverage problem in graphs with so-called chance constraints (Doerr et al., 2020; Neumann and Neumann, 2025), where $c(x)$ denotes the nodes covered by a given solution $x$ and the costs are stochastic. Furthermore, the chance constrained knapsack problem as investigated in Xie et al. (2020, 2019) fits into this problem formulation.

#### 2.2.1 3-Objective Formulation

We investigate the 3-objective formulation given as

$$f_{3D}(x) = (\mu(x), v(x), c(x))$$

where $c(x)$ is the constraint value of a given solution that should be maximized. In our theoretical study, we focus on the case $c(x) = |x|_1$, which turns the constraint $|x|_1 \geq k$ into the additional objective of maximizing the number of bits in the given bitstring.
Similar to the bi-objective model we minimize the expected weight $\mu(x) = \sum_{i=1}^{n} \mu_i x_i$ and the variance $v(x) = \sum_{i=1}^{n} \sigma_i^2 x_i$ of the weight of solution $x$. Note that here we do not consider penalty terms for violating the constraint $|x|_1 \geq k$ as done in the bi-objective formulation. We say that a solution $x$ dominates a solution $y$ ($x \succeq y$) iff $c(x) \geq c(y) \wedge \mu(x) \leq \mu(y) \wedge v(x) \leq v(y)$. Furthermore, a solution $x$ strongly dominates $y$ ($x \succ y$) iff $x \succeq y$ and $f_{3D}(x) \neq f_{3D}(y)$.

Algorithm 1: GSEMO

Choose an initial solution $x \in \{0,1\}^n$;
$P \leftarrow \{x\}$;
repeat
  Choose $x \in P$ uniformly at random;
  Create $y$ by flipping each bit $x_i$ of $x$ with probability $\frac{1}{n}$;
  if $\nexists w \in P : w \prec y$ then
    $P \leftarrow (P \setminus \{z \in P \mid y \preceq z\}) \cup \{y\}$;
until *stop*;

Using the expected cost and variance as objectives for the problem given in Equation 3, allows here to explore the trade-offs with respect to the expected cost and variance for the different values of $B$ and $\alpha$ that lead to a maximum possible value of $c(x)$. It has been shown in Neumann and Witt (2023a) that the 3-objective formulation obtains for any possible $B$ and $\alpha \in [1/2, 1[$ a feasible solution with the maximal value for $c(x) = |x|_1$ in expected pseudo-polynomial time. We will investigate possible speed ups for this 3-objective formulation using our newly developed sliding window technique.

### 2.3 Notation

In this paper, we use $\ln(x)$ and $\log(x)$ to denote the natural and binary logarithm of $x$, respectively.

## 3 2-Objective Sliding Window GSEMO

The classical global simple evolutionary multi-objective optimizer (GSEMO) algorithm (Laumanns et al., 2004; Giel, 2003) (see Algorithm 1) has been widely used in the context of Pareto optimization. As done in Qian et al. (2017), we consider the variant starting with the search point $0^n$ in our investigations for monotone submodular problems with deterministic constraints. The search point $0^n$ is crucial for the success of Pareto optimization algorithms and the basis for obtaining theoretical performance guarantees. More precisely, the theoretical studies will first bound the time to reach $0^n$ and from there the time to find certain points of a good approximation guarantee. GSEMO keeps for each non-dominated objective vector obtained during the optimization run exactly one solution in its current population $P$. In each iteration one solution $x \in P$ is chosen for mutation to produce an offspring $y$. The solution $y$ is accepted and included in the population if there is no solution $z$ in the current population that strictly dominates $y$. If $y$ is accepted, then all solutions that are (weakly) dominated by $y$ are removed from $P$.

We introduce the Sliding Window GSEMO (SW-GSEMO) algorithm given in Algorithm 2. We initialize this algorithm with the search point $0^n$ which is Pareto optimal and therefore never removed from the population, and discuss more general initialization for fast Pareto optimization in Section 4 when considering problems with stochastic constraints. SW-GSEMO differs from the classical GSEMO algorithm by selecting the parent $x$ that is used for mutation in a time dependent way with respect to its constraint value $c(x)$ (see the sliding-selection procedure given in Algorithm 3). Let $t_{max}$ be the total time that we allocate to the sliding window approach. Let $t$ the the current iteration number. If $t \leq t_{max}$, then we select an individual of constraint value which matches the linear time progress from 0 to the constraint bound $B$, i.e. an individual with constraint value $\tilde{c} = B/t_{max}$. As this value might not be integral, we use the interval $[\lfloor \tilde{c} \rfloor, \lceil \tilde{c} \rceil]$. In the case that there is no such individual in the population, the individual $x \in P$ with the largest function value among all individuals with cost less than $\lfloor \tilde{c} \rfloor$ in $P$ is chosen (see line 7 of Algorithm 3). Note, that there is always a valid individual to be selected once the search point $0^n$ has been included in the population.

For mutation we analyze standard bit mutation. Here we create $y$ by flipping each bit $x_i$ of $x$ with probability $\frac{1}{n}$. As standard bit mutations have a probability of roughly $1/e$ of not flipping any bit in a mutation step, we use the standard-bit-mutation-operator-plus outlined in Algorithm 4 in our experimental studies. We note that all theorectical results obtained in this article hold for standard bit mutations and standard bit mutatations plus.

As done in the area of runtime analysis (Doerr and Neumann, 2020), we measure the runtime of an algorithm by the number of fitness evaluations to achieve a desired goal. In particular, we are interested in a $\gamma$-approximation, i.e. a feasible solution $x$ for which $f(x) \geq \gamma \cdot f(x_{OPT})$ holds, where the parameter $\gamma \in [0,1]$ specifies the quality of the approximation and $x_{OPT}$ is an optimal solution for the considered problem.

We analyze the Sliding Window GSEMO algorithm with respect to the number of fitness evaluations and determine values of $t_{max}$ for which the algorithm has obtained good approximation with high probability, i.e. with probability $1 - o(1)$. The determined values of $t_{max}$ in our theorems in Section 5 are significantly

**Algorithm 2:** Sliding Window GSEMO (SW-GSEMO)

```
Set x = 0^n;
P ← {x};
t ← 0;
repeat
    t ← t + 1;
    Choose x = sliding-selection(P, t, t_max, B);
    Create y from x by mutation;
    if ∄ w ∈ P : w ≻ y then
        P ← (P \ {z ∈ P | y ⪰ z}) ∪ {y};
until t ≥ t_max;
```

**Algorithm 3:** sliding-selection$(P, t, t_{max}, B)$

```
if t ≤ t_max then
    ĉ ← (t/t_max) · B;
    P̂ = {x ∈ P | ⌊ĉ⌋ ≤ c(x) ≤ ⌈ĉ⌉};
    if P̂ ≠ ∅ then
        Choose x ∈ P̂ uniformly at random;
    else
        x = arg max_{z∈P}{f(z) | c(z) < ⌊ĉ⌋};
Return x;
```

lower than the bounds on the expected time to obtain the same approximations by the classical GSEMO algorithm.

### 3.1 Determistic Approximations

Friedrich and Neumann (2015) showed that the classical GSEMO finds a $(1-1/e)$-approximation to a monotone submodular function under a uniform constraint of size $B = r$ in expected time $O(n^2(\log n + r))$. A factor $n$ in their analysis stems from the fact that the population of GSEMO can have size up to $r \leq n$. Thanks to the sliding window approach, which only selects from a certain subset of the population, this factor $n$ does not appear in the following bound that we prove for SW-GSEMO. More precisely, the runtime guarantee in the following Theorem 1 is by a factor of $\Theta(n/\log n)$ better if $r \geq \log n$. For smaller $r$, we gain a factor of at least $\Theta(n/r)$.

Theorem 1. Consider the SW-GSEMO using the objective function $f_D$ with $t_{max} = 4ern \ln n$ on a monotone submodular function $f$ under a uniform constraint of size $r$. Then with probability $1 - o(1)$, the time until a $(1-1/e)$-approximation has been found is bounded from above by $t_{max} = O(nr \log n)$.

Proof. We follow the proof of Theorem 2 in Friedrich and Neumann (2015). As in that work, the aim is to include for every $j \in \{0, \ldots, r\}$ an element $x_j$ in the population such that

$$f(x_j) \geq (1 - (1 - 1/r)^j) f(\mathrm{OPT}) \text{ and } |x_j| \leq j, \tag{4}$$

where OPT is an optimal solution. If this holds, then the element $x_r$ satisfies the desired approximation ratio as $(1 - (1 - 1/r)^r) \geq (1 - 1/e)$. We also know from Friedrich and Neumann (2015) that the probability of mutating $x_j$ to $x_{j+1}$ is at least $1/(en)$ since it is sufficient to insert the element yielding the largest increase of $f$ and not to flip the rest of the bits.
We now consider a sequence of events leading to the inclusion of elements $x_j$ in the population for growing $j$. By definition of SW-GSEMO, element $x_0$ is in the population at time 0. Assume that element $x_j$, where $j \in \{0, \ldots, r-1\}$, is in the population $P$ at time $\tau_j := 4ejn \ln n$. We show now that $x_j$ is available for selection up to time

$$\tau_{j+1} - 1 = 4e(j+1)n \ln n - 1$$

since

$$\lfloor ((4e(j+1)n \ln n - 1)/t_{max}) \cdot r \rfloor = j.$$

Algorithm 4: Standard-bit-mutation-plus(x)

```
y ← x;
repeat
    Create y from x by flipping each bit x_i of x with probability 1/n.
until x ≠ y;
Return y;
```

The size of the subset population $\hat{P}$ that the algorithm selects $x_j$ from in the time interval $\{\tau_j, \ldots, \tau_{j+1} - 1\}$ is bounded from above by 2 since $\lceil \hat{c} \rceil - \lfloor \hat{c} \rfloor \leq 1$ and there is at most one non-dominated element for every constraint value. If there is no individual of value in $\{\lfloor \hat{c} \rfloor, \lceil \hat{c} \rceil\}$, then $x_j$ is the individual with highest function value having cost less than $\lfloor \hat{c} \rfloor$. Therefore, the probability of choosing $x_j$ and mutating it to $x_{j+1}$ is at least $1/(2en)$ from time $\tau_j$ until time $\tau_{j+1} - 1$, i. e., for a period of $4en \ln n$ steps, and the probability of this not happening is at most

$$\left(1 - \frac{1}{2en}\right)^{4en \ln n} \leq \frac{1}{n^2}.$$

By a union bound over at most $r$ elements to be included, the probability that there is a $j \in \{1, \ldots, r\}$ such that a desired solution $x_j$ fullfilling Equation 4 is not included by time $\tau_j$ is $O(1/n)$. □

### 3.2 Expected Approximation Qualities in Faster Time

Theorem 1 provides a deterministic bound of $1 - 1/e$ on the approximation quality of the solution that SW-GSEMO has achieved in time $t_{max}$ with high probability. Since the failure probability is $o(1)$, it is sufficient to re-run the algorithm an expected number of $1 + o(1)$ times until no failure occurs and the approximation quality of $1 - 1/e$ is achieved.

Instead of bounding the approximation quality with high probability, there are studies in the literature that derive guarantees for the expected value of the approximation quality in the given scenario. Crawford (2021) studied a multi-objective evolutionary algorithm called PO, which is similar to GSEMO, for achieving approximations on monotone submodular functions, but takes a different perspective on the approximation guarantee. Her main result, adjusted to the notation of the present article, states that PO finds a solution of expected approximation ratio $(1 - \epsilon)(1 - 1/e)$ in time at most $8enr \ln(1/\epsilon)$ for any $0 < \epsilon < 1$. Hence, if, e. g., $\epsilon$ is a small constant, the time is by an asymptotic factor of $\log n$ smaller than in Theorem 1 above.

Mirzasoleiman et al. (2015) presented an algorithm called stochastic greedy that, again in the scenario of optimizing monotone submodular functions under a cardinality constraint, computes a solution of expected approximation ratio $(1 - \epsilon)(1 - 1/e)$ in deterministic time $O(n \log(1/\epsilon))$. Note that the latter bound does not depend on the constraint value $r$.

Inspired by these prior works, the following theorem will analyze the SW-GSEMO and derive a bound on the expected approximation ratio depending on the deterministic time bound $t_{max}$ of the algorithm. If $t_{max} = 2ern \ln(1/\epsilon)$ for a constant $\epsilon > 0$, then an expected approximation of $1 - 1/e - \epsilon$ is obtained, which is similar in style to the result by Crawford mentioned above.

Theorem 2. Consider the SW-GSEMO using objective function $f_D$ on a monotone submodular function under a uniform constraint of size $r$. Then the expected approximation ratio of the solution at time $t_{max}$ is bounded from below by $1 - e^{-1} - e^{-t_{max}/(2ren)}$.

Proof. Our approach is partially inspired by the proof in Mirzasoleiman et al. (2015). However, their algorithm is based on a uniform choice of elements and a subsequent greedy procedure, which is different from the working principles of SW-GSEMO.

Let $A_i$ be the solution of cardinality at most $i$ available for selection when the sliding window chooses from solutions with best function value having at most $i$ elements. (Recall that the algorithm in this window chooses a solution of constraint value, i. e., cardinality $i$ if available; otherwise, it chooses a solution of largest constraint value less than $i$; note that there is at most one solution in the population for every constraint value.) We remark that $f(A_i)$ is random and that the following analysis will derive an approximation guarantee for the expected value $E(f(A_i))$ only.

Let $k \leq r$ be the number of elements of OPT not selected by $A_i$. Let $C_i$ be the event that an element in $\mathrm{OPT} \setminus A_i$ is included in a mutation of $A_i$. If all elements from $\mathrm{OPT} \setminus A_i$ were added to $A_i$ (possibly resulting in a set of too large cardinality), then the resulting function value would be at least $f(\mathrm{OPT})$ and hence the total increase at least $f(\mathrm{OPT}) - f(A_i)$. We claim that if only one such element is included, then on average over all such elements, the increase of function value is at least $(f(\mathrm{OPT}) - f(A_i))/k$. To show the claim, let

$b_1, \ldots, b_k \in \mathrm{OPT} \setminus A_i$ be all elements in OPT but not in $A_i$. Writing the total increase when adding the $b_j$ to $A_i$ after one another as a telescoping sum, we have

$$(f(A_i \cup \{b_1\}) - f(A_i)) + (f(A_i \cup \{b_1, b_2\}) - f(A_i \cup \{b_1\})) + (f(A_i \cup \{b_1, b_2, b_3\}) - f(A_i \cup \{b_1, b_2\})) + \cdots + (f(A_i \cup \{b_1, b_2, \ldots, b_k\}) - f(A_i \cup \{b_1, b_2, \ldots, b_{k-1}\})) = f(\mathrm{OPT}) - f(A_i).$$

By submodularity, we have for each $j \leq k$ that

$$f(A_i \cup \{b_1, b_2, \ldots, b_j\}) - f(A_i \cup \{b_1, b_2, \ldots, b_{j-1}\}) \leq f(A_i \cup \{b_j\}) - f(A_i).$$

Therefore,

$$\sum_{j=1}^{k} (f(A_i \cup \{b_j\}) - f(A_i)) \geq f(\mathrm{OPT}) - f(A_i),$$

so

$$\frac{1}{k} \sum_{j=1}^{k} (f(A_i \cup \{b_j\}) - f(A_i)) \geq \frac{f(\mathrm{OPT}) - f(A_i)}{k},$$

which establishes the claim. Hence, under $C_i$, the expected increase of function value is at least

$$\frac{f(\mathrm{OPT}) - f(A_i)}{k} \geq \frac{f(\mathrm{OPT}) - f(A_i)}{r}$$

since the element from $\mathrm{OPT} \setminus A_i$ is chosen uniformly at random. The probability of $C_i$ happening while the sliding window makes it possible to select $A_i$ is at least

$$1 - \left(1 - \frac{1}{2en}\right)^{t_{max}/r} \geq 1 - e^{-t_{max}/(2ren)},$$

so the expected function value of $A_{i+1}$ is at least

$$E(f(A_{i+1}) \mid A_i) \geq f(A_i) + \frac{1 - e^{-t_{max}/(2Ben)}}{r}(f(\mathrm{OPT}) - f(A_i)),$$

In the following, let $s = 1 - e^{-t_{max}/(2Ben)}$. After taking expectation over $A_i$ on both sides,

$$\begin{aligned} E(f(A_{i+1})) &\geq E(f(A_i)) + \frac{s}{r} E(f(\mathrm{OPT}) - f(A_i)) \\ &= \left(1 - \frac{s}{r}\right) E(f(A_i)) + \frac{s}{r} f(\mathrm{OPT}). \end{aligned}$$

We prove by induction that

$$E(f(A_i)) \geq \left(1 - \left(1 - \frac{s}{r}\right)^i\right) f(\mathrm{OPT}),$$

This is true for $i = 0$ since $f(A_0) = 0$ and establishes the base case. For the inductive step, we plug in the induction hypothesis in the above bound and have

$$\begin{aligned} E(f(A_{i+1})) &\geq \left(1 - \frac{s}{r}\right)\left(1 - \left(1 - \frac{s}{r}\right)^i\right) f(\mathrm{OPT}) + \frac{s}{r} f(\mathrm{OPT}) \\ &= \left(1 - \left(1 - \frac{s}{r}\right)^{i+1}\right) f(\mathrm{OPT}), \end{aligned}$$

which completes the induction. We conclude

$$E(f(A_r)) \geq \left(1 - \left(1 - \frac{s}{r}\right)^r\right) f(\mathrm{OPT}) \geq (1 - e^{-s}) f(\mathrm{OPT}).$$

Substituting $s$, we have

$$E(f(A_r)) \geq (1 - e^{-1} e^{e^{-t_{max}/(2ren)}}) f(\mathrm{OPT})$$

Using $e^x \leq 1 + 2x$ for $x \leq 1$ and $e^{-t_{max}/(2ren)} \leq 1$, this finally yields

$$E(f(A_r)) \geq (1 - e^{-1}(1 + 2e^{-t_{max}/(2ren)})) \geq (1 - e^{-1} - e^{-t_{max}/(2ren)}) f(\mathrm{OPT})$$

Hence, the expected approximation ratio $E(f(A_r))/f(\mathrm{OPT})$ is bounded as suggested. □

### 3.3 General Cost Constraints

We will show that the sliding window approach can also be used for more general optimization problems than submodular functions and also leads to improved runtime guarantees there. Specifically, we extend the approach to cover general cost constraints similar to the scenario studied in Qian et al. (2017). They use GSEMO (named POMC in their work) for the optimization scenario described in Definition 2, i. e., a monotone objective function $f\colon \{0,1\}^n \to \mathbb{R}^+$ under the constraint that a monotone cost function $c\colon \{0,1\}^n \to \mathbb{R}^+$ respects a cost bound $B$.

When transferring the set-up of Qian et al. (2017), we have introduce the following restriction: the image set of the cost function must be the positive integers, i. e., $c\colon \{0,1\}^n \to \mathbb{N}$. Otherwise, all cost values could be in a narrow real-valued interval so that the sliding window approach would not necessarily have any effect.

To formulate our result, we define the submodularity ratio in the same way as in earlier work (e. g., Qian et al. (2017)).

Definition 3. The submodularity ratio of a function $f\colon \{0,1\}^n \to \mathbb{R}^+$ is defined as

$$\alpha_f := \min_{\substack{x,y\in\{0,1\}^n, i\in\{1,\dots,n\} \\ x\leq y \wedge x_i = y_i = 0}} \frac{f(x \oplus e_i) - f(x)}{f(y \oplus e_i) - f(y)},$$

where $a \oplus e_i$ is the bit string obtained by setting bit $i$ of $a$ to 1.

The approximation guarantee proved for GSEMO in Qian et al. (2017) depends on $\alpha_f$ and the following quantity.

Definition 4. The minimal increase of a function $c\colon \{0,1\}^n \to \mathbb{N}$ is defined as

$$\delta_c = \min_{x\in\{0,1\}^n, i\in\{1,\dots,n\}, x_i=0} c(x \oplus e_i) - c(x)$$

In our analysis, we shall follow the assumption from Qian et al. (2017) that $\delta_c > 0$, i. e., adding an element to the solution will always increase the cost value.

Theorem 2 in Qian et al. (2017) depends on the submodularity ratio, minimum increase in function value and the maximum population size $P_{\max}$ reached by a run of GSEMO on the bi-objective function maximizing $f$ and minimizing a variant $\hat{c}$ of $c$ (explained below). It states that within $O(enBP_{\max}/\delta_{\hat{c}})$ iterations, GSEMO finds a solution $x$ such that $f(x) \geq \frac{\alpha_f}{2}(1 - 1/e^{\alpha_f}) \cdot f(\hat{x})$, where $\hat{x}$ is an optimal solution when maximizing $f$ with the original cost function $c$ but under a slightly increased budget with respect to $B$ (precisely defined in Equation (4) in Qian et al. (2017) and further detailed in Zhang and Vorobeychik (2016)). Our main result, formulated in the following theorem, is that the SW-GSEMO obtains solutions with the same quality guarantee in an expected time where the $P_{\max}$ factor does not appear, but an additional factor of $\delta_{\hat{c}} \ln(nB/\delta_{\hat{c}})$ which is usually much smaller.

As a detail in the formulation, the algorithm can be run with an approximation $\hat{c}$ of the original cost function $c$ that is by a certain factor $\phi(n)$ larger (i. e., $c(x) \leq \hat{c}(x) \leq \phi(n)c(x)$, see Zhang and Vorobeychik (2016) for details).

Theorem 3. Consider the problem of maximizing a monotone function $f\colon \{0,1\}^n \to \mathbb{R}^+$ under a monotone approximate cost function $\hat{c}\colon \{0,1\}^n \to \mathbb{N}_0$ with constraint $B$ and apply SW-GSEMO to the objective function $f_D(x) = (f_1(x), f_2(x))$, where $f_2(x) = \hat{c}(x)$ and

$$f_1(x) = \begin{cases} -\infty & \text{if } \hat{c}(x) > B \\ f(x) & \text{otherwise} \end{cases}.$$

If $t_{max} = 2enB\ln(nB/\delta_{\hat{c}})$, then with probability at least $1 - o(1)$ a solution of quality at least $\frac{\alpha_f}{2}(1 - 1/e^{\alpha_f}) \cdot f(\hat{x})$ is found in $t_{max}$ iterations, with $\delta_{\hat{c}} > 0$, $\alpha_f$ and $\hat{x}$ as defined in the two paragraphs preceding the theorem.

Proof. We follow the proof of Theorem 2 in Qian et al. (2017) and adapt it in a similar way to SW-GSEMO as we did in the proof of Theorem 1 above. According to the analysis in Qian et al. (2017), the approximation result is achieved by a sequence of steps choosing an individual of cost value at most $j$, where $j \in \{0, \dots, B-1\}$ (here we adapted the proof to the integrality of $\hat{c}$) and flipping a zero-bit to 1 yielding a certain minimum increase of $f$. More precisely, denoting by $P$ the current population of SW-GSEMO, we analyze the development of the quantity

$$J_{\max} = \max\{j \leq B-1 \mid \exists x \in P\colon \hat{c}(x) \leq j \wedge f(x) \geq (1 - e^{-\alpha_f j/B}) \cdot f(\hat{x})\}.$$

If the search point $0^n$ which is Pareto optimal is included in the population $P$ then $J_{\max}$ is at least 0 as $\hat{c}(0^n) = 0$ and $f(0^n) = (1 - e^0)f(\hat{x}) = 0$. Let $x^*$ be the solution that meets the requirement for $J_{\max}$ in the current population $P$. The value of $J_{\max}$ increases monotonically as each individual that dominates the individual $x^*$ with the largest $j$-value in the population can only increase the value of $J_{\max}$. We recall the key induction step used in previous works (Qian et al., 2017; Neumann and Neumann, 2025) that allows to track the progress of the algorithm. Choosing solution $x^*$ for mutation and introducing the element $e^*$ (not selected by $x^*$) with the largest marginal gain leads to a solution $y$ with $f(y) - f(x^*) \geq \frac{\alpha_f(\hat{c}(y)-\hat{c}(x^*))}{B} \cdot (f(\hat{x}) - f(x^*))$ (Lemma 3 in Qian et al. (2017)) as $f$ is monotone and $\alpha_f$-submodular. Therefore, we have

$$\begin{aligned}
f(y) &\geq f(x^*) + \frac{\alpha_f(\hat{c}(y) - \hat{c}(x^*))}{B} \cdot (f(\hat{x}) - f(x^*)) \\
&\geq f(x^*) + \frac{\alpha_f \delta_{\hat{c}}}{B} \cdot (f(\hat{x}) - f(x^*)) \\
&\geq \left(1 - \frac{\alpha_f \delta_{\hat{c}}}{B}\right) f(x^*) + \frac{\alpha_f \delta_{\hat{c}}}{B} \cdot f(\hat{x}) \\
&\geq \left(1 - \frac{\alpha_f \delta_{\hat{c}}}{B}\right) \left(1 - e^{-\alpha_f j/B}\right) \cdot f(\hat{x}) + \frac{\alpha_f \delta_{\hat{c}}}{B} \cdot f(\hat{x}) \\
&= \left(1 - \left(1 - \frac{\alpha_f \delta_{\hat{c}}}{B}\right) \cdot e^{-\alpha_f j/B}\right) \cdot f(\hat{x}) \\
&\geq \left(1 - e^{-\frac{\alpha_f \delta_{\hat{c}}}{B}} \cdot e^{-\alpha_f j/B}\right) \cdot f(\hat{x}) \\
&= \left(1 - e^{-\alpha_f (j+\delta_{\hat{c}})/B}\right) \cdot f(\hat{x}).
\end{aligned}$$

Hence, including $y$ into population in the case that $y$ is feasible increases $J_{\max}$ by at least $\delta_{\hat{c}}$. If $y$ is not feasible then we have $\hat{c}(y) > B$ and hence $f(y) \geq (1 - e^{-\alpha_f}) \cdot f(\hat{x})$. In this case, we consider the feasible solution $z$ with the highest function value containing a single element only. Note that the function value of $z$ is at least as high as the function value of the solution containing element $e^*$ only and that $z$ can be obtained from the search point $0^n$ by flipping a specific single bit. We have $f(y) \leq f(x^*)+f(z)/\alpha_f \leq (f(x^*)+f(z))/\alpha_f$ as $f$ is $\alpha_f$-submodular and $\alpha_f \in [0, 1]$. Hence, $\max\{f(x^*), f(z)\} \geq \frac{\alpha_f}{2} \cdot (1 - e^{-\alpha_f})f(\hat{x})$.
We now show that $x^*$ is chosen and the appropriate zero-bit is flipped with high probability for each of at most $B/\delta_{\hat{c}}$ values that $J_{\max}$ can take in this sequence of successful steps. By definition of the set $\hat{P}$ in SW-GSEMO, element $x^*$ from the population is available for selection for a period of

$$2en \ln(nB/\delta_{\hat{c}})$$

steps, more precisely between time

$$2enJ_{\max} \ln(nB/\delta_{\hat{c}})$$

and

$$2en(J_{\max} + 1) \ln(nB/\delta_{\hat{c}}) - 1.$$

Moreover, by the same arguments as in the proof of Theorem 1, it holds that $|\hat{P}| \leq 2$ for the subset population $\hat{P}$ that the algorithm selects from, and in the case of $\hat{P} = \emptyset$ the algorithm will chose an individual of highest objective function value of cost at most $J_{\max}$.
Hence, the probability of a success is at least $1/(2en)$ for each step within the mentioned period. Therefore, the probability of not having a successful step with respect to $x^*$ is bounded from above by

$$\left(1 - \frac{1}{2en}\right)^{2en \ln(nB/\delta_{\hat{c}})} \leq \frac{1}{n(B/\delta_{\hat{c}})}.$$

By a union bound over the at most $B/\delta_{\hat{c}}$ required successes, the probability of missing at least one success is at most $1/n$, so altogether the desired solution quality is achieved with probability at least $1-1/n = 1-o(1)$. □

The results from Theorems 1 and 3 are just two examples of a runtime result following an inductive sequence of improving steps based on constraint cost value. In the literature, there are further analyses of GSEMO following a similar approach (e. g., Qian et al. (2023)), which we believe can be transferred to our sliding window approach to yield improved runtime guarantees.

### 3.4 Experimental investigations

We now carry out experimental investigations of the new Sliding Window GSEMO approach and compare it against the standard GSEMO which has been used in many theoretical and experimental studies on submodular optimization. We initialize the algorithms with the search point $0^n$ as done in the investigations in Qian et al. (2017).

#### 3.4.1 Experimental setup

We consider the maximum coverage problem in graphs which is one of the best known NP-hard submodular combinatorial optimization problems. Given an undirected graph $G = (V, E)$ with node set $V = \{v_1, \ldots, v_n\}$, we denote by $N(v_i)$ the set of nodes containing $v_i$ and its neighbours in $G$. Each node $v_i$ has a cost $c(v_i)$ and the goal is to select a set of nodes indicated by $x \in \{0,1\}^n$ such that the number of nodes covered by $x$ given as

$$\text{Coverage}(x) = \left| \bigcup_{i=1}^{n} N(v_i) x_i \right|$$

is maximized under the condition that the cost of the selected nodes does not exceed a given bound B, i. e.

$$\sum_{i=1}^{n} c(v_i) x_i \leq B$$

holds.

Previous studies investigated GSEMO either considered relatively small graphs or small budgets on the constraints that only allowed a small number of trade-offs to be produced. We consider sparse graphs from the network data repository (Rossi and Ahmed, 2015) as dense graphs are usually easy to cover with just a small number of nodes and do not pose a challenge when considering the maximum coverage problem. We use the graphs ca-CSphd, ca-GrQc, Erdos992, ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 1882, 4158, 6100, 11204, 17903, 21363 nodes, respectively. Note, that especially the large graphs with more than 10000 nodes are pushing the limits for both algorithms when considering the given fitness evaluation budgets. To match our theoretical investigations for the uniform constraint, we consider the uniform setting where $c(v_i) = 1$ holds for any node $v_i$ in the given graph. Note that the uniform setting implies that the number of trade-offs in the two objectives is at most $B + 1$. In order to investigate the behaviour of the GSEMO approaches when there are more possible trade-offs, we investigate for each graph the following random setting. For an instance in the random setting, the cost of each node $v_i$ is chosen independently of the other uniformly at random in the interval $[0.5, 1.5]$. Note that the expected cost of each node in the random setting is 1. This setup allows us to work with the same bounds for the uniform and random setting. We use

$$B = \log_2 n, \sqrt{n}, \lfloor n/20 \rfloor, \lfloor n/10 \rfloor$$

such that the bounds scale with the given number of nodes in the considered graphs. Note that for the uniform case, all costs are integers and the effective bounds are the stated bounds rounded down to the next smaller integer. We consider the performance of both algorithms when given them a budget of

$$t_{max} = 100000, 500000, 1000000$$

fitness evaluations. Note, that especially for the large graphs and for the larger values of $B$ this is pushing the limits in terms of the size of the instances as the fitness evaluations budgets are much less then what is stated in the upper bounds on $t_{max}$ in Theorems 1 and 3. For every setting, we carry out 30 independent runs. We show the coverage values obtained in Table 1. In this table, we report the mean, standard deviation and the $p$-value (with 3 decimal places) obtained by the rank-based Mann Whitney-U test. We call a result statistically significant if the $p$-value is at most 0.05. To gain additional insights, we present the mean final population sizes among the 30 runs for each setting in Table 2.

#### 3.4.2 Experimental results

The results for all considered graphs, constraint bounds, fitness evaluation budgets in the uniform and random cost setting are given in Table 1. The best results are highlighted in bold. Overall, SW-GSEMO is clearly outperforming GSEMO for almost all settings and almost all results are statistically significant. The only instances where both algorithms perform equally good are when considering the small constraint bounds of $\log_2 n$ or $\sqrt{n}$ for ca-CSphd and $\log_2 n$ for ca-GrQc, Erdos992, ca-HepPh. An important observation is that SW-GSEMO with only 100,000 iterations is already much better than GSEMO with 1,000,000 fitness evaluations if the constraint bound is not too small.

Considering the larger constraint bounds $\lfloor n/20 \rfloor, \lfloor n/10 \rfloor$ for the graphs, we can see that SW-GSEMO is significantly outperforming GSEMO. The difference in terms of coverage values between the two algorithms

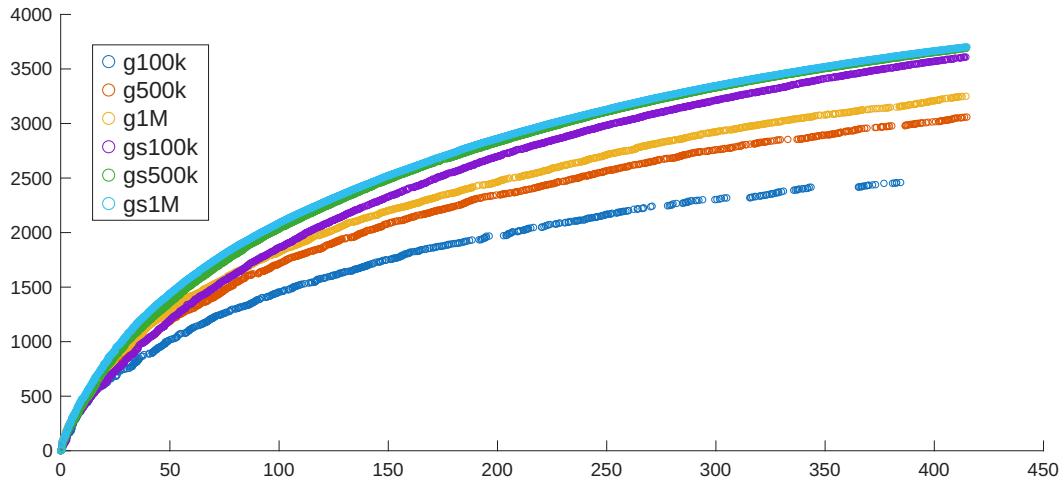


Figure 1: Example final trade-offs for the graph ca-GrQc (4158 nodes) with randomly chosen costs and budget $B = 415$ obtained by GSEMO (g) and Sliding Window GSEMO (gs) for runs with $100k$, $500k$, and $1M$ iterations.

becomes even more pronounced when considering the random instances compared to the uniform ones. Considering the graph ca-HepPh, we can see that standard GSEMO obtains better results than SW-GSEMO when considering the constraint bound of $\log_2 n$ in the uniform setting while SW-GSEMO is significantly outperforming GSEMO in all other settings. Looking at the results for the two largest graphs ca-AstroPh and ca-CondMat, we can see that SW-GSEMO significantly outperforms GSEMO in all considered settings. Comparing the uniform and the random setting, we can see that the coverage values obtained by GSEMO in the random setting are much smaller than in the uniform setting when considering the three larger graphs ca-HepPh, ca-AstroPh, ca-CondMat. The only exception are some results for the small bound $B = \log_2 n$. We attribute this deterioration of GSEMO to the larger number of trade-offs per cost unit encountered in the optimization process which significantly slows down GSEMO making progress towards the constraint bound. In contrast to this, the coverage values obtained by SW-GSEMO in the uniform and random setting when considering $t_{max} = 500000, 1000000$ are quite similar and sometimes higher for the random setting. This indicates that the sliding window selection keeps steady progress for the random setting and is not negatively impacted by the larger number of trade-offs in the random setting.

In order to better understand the performance difference between the algorithms, we examine the trade-offs produced by the algorithms. Figure 1 illustrates the final set of trade-offs for GSEMO and SW-GSEMO with respect to cost and coverage values for the different number of fitness evaluations considered. The three lower trade-off fronts depicted in blue, red, and yellow are obtaining running GSEMO with 100,000, 500,000, and 1,000,000 iterations. The three higher trade-off fronts shown in purple, green, and light blue have been obtained by SW-GSEMO in 100000, 500000, and 1000000 iterations. It can be observed that the fronts obtained by SW-GSEMO are significantly better than the ones obtained by GSEMO. The fronts for SW-GSEMO with 500000 and 1000000 iterations are very similar while the results for 100000 are already better than the ones obtained by GSEMO with 1000000 iterations. This matches the behaviour that can already be observed for many results shown in Table 1. Furthermore, it can be seen that GSEMO has already difficulties obtaining a solution with cost close to the constraint bound when using the smallest considered budget of 100000 fitness evaluations.

In Table 2, we show the average number of trade-offs given by the final populations of the two algorithms for the 30 runs of each setting. We first examine the uniform setting. The budgets for our experiments are chosen small enough such that not all nodes can be covered by any solution. Therefore, in the ideal case, both algorithms would produce $B+1$ trade-offs in the uniform setting. It can be observed that this is roughly happening for the two smallest graphs ca-CSphd, ca-GrQc. For the remaining 4 graphs, GSEMO produces significantly less points when considering the constraint bound $\lfloor n/20 \rfloor, \lfloor n/10 \rfloor$ while the number of trade-offs obtained by SW-GSEMO is close to $B + 1$ is most uniform settings. Considering the random setting, we can see that the number of trade-offs produced by SW-GSEMO is significantly higher than for GSEMO. In the case of large graphs, the number of trade-offs produced is up to four times larger than the number of trade-offs produced by GSEMO, e.g. for graph ca-CondMat and $B = 2136$. Overall this suggests that the larger number of trade-offs produced in a systematic way by the sliding window approach significantly contributes to the superior performance of SW-GSEMO.

| | | | Uniform | | | | | Random | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | GSEMO | | SW-GSEMO | | | GSEMO | | SW-GSEMO | | |
| Graph | $B$ | $t_{\max}$ | Mean | Std | Mean | Std | $p$-value | Mean | Std | Mean | Std | $p$-value |
| ca-CSphd | 10 | 100000 | 222 | 0.183 | 222 | 0.000 | 0.824 | 244 | 12.904 | 254 | 13.278 | 0.006 |
| | 10 | 500000 | 222 | 0.000 | 222 | 0.000 | 1.000 | 257 | 13.962 | 257 | 13.833 | 0.929 |
| | 10 | 1000000 | 222 | 0.000 | 222 | 0.000 | 1.000 | 258 | 13.938 | 258 | 13.788 | 0.935 |
| | 43 | 100000 | 568 | 5.756 | 600 | 0.535 | 0.000 | 539 | 13.808 | 624 | 14.043 | 0.000 |
| | 43 | 500000 | 600 | 0.254 | 600 | 0.000 | 0.657 | 615 | 13.150 | 629 | 13.518 | 0.000 |
| | 43 | 1000000 | 600 | 0.000 | 600 | 0.000 | 1.000 | 626 | 13.588 | 630 | 13.554 | 0.225 |
| | 94 | 100000 | 823 | 6.150 | 928 | 0.407 | 0.000 | 779 | 12.898 | 957 | 12.190 | 0.000 |
| | 94 | 500000 | 924 | 1.570 | 928 | 0.000 | 0.000 | 915 | 12.252 | 962 | 12.156 | 0.000 |
| | 94 | 1000000 | 928 | 0.254 | 928 | 0.000 | 0.657 | 946 | 11.660 | 963 | 12.171 | 0.000 |
| | 188 | 100000 | 1087 | 11.676 | 1279 | 0.770 | 0.000 | 1036 | 12.868 | 1334 | 12.588 | 0.000 |
| | 188 | 500000 | 1256 | 2.809 | 1280 | 0.254 | 0.000 | 1235 | 12.964 | 1341 | 12.936 | 0.000 |
| | 188 | 1000000 | 1278 | 1.119 | 1280 | 0.000 | 0.000 | 1291 | 11.671 | 1341 | 12.831 | 0.000 |
| ca-GrQc | 12 | 100000 | 490 | 8.798 | 506 | 5.128 | 0.000 | 535 | 17.964 | 596 | 22.543 | 0.000 |
| | 12 | 500000 | 509 | 2.539 | 510 | 0.000 | 0.046 | 601 | 23.067 | 619 | 20.085 | 0.004 |
| | 12 | 1000000 | 510 | 0.000 | 510 | 0.000 | 1.000 | 615 | 22.523 | 623 | 21.385 | 0.169 |
| | 64 | 100000 | 1320 | 15.662 | 1512 | 6.358 | 0.000 | 1281 | 24.326 | 1638 | 25.821 | 0.000 |
| | 64 | 500000 | 1490 | 8.904 | 1527 | 3.910 | 0.000 | 1516 | 25.832 | 1690 | 26.298 | 0.000 |
| | 64 | 1000000 | 1512 | 6.244 | 1529 | 2.966 | 0.000 | 1594 | 24.659 | 1700 | 26.624 | 0.000 |
| | 207 | 100000 | 2151 | 20.651 | 2747 | 7.867 | 0.000 | 2044 | 25.039 | 2837 | 21.088 | 0.000 |
| | 207 | 500000 | 2530 | 13.505 | 2775 | 4.291 | 0.000 | 2450 | 18.991 | 2916 | 20.257 | 0.000 |
| | 207 | 1000000 | 2655 | 9.295 | 2778 | 2.755 | 0.000 | 2600 | 13.956 | 2926 | 19.962 | 0.000 |
| | 415 | 100000 | 2704 | 24.887 | 3573 | 7.840 | 0.000 | 2407 | 51.554 | 3627 | 13.286 | 0.000 |
| | 415 | 500000 | 3171 | 13.424 | 3616 | 3.309 | 0.000 | 3047 | 19.085 | 3701 | 12.535 | 0.000 |
| | 415 | 1000000 | 3339 | 10.258 | 3621 | 2.177 | 0.000 | 3232 | 13.908 | 3710 | 12.891 | 0.000 |
| Erdos992 | 12 | 100000 | 584 | 8.094 | 602 | 2.385 | 0.000 | 635 | 26.211 | 747 | 34.425 | 0.000 |
| | 12 | 500000 | 603 | 1.837 | 604 | 0.730 | 0.016 | 751 | 36.183 | 782 | 37.781 | 0.003 |
| | 12 | 1000000 | 604 | 0.254 | 604 | 0.000 | 0.657 | 774 | 37.049 | 786 | 37.919 | 0.176 |
| | 78 | 100000 | 1835 | 35.766 | 2453 | 5.123 | 0.000 | 1658 | 31.957 | 2511 | 42.895 | 0.000 |
| | 78 | 500000 | 2345 | 18.151 | 2472 | 0.791 | 0.000 | 2169 | 35.736 | 2625 | 43.764 | 0.000 |
| | 78 | 1000000 | 2438 | 7.439 | 2473 | 0.254 | 0.000 | 2366 | 41.017 | 2634 | 43.701 | 0.000 |
| | 305 | 100000 | 2862 | 54.832 | 4711 | 10.543 | 0.000 | 2547 | 60.705 | 4623 | 30.507 | 0.000 |
| | 305 | 500000 | 3824 | 27.333 | 4772 | 1.159 | 0.000 | 3534 | 30.716 | 4789 | 21.852 | 0.000 |
| | 305 | 1000000 | 4201 | 20.239 | 4775 | 1.042 | 0.000 | 3898 | 28.104 | 4799 | 21.513 | 0.000 |
| | 610 | 100000 | 3076 | 46.668 | 5252 | 2.848 | 0.000 | 2553 | 57.540 | 5325 | 9.938 | 0.000 |
| | 610 | 500000 | 4428 | 31.694 | 5263 | 0.679 | 0.000 | 4017 | 50.324 | 5376 | 8.608 | 0.000 |
| | 610 | 1000000 | 4791 | 22.216 | 5264 | 0.305 | 0.000 | 4516 | 33.380 | 5378 | 8.794 | 0.000 |
| ca-HepPh | 13 | 100000 | 1687 | 34.310 | 1800 | 25.713 | 0.000 | 1750 | 53.601 | 1966 | 52.019 | 0.000 |
| | 13 | 500000 | 1830 | 27.572 | 1844 | 11.230 | 0.018 | 1980 | 50.385 | 2111 | 41.582 | 0.000 |
| | 13 | 1000000 | 1854 | 15.539 | 1840 | 4.173 | 0.000 | 2058 | 50.351 | 2143 | 53.197 | 0.000 |
| | 105 | 100000 | 3740 | 48.996 | 4641 | 26.834 | 0.000 | 3598 | 31.674 | 4820 | 56.886 | 0.000 |
| | 105 | 500000 | 4309 | 28.317 | 4777 | 15.980 | 0.000 | 4238 | 40.329 | 5114 | 54.093 | 0.000 |
| | 105 | 1000000 | 4516 | 20.372 | 4799 | 14.067 | 0.000 | 4490 | 43.562 | 5172 | 48.838 | 0.000 |
| | 560 | 100000 | 5909 | 73.941 | 8543 | 19.580 | 0.000 | 5108 | 86.715 | 8639 | 38.123 | 0.000 |
| | 560 | 500000 | 7092 | 29.683 | 8798 | 12.650 | 0.000 | 6802 | 36.876 | 9060 | 31.097 | 0.000 |
| | 560 | 1000000 | 7527 | 24.174 | 8825 | 6.897 | 0.000 | 7249 | 32.562 | 9126 | 31.146 | 0.000 |
| | 1120 | 100000 | 5919 | 81.766 | 10225 | 19.108 | 0.000 | 5098 | 92.907 | 10290 | 19.428 | 0.000 |
| | 1120 | 500000 | 8148 | 55.111 | 10492 | 8.719 | 0.000 | 7209 | 62.132 | 10640 | 16.077 | 0.000 |
| | 1120 | 1000000 | 8795 | 30.765 | 10525 | 5.148 | 0.000 | 8138 | 62.450 | 10689 | 14.832 | 0.000 |
| ca-AstroPh | 14 | 100000 | 2594 | 85.426 | 2881 | 43.102 | 0.000 | 2598 | 96.079 | 3026 | 83.889 | 0.000 |
| | 14 | 500000 | 2914 | 31.062 | 2975 | 6.056 | 0.000 | 3016 | 64.712 | 3322 | 95.037 | 0.000 |
| | 14 | 1000000 | 2962 | 12.480 | 2980 | 2.564 | 0.000 | 3195 | 77.509 | 3395 | 91.802 | 0.000 |
| | 133 | 100000 | 6484 | 77.132 | 8355 | 58.234 | 0.000 | 6221 | 82.193 | 8559 | 70.305 | 0.000 |
| | 133 | 500000 | 7551 | 51.042 | 8710 | 18.685 | 0.000 | 7362 | 61.676 | 9218 | 63.689 | 0.000 |
| | 133 | 1000000 | 7968 | 40.202 | 8751 | 18.720 | 0.000 | 7817 | 62.797 | 9369 | 57.241 | 0.000 |
| | 895 | 100000 | 9511 | 120.361 | 15029 | 32.420 | 0.000 | 8170 | 122.610 | 15104 | 38.804 | 0.000 |
| | 895 | 500000 | 12360 | 60.272 | 15608 | 16.070 | 0.000 | 11387 | 100.119 | 15868 | 31.193 | 0.000 |
| | 895 | 1000000 | 13017 | 35.102 | 15691 | 11.214 | 0.000 | 12490 | 39.435 | 16013 | 25.022 | 0.000 |
| | 1790 | 100000 | 9502 | 97.540 | 17036 | 20.145 | 0.000 | 8137 | 104.415 | 17148 | 19.337 | 0.000 |
| | 1790 | 500000 | 12750 | 88.401 | 17471 | 9.200 | 0.000 | 11374 | 104.602 | 17688 | 12.920 | 0.000 |
| | 1790 | 1000000 | 14103 | 54.069 | 17525 | 8.327 | 0.000 | 12743 | 112.303 | 17764 | 11.854 | 0.000 |
| ca-CondMat | 14 | 100000 | 1514 | 56.623 | 1748 | 51.276 | 0.000 | 1411 | 76.166 | 1762 | 64.178 | 0.000 |
| | 14 | 500000 | 1802 | 25.451 | 1856 | 2.793 | 0.000 | 1773 | 62.028 | 2018 | 76.876 | 0.000 |
| | 14 | 1000000 | 1846 | 8.628 | 1857 | 1.925 | 0.000 | 1912 | 69.264 | 2065 | 76.075 | 0.000 |
| | 146 | 100000 | 4388 | 71.953 | 6655 | 66.632 | 0.000 | 4106 | 91.413 | 6642 | 69.760 | 0.000 |
| | 146 | 500000 | 5585 | 61.992 | 7056 | 12.468 | 0.000 | 5264 | 65.588 | 7413 | 74.394 | 0.000 |
| | 146 | 1000000 | 6092 | 50.561 | 7091 | 11.059 | 0.000 | 5776 | 64.557 | 7556 | 73.427 | 0.000 |
| | 1068 | 100000 | 7187 | 130.346 | 15763 | 35.555 | 0.000 | 5779 | 149.352 | 15759 | 55.651 | 0.000 |
| | 1068 | 500000 | 11334 | 67.041 | 16727 | 17.535 | 0.000 | 9655 | 129.937 | 17041 | 44.701 | 0.000 |
| | 1068 | 1000000 | 12364 | 69.321 | 16843 | 13.705 | 0.000 | 11533 | 79.858 | 17296 | 43.174 | 0.000 |
| | 2136 | 100000 | 7211 | 133.166 | 19164 | 33.256 | 0.000 | 5823 | 137.877 | 19229 | 47.788 | 0.000 |
| | 2136 | 500000 | 11556 | 115.359 | 20089 | 13.959 | 0.000 | 9675 | 135.492 | 20307 | 24.633 | 0.000 |
| | 2136 | 1000000 | 13652 | 84.783 | 20217 | 11.957 | 0.000 | 11632 | 96.193 | 20489 | 25.097 | 0.000 |

Table 1: Maximum coverage scores obtained by GSEMO and SW-GSEMO

## 4 3-Objective Sliding Window GSEMO

In this section, we define the algorithmic framework incorporating sliding window selection into 3-objective optimization problems under constraints. It combines the 3-objective problem formulation from Neumann and Witt (2023a), where the underlying problem is 2-objective and a constraint is converted to a helper objective, with the 2-objective window approach introduction in Section 3 where the problem is single-objective and the constraint is converted to a helper objective. More precisely, sliding window is based on the observation that several problems under uniform constraints can be solved by iterating over increasing constraint values and optimizing the actual objectives for each fixed constraint value.
We recall the 3-objective formulation

$$f_{3D}(x) = (\mu(x), v(x), c(x))$$

given in Section 2.2.1. We consider problems defined on bit strings $x \in \{0,1\}^n$ involving the minimization of two objective functions $\mu(x), v(x) \colon \{0,1\}^n \to \mathbb{R}_0^+$ and an integer-valued constraint function $c(x) \colon \{0,1\}^n \to \mathbb{N}$ (together with a bound $B$) that should be maximized to reach a feasible solution, i.e. a solution $x$ with $c(x) \geq B$. The constraint function $c(x)$ and bound $B$ form the basis for our sliding window selection. Our new approach called SW-GSEMO3D is shown in Algorithm 5 (which will later be extended to Algorithm 7 as

| | | | Uniform | | Random | |
|---|---|---|---|---|---|---|
| Graph | $B$ | $t_{\max}$ | G | SWG | G | SWG |
| ca-CSphd | 10 | 100000 | 11 | 11 | 73 | 94 |
| | 10 | 500000 | 11 | 11 | 125 | 131 |
| | 10 | 1000000 | 11 | 11 | 133 | 132 |
| | 43 | 100000 | 44 | 44 | 172 | 280 |
| | 43 | 500000 | 44 | 44 | 287 | 432 |
| | 43 | 1000000 | 44 | 44 | 390 | 476 |
| | 94 | 100000 | 94 | 95 | 281 | 500 |
| | 94 | 500000 | 95 | 95 | 422 | 692 |
| | 94 | 1000000 | 95 | 95 | 540 | 763 |
| | 188 | 100000 | 180 | 189 | 404 | 784 |
| | 188 | 500000 | 189 | 189 | 591 | 988 |
| | 188 | 1000000 | 189 | 189 | 734 | 1071 |
| ca-GrQc | 12 | 100000 | 13 | 13 | 89 | 115 |
| | 12 | 500000 | 13 | 13 | 142 | 191 |
| | 12 | 1000000 | 13 | 13 | 185 | 235 |
| | 64 | 100000 | 65 | 65 | 261 | 437 |
| | 64 | 500000 | 65 | 65 | 372 | 688 |
| | 64 | 1000000 | 65 | 65 | 448 | 851 |
| | 207 | 100000 | 200 | 208 | 493 | 1020 |
| | 207 | 500000 | 207 | 208 | 710 | 1477 |
| | 207 | 1000000 | 208 | 208 | 841 | 1698 |
| | 415 | 100000 | 347 | 414 | 611 | 1512 |
| | 415 | 500000 | 400 | 416 | 967 | 1993 |
| | 415 | 1000000 | 409 | 416 | 1134 | 2227 |
| Erdos992 | 12 | 100000 | 13 | 13 | 76 | 104 |
| | 12 | 500000 | 13 | 13 | 129 | 187 |
| | 12 | 1000000 | 13 | 13 | 181 | 248 |
| | 78 | 100000 | 78 | 79 | 237 | 394 |
| | 78 | 500000 | 79 | 79 | 330 | 693 |
| | 78 | 1000000 | 79 | 79 | 393 | 911 |
| | 305 | 100000 | 253 | 304 | 441 | 981 |
| | 305 | 500000 | 291 | 306 | 668 | 1491 |
| | 305 | 1000000 | 298 | 306 | 777 | 1894 |
| | 610 | 100000 | 296 | 589 | 440 | 1431 |
| | 610 | 500000 | 483 | 610 | 818 | 1766 |
| | 610 | 1000000 | 522 | 611 | 1014 | 2066 |
| ca-HepPh | 13 | 100000 | 14 | 14 | 91 | 116 |
| | 13 | 500000 | 14 | 14 | 125 | 160 |
| | 13 | 1000000 | 14 | 14 | 143 | 195 |
| | 105 | 100000 | 105 | 106 | 344 | 628 |
| | 105 | 500000 | 106 | 106 | 489 | 876 |
| | 105 | 1000000 | 106 | 106 | 553 | 1073 |
| | 560 | 100000 | 408 | 558 | 634 | 2109 |
| | 560 | 500000 | 521 | 560 | 1176 | 2837 |
| | 560 | 1000000 | 543 | 561 | 1364 | 3287 |
| | 1120 | 100000 | 409 | 1104 | 644 | 3150 |
| | 1120 | 500000 | 813 | 1115 | 1306 | 3922 |
| | 1120 | 1000000 | 951 | 1117 | 1718 | 4333 |
| ca-AstroPh | 14 | 100000 | 15 | 15 | 94 | 120 |
| | 14 | 500000 | 15 | 15 | 125 | 159 |
| | 14 | 1000000 | 15 | 15 | 142 | 193 |
| | 133 | 100000 | 132 | 134 | 404 | 802 |
| | 133 | 500000 | 134 | 134 | 565 | 1075 |
| | 133 | 1000000 | 134 | 134 | 663 | 1279 |
| | 895 | 100000 | 416 | 889 | 653 | 3050 |
| | 895 | 500000 | 764 | 895 | 1372 | 3946 |
| | 895 | 1000000 | 818 | 895 | 1755 | 4467 |
| | 1790 | 100000 | 413 | 1740 | 643 | 4390 |
| | 1790 | 500000 | 848 | 1770 | 1366 | 5265 |
| | 1790 | 1000000 | 1121 | 1779 | 1830 | 5673 |
| ca-CondMat | 14 | 100000 | 15 | 15 | 87 | 107 |
| | 14 | 500000 | 15 | 15 | 111 | 132 |
| | 14 | 1000000 | 15 | 15 | 124 | 165 |
| | 146 | 100000 | 144 | 147 | 410 | 816 |
| | 146 | 500000 | 147 | 147 | 572 | 1083 |
| | 146 | 1000000 | 147 | 147 | 662 | 1295 |
| | 1068 | 100000 | 424 | 1062 | 650 | 3732 |
| | 1068 | 500000 | 864 | 1068 | 1437 | 5016 |
| | 1068 | 1000000 | 952 | 1069 | 1929 | 5758 |
| | 2136 | 100000 | 425 | 2098 | 655 | 5587 |
| | 2136 | 500000 | 906 | 2126 | 1428 | 7097 |
| | 2136 | 1000000 | 1228 | 2132 | 1953 | 7857 |

Table 2: Final number of trade-off solutions obtained by GSEMO (G) and SW-GSEMO (SWG)

explained below). The sliding window selection in Algorithm 6 will be used as a module in SW-GSEMO3D and chooses from its current population $P$, which is the first parameter of the algorithm. The idea is to select only from a subpopulation of constraint values in a specific interval determined by the maximum constrained value $B$, the current generation $t$, the maximum number of iterations of the algorithm $t_{max}$, and further parameters. In the simplest case (where the remaining parameters are set to $a = 1$, $c_{\max} = -1$ and $t_{frac} = 1$), the time interval $[1, t_{max}]$ is uniformly divided into $B$ time intervals in which individuals from the subpopulation having constraint values in the interval $[\lfloor \tilde{c} \rfloor - \text{std}, \lceil \hat{c} \rceil + \text{std}]$, where $\hat{c} = t/t_{max})B$, are selected. Here, std $\geq 0$ is a deviation from the interval defined by $\hat{c}$ in line 1 of Algorithm 6 that allows selection from a larger interval, which is another heuristic component that we will investigate in Section 4.2. Moreover, as not all problems may benefit from selecting according to the specific interval order, the calls to Algorithm 6 resort to selection from the interval $[B - \text{std}, B]$ for the last $(1 - t_{frac})t_{max}$ steps. Finally, since making progress may become increasingly difficult for increasing constraint values, the selection provides the parameter $a$ which will allow time intervals of varying length for the different constraint values to choose from. If $a < 1$, the time allocated to choosing from a specific constraint value (interval) increases with the constraint value; if $a > 1$, it decreases with the constraint value. Lines 8–10 of the algorithm make sure that solutions with too low constraint value (less than $\ell$), but not equaling the parameter $c_{\max}$ are permanently removed from the population. Line 11 confines the population to select from to the desired window of

**Algorithm 5:** Sliding Window GSEMO3D (SW-GSEMO3D)

```
1  Choose initial solution x ∈ {0,1}^n;
2  Set t_0 ← −1;
3  P ← {x};
4  Compute f_3D(x) = (μ(x), v(x), c(x));
5  t ← 1;
6  μ_min ← μ(x);
7  if μ_min = 0 then
8      t_0 ← t;
9  repeat
10     if (t_0 = −1) ∧ (t ≤ t_max) then
11         x ← arg min{μ(z) | z ∈ P} (breaking ties arbitrarily)
12     else
13         x = sliding-selection3D(P, t − t_0, t_max − t_0, 0, B, 1, 1, −1);
14     Create y from x by mutation;
15     Compute f_3D(y) = (μ(y), v(y), c(y));
16     if μ(y) < μ_min then
17         μ_min ← μ(y);
18     if (t_0 = −1) ∧ (μ_min = 0) then
19         t_0 ← t;
20     if ∄ w ∈ P : w ≻ y then
21         P ← (P \ {z ∈ P | y ⪰ z}) ∪ {y};
22     t ← t + 1;
23 until t ≥ t_max;
```

**Algorithm 6:** sliding-selection3D($P, t, t_{max}, std, B, t_{frac}, a, c_{\max}$)

```
1  c̃ ← (t^a/(t_frac · t_max)^a) · B;
2  if t ≤ (t_frac · t_max) then
3      ℓ = ⌊c̃⌋ − std;
4      h = ⌈c̃⌉ + std;
5  else
6      ℓ = B − std;
7      h = B;
8  for x ∈ P do
9      if (c(x) < ℓ) ∧ (c(x) ≠ c_max) ∧ (c_max ≠ −1) ∧ (|P| > 1) then
10         P ← P \ {x};
11 P̂ = {x ∈ P | ℓ ≤ c(x) ≤ h};
12 if P̂ = ∅ then
13     P̂ ← P;
14 Choose x ∈ P̂ uniformly at random;
15 Return x;
```

constraint values $[\ell, h]$. In case that no solution of those values exists, a uniform choice from the population remaining after removal of individuals of too low constraint values is made. Hence, even if there are no individuals with constraint values in the interval $[\ell, h]$, then lines 8–10 favor increasing constraint values.

The SW-GSEMO3D starts out with a solution $x \in \{0,1\}^n$ chosen by the user, e. g., as the all-zeros string or uniformly at random. It works in two phases. As long as the minimum $\mu$-value of the population called $\mu_{\min}$ is positive, it chooses a solution of this smallest $\mu$-value, applies mutation, usually standard bit mutation avoiding duplicates (Algorithm 4), and accepts the offspring into the population if it is not strictly dominated by another member of the population. All individuals that are weakly dominated by the offspring are then removed from the population. In any case, the current population always consists of mutually non-

Algorithm 7: Fast Sliding-Window GSEMO3D (Fast SW-GSEMO3D) (Parameters: $t_{max}, t_{frac}, std, a, \epsilon$)

```
1  Choose initial solution x ∈ {0,1}^n;
2  t_0 ← −1, t ← 1, μ_min ← μ(x), c_max ← −1;
3  P ← {x};
4  Compute f(x) = (μ(x), v(x), c(x));
5  if (c(x) > c_max) ∧ (c(x) ≤ B) then
6      c_max ← c(x);
7  if μ_min = 0 then
8      t_0 ← t;
9  repeat
10     t ← t + 1;
11     if (t_0 = −1) ∧ (t ≤ t_frac · t_max) then
12         x ← arg min{μ(z) | z ∈ P}
13     else
14         if (t > t_frac · t_max) ∧ (c_max < B − ε) then
15             x ← arg max{c(z) | z ∈ P}
16         else
17             x = sliding-selection3D(P, t − t_0, t_max − t_0, std, B, t_frac, a, c_max);
18     Create y from x by mutation;
19     Compute f(y) = (μ(y), v(y), c(y));
20     if μ(y) < μ_min then
21         μ_min ← μ(y);
22     if (t_0 = −1) ∧ (μ_min = 0) then
23         t_0 ← t;
24     if (c(y) > c_max) ∧ (c(y) ≤ B) then
25         c_max ← c(y);
26     if ∄ w ∈ P : w ≻ y then
27         P ← (P \ {z ∈ P | y ⪰ z}) ∪ {y};
28 until t ≥ t_max;
```

dominating solutions only. From the point of time $t_0$ on where a solution $x$ satisfying $\mu(x) = 0$ is found for the first time, the algorithm chooses from the population using sliding window selection (see Algorithm 6) for the remaining $t_{max} - t_0$ steps. In Algorithm 6, the choice $c_{\max} = -1$ implies that lines 8–10 do nothing. Algorithm 7 called Fast SW-GSEMO3D extends Algorithm 5 with heuristic elements as follows. First of all, sliding window selection is called with user-specified choices of std, $t_{frac}$ and $a$ as defined above. Moreover, it keeps track of the maximum constraint value $c_{\max}$ found in the population (lines 24–25), uses that in the sliding window selection and introduces a margin parameter $\epsilon$ that enables the algorithm to obtain solutions at the constraint bound.

Here, the parameter $\epsilon$ defines an interval at the boundary where we aim to produce solutions. If the algorithm has not obtained enough process towards the constraint boundary as indicated by a value of $c_{\max}$ less than $B - \epsilon$ , then the last $t_{frac} \cdot t_{max}$ steps are used to produce such an individual by selecting the individual $z$ with maximal $c(z)$-value to produce an offspring.

Afterwards, i. e., when the algorithm is close to the constraint boundary, making further progress in the constraint value may be too difficult for sliding window selection. Therefore, for the last $(1 - t_{frac})t_{max}$ steps, the algorithm chooses an individual of maximum constraint value if $c_{\max} < B - \epsilon$ holds.

These heuristic elements underlying the parameters std, $t_{frac}$, $a$ and $\epsilon$ and the use of $c_{\max}$ in the sliding window selection will show some empirical benefits in Section 4.2.

The classical GSEMO algorithm (see Algorithm 1) that has inspired the developments of Algorithms 5 and 7 serves as a baseline in our experiments. Depending on the number of objectives used in the experiments in Section 4.2, we will consider specific instances of the algorithm called GSEMO2D and GSEMO3D which denotes GSEMO with the respective number of objectives.

### 4.1 Runtime Analysis of 3D Sliding Window Algorithm

Based on the ideas for the 3-objective GSEMO from Neumann and Witt (2023a), we formulate the following result for SW-GSEMO3D (Algorithm 5). The analysis is additionally inspired by our previous analysis for the bi-objective sliding window approach. Our theorem assumes an initialization with the all-zeros string. If

uniform initialization is used, SW-GSEMO3D nevertheless reaches the all-zeros string efficiently, as shown in a subsequent theorem (Theorem 5).

The following theorem is based on the maximum population size $P^{(i)}_{\max}$ observed in any of the sliding window intervals. Note that when running the algorithm, the runtime for a given sliding window can be adapted to $t^{(i)}_{\max} = P^{(i)}_{\max} n \ln n$ during the run based on the observed value of $P^{(i)}_{\max}$ in order to guarantee the stated approximation result. Note that the previous result from Neumann and Witt (2023a) showed an upper bound of $O(n^2 P_{\max})$, where $P_{\max}$ is the overall maximum population size observed in the run of the algorithm. If the largest $P^{(i)}_{\max}$ is significantly smaller than $P_{\max}$, the following theorem gives a significantly stronger upper bound.

Theorem 4. Let $P^{(i)}_{\max}$ denote the largest number of individuals with constraint value $i$ present in the population at all points in time where SW-GSEMO3D can select such individuals, let $t^{(i)}_{\max} = P^{(i)}_{\max} n \ln n$ and let $t_{max} = 4en \max_{i=0}^{n-1} t^{(i)}_{\max}$. Then SW-GSEMO3D, initialized with $0^n$, computes a population which includes an optimal solution for the stochastic problem given in Equation (1) (for any choice of $k \in \{0,\dots,n\}$ and $\alpha \in [1/2, 1[$) and Equation (3) (with $c(x) = |x|_1$ for any choice of $B \in \{0,\dots,n\}$ and $\alpha \in [1/2, 1[$) until time $t_{max} = O(\max_{i=0}^{n-1}\{P^{(i)}_{\max}\} \cdot n^2 \log n)$ with probability $1 - o(1)$.

Proof. Let $X^k = \{x \in \{0,1\}^n \mid |x|_1 = k\}$ be the set of all solutions having exactly $k$ elements. We show the following more technical statement $S$: the population $P$ at time $t_{max}$ will, with the probability bound claimed in the theorem, contain for each $\alpha \in [1/2, 1[$ and $k \in \{0,\dots,n\}$ a solution

$$x^k_\alpha = \arg\min_{x \in X^k} \left\{ \mu(x) + K_\alpha \sqrt{v(x)} \right\}, \tag{5}$$

i. e., $P \supseteq \{x^k_\alpha \mid 0 \le k \le n, \alpha \in [1/2, 1[\}$. By Theorem 4.3 in Neumann and Witt (2023a), such a population contains the optimal solutions for any choice of $\alpha \in [1/2, 1[$. Note that not the whole set of Pareto optimal solutions is necessarily required.

To show statement $S$, we re-use the following definitions from Neumann and Witt (2025). Let $\lambda_{i,j} = \frac{\sigma_j^2 - \sigma_i^2}{(\mu_i - \mu_j) + (\sigma_j^2 - \sigma_i^2)}$ for the pair of elements $e_i$ and $e_j$ of the given input where $\sigma_i^2 < \sigma_j^2$ and $\mu_i > \mu_j$ holds, $1 \le i < j \le n$. The set $\Lambda = \{\lambda_0, \lambda_1, \dots, \lambda_\ell, \lambda_{\ell+1}\}$ where $\lambda_1, \dots, \lambda_\ell$ are the values $\lambda_{i,j}$ in increasing order and $\lambda_0 = 0$ and $\lambda_{\ell+1} = 1$. Moreover, we define the function $f_\lambda(x) = \lambda\mu(x) + (1-\lambda)v(x)$ and also use it applied to elements $e_i$, i. e. $f_\lambda(e_i) = \lambda\mu_i + (1-\lambda)\sigma_i^2$.

As noted in Neumann and Witt (2025), for a given $\lambda$ and a given number $k$ of elements to include, the function $f_\lambda$ can be optimized by a greedy approach which iteratively selects a set of $k$ smallest elements according to $f_\lambda(e_i)$. For any $\lambda \in [0,1]$, an optimal solution for $f_\lambda$ with $k$ elements is Pareto optimal as there is no other solution with at least $k$ elements that improves the expected cost or variance without impairing the other. Hence, once obtained such a solution $x$, the resulting objective vector $f_{3D}(x)$ will remain in the population for the rest of the run of SW-GSEMO3D. Furthermore, the set of optimal solutions for different $\lambda$ values only change at the $\lambda$ values of the set $\Lambda$ as these $\lambda$ values constitute the weighting where the order of items according to $f_\lambda$ can switch. This is a direct consequence of the definition of the $\lambda_{i,j}$ above and has already been used in the same way in Ishii et al. (1981); Neumann and Witt (2025).

We consider a $\lambda_i \in \Lambda$ with $0 \le i \le \ell$. Similarly to Ishii et al. (1981), we define $\lambda_i^* = (\lambda_i + \lambda_{i+1})/2$. The order of items according to the weighting of expected value and variance can only change at values $\lambda_i \in \Lambda$ and the resulting objective vectors are not necessarily unique for values $\lambda_i \in \Lambda$. Choosing the $\lambda_i^*$-values in the defined way gives optimal solutions for all $\lambda \in [\lambda_i, \lambda_{i+1}]$ which means that we consider all orders of the items that can lead to optimal solutions when inserting the items greedily according to any fixed weighting of expected weights and variances.

In the following, we analyze the time until an optimal solution with exactly $k$ elements has been produced for $f_{\lambda_i^*}(x) = \lambda_i^*\mu(x) + (1-\lambda_i^*)v(x)$ for any $k \in \{0,\dots,n\}$ and any $i \in \{0,\dots,\ell\}$. Note that these $\lambda_i^*$ values allow to obtain all optimal solutions for the set of functions $f_\lambda$, $\lambda \in [0,1]$.

For a given $i$, let the items be ordered such that $f_{\lambda_i^*}(e_1) \le \dots \le f_{\lambda_i^*}(e_k) \le \dots \le f_{\lambda_i^*}(e_n)$ holds. An optimal solution for $k$ elements and $\lambda_i^*$ consists of $k$ elements with the smallest $f_{\lambda_i^*}(e_i)$-value. If there are more than one element with the value $f_{\lambda_i^*}(e_k)$ then reordering these elements does not change the objective vector or $f_{\lambda_i^*}$-value.

Note that for $k = 0$ the search point $0^n$ is optimal for any $\lambda \in [0,1]$. Picking an optimal solution with $k$ elements for $f_{\lambda_i^*}$ and inserting an element with value $f_{\lambda_i^*}(e_{k+1})$ leads to an optimal solution for $f_{\lambda_i^*}$ with $k+1$ elements. We call such a step, picking the solution that is optimal for $f_{\lambda_i^*}$ with $k$ elements and inserting an element with value $f_{\lambda_i^*}(e_{k+1})$, a success. Assuming such a solution is picked, the probability of inserting the

element is at least $(1/n)(1-1/n)^{n-1} \geq 1/(en)$ since it suffices that SW-GSEMO3D flips a specific bit and does not flip the rest.

We now consider a sequence of events leading to the successes for all values of $k \in \{0,\dots,n-1\}$ and all $i \in \{0,\dots,\ell-1\}$. We abbreviate $P^*_{\max} \coloneqq \max_{j=0}^{n-1} P^{(j)}_{\max}$. By the assumption from the theorem, $0^n$, an optimal solution for $k=0$, is in the population at time 0. Assume that optimal solutions with $k$ elements all for $f_{\lambda^*_i}$, where $i \in \{0,\dots,\ell\}$, are in the population $P$ at time $\tau_k \coloneqq kt_{max}/n = 4eP^*_{\max}kn\ln n$.

Then, by definition of the set $\hat{P}$ of GSEMO3D, for any fixed $i$, such a solution is available for selection up to time

$$\tau_{k+1} - 1 = (k+1)t_{max}/n - 1 = 4eP^*_{\max}(k+1)n\ln n - 1$$

since $\lfloor((k+1)t_{max}/n - 1)/t_{max}) \cdot n\rfloor = k$. The size of the subset population that the algorithm selects from during this period has been denoted by $P^{(k)}_{\max}$. Therefore, the probability of a success at any fixed value $k$ and $i$ is at least $1/(P^{(k)}_{\max}en)$ from time $\tau_k$ until time $\tau_{k+1}-1$, i. e., for a period of $4eP^*_{\max}n\ln n \geq 4eP^{(k)}_{\max}n\ln n$ steps, and the probability of this not happening is at most

$$\left(1 - \frac{1}{P^{(k)}_{\max}en}\right)^{4eP^{(k)}_{\max}n\ln n} \leq \frac{1}{n^4}.$$

The number $\ell$ of different values of $\lambda^*_i$ is at most the number of pairs of elements and therefore at most $n^2$. By a union bound over this number of values and all $k$, the probability to have not obtained all optimal solutions for all $f_{\lambda^*_i}$, where $i \in \{0,\dots,\ell\}$, and all values of $k \in \{0,\dots,B\}$ by time $t_{max}$ is $O(1/n)$. This shows the result for Equation (1).

We obtain the result for Equation (3) by executing the arguments used in the proof of (Neumann and Witt, 2023a, Theorem 4.3). Consider any fixed $B \in \{0,\dots,n\}$. Let $x^*_\alpha$, $\alpha \in [1/2,1[$, be an optimal solution for the problems given in Equation (3). According to Equation (5), the solution $x^j_\alpha$ in $P$ is the solution with exactly $j$ elements that minimizes $\mu(x) + K_\alpha\sqrt{v(x)}$. Hence, the solution $x^j_\alpha$ with the maximal value of $j$ for which $\mu(x^j_\alpha) + K_\alpha\sqrt{v(x^j_\alpha)} \leq B$ holds satisfies $|x^j_\alpha| = |x^*_\alpha|$ as otherwise $x^j_\alpha$ would not be a solution with the maximal number of elements for which the constraint holds or $x^*_\alpha$ would not be an optimal solution for the given value of $\alpha$. Therefore, $P$ includes with probability $1-o(1)$ after $t_{max}$ steps an optimal solution for Equation (3) and any $B \in \{0,\dots,n\}$ and any $\alpha \in [1/2,1[$, which completes the proof.

□

Finally, as mentioned above, we consider a uniform choice of the initial individual of SW-GSEMO3D and show that the time to reach the all-zeros string is bounded by $O(n\log n)$ if the largest possible expected value $\mu_{\max} \coloneqq \sum_{i=1}^n \mu_i$ of an individual is polynomially bounded. Hence, this constitutes a lower-order term in terms of the optimization time bound proved in Theorem 4 above. Even if $\mu_{\max}$ is exponential like $2^{n^c}$ for a constant $c$, the bound of the following theorem is still polynomial.

Theorem 5. Consider SW-GSEMO3D initialized with a random bit string. Then the expected time until its population includes the all-zeros string for the first time is bounded from above by $O(n(\log\mu_{\max}+1))$.

Proof. We apply multiplicative drift analysis (Doerr et al., 2012) with respect to the stochastic process $X_t \coloneqq \min\{\mu(x) \mid x \in P_t\}$, i. e., the minimum expected value of the individuals of the population at time $t$. By definition, before the all-zeros string is included in the population, SW-GSEMO3D chooses only individuals of minimum $\mu$-value for mutation. Note that there might be more than one such individual. The current $\mu$-value of an individual is the sum of the expected values belonging to the bit positions that are set to 1. Standard-bit mutation flips each of these positions to 0 without flipping any other bit with probability at least $(1/n)(1-1/n)^{n-1} \geq 1/(en)$. Such steps decrease the $\mu$-value of the solution, which is therefore not dominated by any other solution in the population and will be included afterwards. Hence, we obtain the drift $E(X_t - X_{t+1} \mid X_t) \geq X_t/(en)$. We can now set the parameters of the multiplicative drift theorem.. First, from the analysis above we have $\delta = 1/(en)$. Moreover, clearly $X_0 \leq \mu_{\max}$. Since by assumption $\mu_i \geq 1$ for all $i \in \{1,\dots,n\}$, the smallest possible non-zero value of $X_t$ is at least 1. Hence, the multiplicative drift theorem gives an expected time of at most $\frac{\ln(\mu_{\max})+1}{\delta} = O(n(\log\mu_{\max}+1))$ to reach state 0 in the $X_t$-process, i. e., an individual with all zeros. □

It should be noted that our approaches using sliding-window selection are not able to go back and correct important steps that they might have missed. This can be observed from the statements in Theorem 3 and 4 that hold with probability $1-o(1)$. There is a failure of achieving the desired result which can happen if important steps of inserting the "right" element are missed.

### 4.2 Experiments

We carry out experimental investigations for the new sliding window approach on the chance constrained dominating set problem and show where the new approach performs significantly better than the ones introduced in Neumann and Witt (2023a, 2025).

Table 3: Results for stochastic minimum weight dominating set with uniform weights and different confidence levels of $\alpha$ where $\alpha = 1-\beta$. Results after 10M fitness evaluations. $p_1$: Test GSEMO2D vs GSEMO3D, $p_2$: Test GSEMO2D vs Fast SW-GSEMO3D, $p_3$: Test GSEMO3D vs Fast SW-GSEMO3D, $p_4$: Fast GSEMO3D vs Fast SW-GSEMO3D$_0$. Penalty function value for run not obtaining a feasible solution is $10^{10}$ (applied to GSEMO3D for graphs ca-GrQc and Erdos992 )

| Graph | $\beta$ | GSEMO2D | | GSEMO3D | | $p_1$ | Fast SW-GSEMO3D | | $p_2$ | $p_3$ | Fast SW-GSEMO3D$_0$ | | $p_4$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | Std | Mean | Std | | Mean | Std | | | Mean | Std | |
| cfat200-1 | 0.2 | 3615 | 91 | 3599 | 79 | 0.544 | 3594 | 75 | 0.420 | 0.807 | 3598 | 74 | 0.767 |
| | 0.1 | 3989 | 96 | 3972 | 80 | 0.544 | 3967 | 77 | 0.391 | 0.734 | 3971 | 79 | 0.745 |
| | 0.01 | 4866 | 109 | 4845 | 86 | 0.535 | 4842 | 87 | 0.383 | 0.836 | 4846 | 90 | 0.784 |
| | 1.0E-4 | 6015 | 126 | 5991 | 98 | 0.455 | 5989 | 101 | 0.412 | 0.894 | 5989 | 100 | 0.888 |
| | 1.0E-6 | 6855 | 138 | 6832 | 108 | 0.605 | 6827 | 108 | 0.420 | 0.712 | 6825 | 107 | 0.848 |
| | 1.0E-8 | 7546 | 147 | 7525 | 118 | 0.641 | 7517 | 115 | 0.455 | 0.723 | 7514 | 114 | 0.935 |
| | 1.0E-10 | 8145 | 154 | 8125 | 125 | 0.751 | 8115 | 120 | 0.525 | 0.717 | 8112 | 120 | 0.853 |
| | 1.0E-12 | 8680 | 159 | 8659 | 130 | 0.859 | 8651 | 126 | 0.615 | 0.790 | 8646 | 124 | 0.802 |
| | 1.0E-14 | 9169 | 164 | 9148 | 133 | 0.842 | 9139 | 130 | 0.600 | 0.728 | 9133 | 128 | 0.890 |
| cfat200-2 | 0.2 | 1791 | 49 | 1767 | 32 | 0.049 | 1766 | 33 | 0.031 | 0.712 | 1765 | 33 | 0.971 |
| | 0.1 | 2040 | 54 | 2016 | 37 | 0.074 | 2014 | 36 | 0.044 | 0.819 | 2013 | 37 | 0.824 |
| | 0.01 | 2621 | 72 | 2593 | 51 | 0.162 | 2588 | 49 | 0.066 | 0.610 | 2587 | 50 | 0.912 |
| | 1.0E-4 | 3381 | 97 | 3336 | 65 | 0.070 | 3334 | 66 | 0.061 | 0.836 | 3334 | 67 | 0.947 |
| | 1.0E-6 | 3937 | 113 | 3880 | 71 | 0.044 | 3879 | 75 | 0.036 | 0.853 | 3879 | 76 | 0.994 |
| | 1.0E-8 | 4394 | 124 | 4329 | 77 | 0.032 | 4328 | 79 | 0.027 | 0.853 | 4328 | 79 | 1.000 |
| | 1.0E-10 | 4793 | 132 | 4720 | 82 | 0.028 | 4719 | 82 | 0.021 | 0.877 | 4718 | 82 | 1.000 |
| | 1.0E-12 | 5149 | 139 | 5071 | 85 | 0.024 | 5069 | 85 | 0.020 | 0.888 | 5068 | 85 | 0.988 |
| | 1.0E-14 | 5475 | 145 | 5391 | 88 | 0.020 | 5389 | 87 | 0.016 | 0.912 | 5389 | 87 | 0.994 |
| ca-netscience | 0.2 | 33042 | 1289 | 33007 | 1023 | 0.712 | 32398 | 814 | 0.038 | 0.018 | 32399 | 861 | 0.976 |
| | 0.1 | 34568 | 1302 | 34514 | 1028 | 0.745 | 33907 | 815 | 0.031 | 0.019 | 33908 | 865 | 0.988 |
| | 0.01 | 38189 | 1334 | 38089 | 1040 | 0.848 | 37486 | 821 | 0.019 | 0.019 | 37489 | 874 | 0.941 |
| | 1.0E-4 | 43012 | 1380 | 42846 | 1054 | 1.000 | 42248 | 841 | 0.011 | 0.020 | 42252 | 881 | 0.988 |
| | 1.0E-6 | 46591 | 1415 | 46377 | 1065 | 0.824 | 45763 | 858 | 0.009 | 0.023 | 45786 | 888 | 0.882 |
| | 1.0E-8 | 49557 | 1442 | 49303 | 1076 | 0.712 | 48712 | 870 | 0.008 | 0.021 | 48717 | 896 | 0.894 |
| | 1.0E-10 | 52145 | 1465 | 51857 | 1087 | 0.615 | 51266 | 883 | 0.009 | 0.028 | 51275 | 906 | 0.935 |
| | 1.0E-12 | 54467 | 1487 | 54150 | 1096 | 0.564 | 53557 | 894 | 0.007 | 0.028 | 53570 | 914 | 0.923 |
| | 1.0E-14 | 56592 | 1507 | 56249 | 1105 | 0.487 | 55653 | 905 | 0.006 | 0.029 | 55670 | 923 | 0.912 |
| ca-GrQc | 0.2 | 5646101 | 79194 | 9666938258 | 1824254292 | 0.000 | 4920986 | 45094 | 0.000 | 0.000 | 4924856 | 40968 | 0.756 |
| | 0.1 | 5712770 | 79494 | 9666940921 | 1824239705 | 0.000 | 4983403 | 45308 | 0.000 | 0.000 | 4987255 | 41159 | 0.756 |
| | 0.01 | 5871104 | 80213 | 9666947246 | 1824205061 | 0.000 | 5131640 | 45823 | 0.000 | 0.000 | 5135447 | 41621 | 0.790 |
| | 1.0E-4 | 6082155 | 81182 | 9666955677 | 1824158882 | 0.000 | 5329219 | 46511 | 0.000 | 0.000 | 5332980 | 42256 | 0.767 |
| | 1.0E-6 | 6238913 | 81909 | 9666961940 | 1824124582 | 0.000 | 5475970 | 47028 | 0.000 | 0.000 | 5479688 | 42740 | 0.779 |
| | 1.0E-8 | 6369023 | 82517 | 9666967137 | 1824096113 | 0.000 | 5597768 | 47451 | 0.000 | 0.000 | 5601451 | 43151 | 0.802 |
| | 1.0E-10 | 6482579 | 83051 | 9666971674 | 1824071266 | 0.000 | 5704069 | 47822 | 0.000 | 0.000 | 5707719 | 43516 | 0.779 |
| | 1.0E-12 | 6584589 | 83534 | 9666975749 | 1824048945 | 0.000 | 5799961 | 48160 | 0.000 | 0.000 | 5803181 | 43848 | 0.767 |
| | 1.0E-14 | 6677976 | 83978 | 9666979480 | 1824028511 | 0.000 | 5886980 | 48471 | 0.000 | 0.000 | 5890573 | 44156 | 0.767 |
| Erdos992 | 0.2 | 13716872 | 82588 | 10000000000 | 0 | 0.000 | 13382678 | 62860 | 0.000 | 0.000 | 13477560 | 55830 | 0.848 |
| | 0.1 | 13842990 | 82789 | 10000000000 | 0 | 0.000 | 13607667 | 62812 | 0.000 | 0.000 | 13602550 | 55731 | 0.848 |
| | 0.01 | 14142509 | 83278 | 10000000000 | 0 | 0.000 | 13904505 | 62706 | 0.000 | 0.000 | 13899386 | 55512 | 0.813 |
| | 1.0E-4 | 14541754 | 83954 | 10000000000 | 0 | 0.000 | 14300178 | 62586 | 0.000 | 0.000 | 14295055 | 55242 | 0.790 |
| | 1.0E-6 | 14838295 | 84474 | 10000000000 | 0 | 0.000 | 14594065 | 62512 | 0.000 | 0.000 | 14588938 | 55059 | 0.836 |
| | 1.0E-8 | 15084429 | 84917 | 10000000000 | 0 | 0.000 | 14837996 | 62461 | 0.000 | 0.000 | 14832866 | 54917 | 0.836 |
| | 1.0E-10 | 15299247 | 85313 | 10000000000 | 0 | 0.000 | 15050890 | 62423 | 0.000 | 0.000 | 15045759 | 54801 | 0.824 |
| | 1.0E-12 | 15492221 | 85674 | 10000000000 | 0 | 0.000 | 15242138 | 62395 | 0.000 | 0.000 | 15237005 | 54703 | 0.836 |
| | 1.0E-14 | 15668883 | 86011 | 10000000000 | 0 | 0.000 | 15417219 | 62375 | 0.000 | 0.000 | 15412085 | 54619 | 0.871 |

Table 4: Results for stochastic minimum weight dominating set with degree-based weights and different confidence levels of $\alpha$ where $\alpha = 1 - \beta$. Results after 10M fitness evaluations. $p_1$: Test GSEMO2D vs GSEMO3D, $p_2$: Test GSEMO2D vs Fast SW-GSEMO3D, $p_3$: Test GSEMO3D vs Fast SW-GSEMO3D, $p_4$: Fast GSEMO3D vs Fast SW-GSEMO3D$_0$. Penalty function value for run not obtaining a feasible solution is $10^{10}$ (applied to GSEMO3D for graphs ca-GrQc and Erdos992 )

| Graph | $\beta$ | GSEMO2D | | GSEMO3D | | $p_1$ | Fast SW-GSEMO3D | | $p_2$ | $p_3$ | Fast SW-GSEMO3D$_0$ | | $p_4$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | Std | Mean | Std | | Mean | Std | | | Mean | Std | |
| cfat200-1 | 0.2 | 4444 | 115 | 4387 | 6 | 0.001 | 4407 | 75 | 0.011 | 0.535 | 4398 | 55 | 0.779 |
| | 0.1 | 4781 | 119 | 4721 | 9 | 0.003 | 4742 | 77 | 0.023 | 0.446 | 4733 | 56 | 0.790 |
| | 0.01 | 5582 | 129 | 5512 | 16 | 0.004 | 5535 | 83 | 0.036 | 0.348 | 5526 | 61 | 0.819 |
| | 1.0E-4 | 6650 | 143 | 6566 | 26 | 0.003 | 6592 | 91 | 0.035 | 0.287 | 6584 | 68 | 0.830 |
| | 1.0E-6 | 7443 | 154 | 7349 | 34 | 0.003 | 7378 | 98 | 0.037 | 0.268 | 7369 | 74 | 0.830 |
| | 1.0E-8 | 8101 | 163 | 7999 | 40 | 0.003 | 8029 | 103 | 0.041 | 0.268 | 8021 | 79 | 0.830 |
| | 1.0E-10 | 8675 | 171 | 8567 | 45 | 0.003 | 8598 | 108 | 0.044 | 0.261 | 8590 | 84 | 0.865 |
| | 1.0E-12 | 9191 | 178 | 9076 | 50 | 0.003 | 9109 | 113 | 0.043 | 0.261 | 9101 | 88 | 0.865 |
| | 1.0E-14 | 9663 | 185 | 9542 | 55 | 0.003 | 9577 | 118 | 0.041 | 0.268 | 9569 | 92 | 0.853 |
| cfat200-2 | 0.2 | 3041 | 172 | 2963 | 4 | 0.027 | 2963 | 4 | 0.027 | 0.929 | 2963 | 4 | 0.830 |
| | 0.1 | 3267 | 178 | 3185 | 6 | 0.027 | 3185 | 6 | 0.027 | 0.929 | 3185 | 6 | 0.830 |
| | 0.01 | 3803 | 194 | 3713 | 11 | 0.027 | 3713 | 10 | 0.027 | 0.929 | 3712 | 10 | 0.830 |
| | 1.0E-4 | 4518 | 216 | 4416 | 17 | 0.027 | 4415 | 16 | 0.027 | 0.929 | 4415 | 17 | 0.830 |
| | 1.0E-6 | 5049 | 232 | 4938 | 22 | 0.027 | 4937 | 21 | 0.027 | 0.929 | 4937 | 21 | 0.830 |
| | 1.0E-8 | 5490 | 245 | 5371 | 26 | 0.027 | 5371 | 24 | 0.027 | 0.929 | 5370 | 25 | 0.830 |
| | 1.0E-10 | 5875 | 257 | 5749 | 30 | 0.027 | 5749 | 28 | 0.027 | 0.929 | 5748 | 28 | 0.830 |
| | 1.0E-12 | 6220 | 267 | 6089 | 33 | 0.027 | 6088 | 30 | 0.027 | 0.929 | 6087 | 31 | 0.830 |
| | 1.0E-14 | 6537 | 277 | 6400 | 36 | 0.027 | 6399 | 33 | 0.027 | 0.929 | 6398 | 34 | 0.830 |
| ca-netscience | 0.2 | 28164 | 1002 | 26169 | 196 | 0.000 | 26097 | 197 | 0.000 | 0.017 | 26098 | 193 | 0.900 |
| | 0.1 | 29689 | 1029 | 27657 | 200 | 0.000 | 27580 | 207 | 0.000 | 0.038 | 27583 | 201 | 0.853 |
| | 0.01 | 33300 | 1098 | 31183 | 216 | 0.000 | 31092 | 238 | 0.000 | 0.114 | 31098 | 224 | 0.848 |
| | 1.0E-4 | 38103 | 1192 | 35874 | 251 | 0.000 | 35758 | 284 | 0.000 | 0.092 | 35767 | 266 | 0.813 |
| | 1.0E-6 | 41665 | 1265 | 39355 | 285 | 0.000 | 39220 | 324 | 0.000 | 0.076 | 39230 | 303 | 0.813 |
| | 1.0E-8 | 44620 | 1327 | 42243 | 317 | 0.000 | 42091 | 359 | 0.000 | 0.067 | 42103 | 336 | 0.813 |
| | 1.0E-10 | 47198 | 1381 | 44763 | 347 | 0.000 | 44596 | 390 | 0.000 | 0.067 | 44610 | 366 | 0.784 |
| | 1.0E-12 | 49514 | 1429 | 47026 | 374 | 0.000 | 46845 | 418 | 0.000 | 0.074 | 46861 | 394 | 0.842 |
| | 1.0E-14 | 51633 | 1474 | 49098 | 400 | 0.000 | 48905 | 444 | 0.000 | 0.081 | 48921 | 419 | 0.830 |
| ca-GrQc | 0.2 | 4032668 | 60538 | 9666845352 | 1824763160 | 0.000 | 3455870 | 17041 | 0.000 | 0.000 | 3457971 | 16176 | 0.460 |
| | 0.1 | 4100297 | 61062 | 9666847956 | 1824748898 | 0.000 | 3517608 | 17204 | 0.000 | 0.000 | 3519680 | 16336 | 0.442 |
| | 0.01 | 4260901 | 62312 | 9666854140 | 1824715027 | 0.000 | 3664186 | 17591 | 0.000 | 0.000 | 3666208 | 16722 | 0.442 |
| | 1.0E-4 | 4474975 | 63984 | 9666862383 | 1824669878 | 0.000 | 3859529 | 18140 | 0.000 | 0.000 | 3861506 | 17249 | 0.408 |
| | 1.0E-6 | 4633978 | 65230 | 9666868505 | 1824636344 | 0.000 | 4004604 | 18568 | 0.000 | 0.000 | 4006550 | 17648 | 0.460 |
| | 1.0E-8 | 4765953 | 66266 | 9666873587 | 1824608510 | 0.000 | 4125007 | 18930 | 0.000 | 0.000 | 4126933 | 17988 | 0.469 |
| | 1.0E-10 | 4881136 | 67173 | 9666878022 | 1824584217 | 0.000 | 4230080 | 19246 | 0.000 | 0.000 | 4232003 | 18291 | 0.460 |
| | 1.0E-12 | 4984607 | 67988 | 9666882006 | 1824562394 | 0.000 | 4324470 | 19538 | 0.000 | 0.000 | 4326386 | 18569 | 0.451 |
| | 1.0E-14 | 5079332 | 68736 | 9666885654 | 1824542416 | 0.000 | 4410880 | 19810 | 0.000 | 0.000 | 4412792 | 18828 | 0.478 |
| Erdos992 | 0.2 | 9307396 | 60880 | 10000000000 | 0 | 0.000 | 9104433 | 4932 | 0.000 | 0.000 | 9104421 | 4931 | 0.965 |
| | 0.1 | 9433699 | 61228 | 10000000000 | 0 | 0.000 | 9229249 | 5100 | 0.000 | 0.000 | 9229244 | 4958 | 0.906 |
| | 0.01 | 9733657 | 62061 | 10000000000 | 0 | 0.000 | 9525667 | 5566 | 0.000 | 0.000 | 9525686 | 5110 | 0.988 |
| | 1.0E-4 | 10133488 | 63184 | 10000000000 | 0 | 0.000 | 9920775 | 6299 | 0.000 | 0.000 | 9920827 | 5490 | 0.953 |
| | 1.0E-6 | 10430463 | 64027 | 10000000000 | 0 | 0.000 | 10214242 | 6902 | 0.000 | 0.000 | 10214318 | 5882 | 0.941 |
| | 1.0E-8 | 10676958 | 64732 | 10000000000 | 0 | 0.000 | 10457822 | 7430 | 0.000 | 0.000 | 10457921 | 6265 | 0.988 |
| | 1.0E-10 | 10892090 | 65351 | 10000000000 | 0 | 0.000 | 10670412 | 7907 | 0.000 | 0.000 | 10670530 | 6635 | 0.976 |
| | 1.0E-12 | 11085348 | 65911 | 10000000000 | 0 | 0.000 | 10861385 | 8347 | 0.000 | 0.000 | 10861521 | 6990 | 0.976 |
| | 1.0E-14 | 11262269 | 66425 | 10000000000 | 0 | 0.000 | 11036215 | 8757 | 0.000 | 0.000 | 11036367 | 7332 | 1.000 |

Table 5: Results for stochastic minimum weight dominating set with uniform weights and different confidence levels of $\alpha$ where $\alpha = 1 - \beta$. Results after 1M fitness evaluations. $p_1$: Test (1+1) EA vs GSEMO2D, $p_2$: Test (1+1) EA vs Fast SW-GSEMO3D, $p_3$: Test GSEMO2D vs Fast SW-GSEMO3D, $p_4$: Test (1+1) EA vs Fast SW-GSEMO3D$_0$, $p_5$: Test GSEMO2D vs Fast SW-GSEMO3D$_0$, $p_6$: Test Fast GSEMO3D vs Fast SW-GSEMO3D$_0$.

| Graph | $\beta$ | (1+1) EA | | GSEMO2D | | | Fast SW-GSEMO3D | | | Fast SW-GSEMO3D$_0$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | Std | Mean | Std | $p_1$ | Mean | Std | $p_2$ | $p_3$ | Mean | Std | $p_4$ | $p_5$ | $p_6$ |
| ca-CSphd | 0.2 | 1176951 | 29560 | 1149185 | 21187 | 0.000 | 1053428 | 5919 | 0.000 | 0.000 | 1052480 | 4910 | 0.000 | 0.000 | 0.367 |
| | 0.1 | 1200964 | 25599 | 1173498 | 21419 | 0.000 | 1076406 | 5973 | 0.000 | 0.000 | 1075454 | 4965 | 0.000 | 0.000 | 0.367 |
| | 0.01 | 1235668 | 29329 | 1231241 | 21969 | 0.836 | 1130976 | 6108 | 0.000 | 0.000 | 1130017 | 5105 | 0.000 | 0.000 | 0.383 |
| | 1.0E-4 | 1314570 | 28190 | 1308208 | 22705 | 0.451 | 1203715 | 6301 | 0.000 | 0.000 | 1202747 | 5308 | 0.000 | 0.000 | 0.375 |
| | 1.0E-6 | 1378890 | 25618 | 1365376 | 23254 | 0.062 | 1257743 | 6455 | 0.000 | 0.000 | 1256767 | 5471 | 0.000 | 0.000 | 0.391 |
| | 1.0E-8 | 1410240 | 22358 | 1412826 | 23711 | 0.712 | 1302586 | 6587 | 0.000 | 0.000 | 1301605 | 5612 | 0.000 | 0.000 | 0.383 |
| | 1.0E-10 | 1455663 | 21030 | 1454239 | 24110 | 0.894 | 1341724 | 6707 | 0.000 | 0.000 | 1340738 | 5740 | 0.000 | 0.000 | 0.375 |
| | 1.0E-12 | 1495936 | 29008 | 1491441 | 24470 | 0.574 | 1376883 | 6818 | 0.000 | 0.000 | 1375892 | 5859 | 0.000 | 0.000 | 0.433 |
| | 1.0E-14 | 1526403 | 25752 | 1525499 | 24799 | 1.000 | 1409069 | 6921 | 0.000 | 0.000 | 1408074 | 5970 | 0.000 | 0.000 | 0.469 |
| ca-HepPh | 0.2 | 24866045 | 323815 | 24664200 | 251849 | 0.010 | 21903190 | 229592 | 0.000 | 0.000 | 21655867 | 211163 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 25126756 | 223438 | 24941951 | 253168 | 0.006 | 22162387 | 230935 | 0.000 | 0.000 | 21913353 | 212217 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 25709929 | 219138 | 25601440 | 256304 | 0.101 | 22777957 | 234129 | 0.000 | 0.000 | 22524858 | 214726 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 26602650 | 271535 | 26480507 | 260496 | 0.132 | 23598486 | 238398 | 0.000 | 0.000 | 23339968 | 218088 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 27104133 | 304517 | 27133437 | 263618 | 0.595 | 24207935 | 241578 | 0.000 | 0.000 | 23945393 | 220598 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 27675018 | 335011 | 27675380 | 266213 | 0.953 | 24713788 | 244221 | 0.000 | 0.000 | 24447901 | 222696 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 28123068 | 314336 | 28148371 | 268482 | 0.941 | 25155281 | 246532 | 0.000 | 0.000 | 24886470 | 224540 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 28616742 | 357514 | 28573268 | 270522 | 0.636 | 25551883 | 248611 | 0.000 | 0.000 | 25280441 | 226199 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 28831138 | 286317 | 28962248 | 272392 | 0.143 | 25914960 | 250516 | 0.000 | 0.000 | 25641110 | 227723 | 0.000 | 0.000 | 0.000 |
| ca-AstroPh | 0.2 | 51557918 | 568600 | 51043030 | 528254 | 0.001 | 64103184 | 4470490 | 0.000 | 0.000 | 45226809 | 500442 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 51942457 | 555700 | 51548678 | 531285 | 0.021 | 64668884 | 4491484 | 0.000 | 0.000 | 45698407 | 502905 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 53161346 | 552583 | 52749539 | 538490 | 0.017 | 66012371 | 4541343 | 0.000 | 0.000 | 46811411 | 508259 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 54581672 | 577272 | 54350226 | 548110 | 0.160 | 67803180 | 4607807 | 0.000 | 0.000 | 48311327 | 516571 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 55574306 | 568036 | 55539139 | 555269 | 0.965 | 69133305 | 4657175 | 0.000 | 0.000 | 49420191 | 522386 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 56482376 | 659036 | 56525957 | 561218 | 0.525 | 70237331 | 4698155 | 0.000 | 0.000 | 50340566 | 527216 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 56997947 | 442067 | 57387223 | 566415 | 0.003 | 71200892 | 4733922 | 0.000 | 0.000 | 51143842 | 531435 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 58002729 | 535712 | 58160914 | 571088 | 0.255 | 72066476 | 4766054 | 0.000 | 0.000 | 51865440 | 535228 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 58598177 | 480173 | 58869203 | 575369 | 0.033 | 72858892 | 4795471 | 0.000 | 0.000 | 52526040 | 538702 | 0.000 | 0.000 | 0.000 |
| ca-CondMat | 0.2 | 87564936 | 940507 | 86293144 | 783450 | 0.000 | 431800766 | 1807172824 | 0.000 | 0.000 | 75931086 | 610598 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 87993459 | 758163 | 87014750 | 786716 | 0.000 | 432555511 | 1807030509 | 0.000 | 0.000 | 76602241 | 613185 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 89127748 | 754815 | 88728501 | 794478 | 0.069 | 434347964 | 1806692530 | 0.000 | 0.000 | 78196177 | 619334 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 91086972 | 739979 | 91012856 | 804836 | 0.859 | 436737226 | 1806242026 | 0.000 | 0.000 | 80320825 | 627546 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 92467544 | 650611 | 92709566 | 812539 | 0.204 | 438511855 | 1859074220 | 0.000 | 0.000 | 81898913 | 633655 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 93588939 | 815736 | 94117866 | 818937 | 0.015 | 439984829 | 1805629695 | 0.000 | 0.000 | 83208753 | 638731 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 94693345 | 520061 | 95346987 | 824526 | 0.002 | 441270396 | 1805387908 | 0.000 | 0.000 | 84351942 | 643167 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 96086744 | 975803 | 96451130 | 829550 | 0.183 | 442425245 | 1805169569 | 0.000 | 0.000 | 85378890 | 647155 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 96686021 | 889063 | 97461938 | 834151 | 0.001 | 443462473 | 1804970239 | 0.000 | 0.000 | 86319029 | 650809 | 0.000 | 0.000 | 0.000 |

Table 6: Results for stochastic minimum weight dominating set with degree-based weights and different confidence levels of $\alpha$ where $\alpha = 1-\beta$. Results after 1M fitness evaluations. $p_1$: Test (1+1) EA vs GSEMO2D, $p_2$: Test (1+1) EA vs Fast SW-GSEMO3D, $p_3$: Test GSEMO2D vs Fast SW-GSEMO3D, $p_4$: Test (1+1) EA vs Fast SW-GSEMO3D$_0$, $p_5$: Test GSEMO2D vs Fast SW-GSEMO3D$_0$, $p_6$: Test Fast GSEMO3D vs Fast SW-GSEMO3D$_0$.

| Graph | $\beta$ | (1+1) EA | | GSEMO2D | | $p_1$ | Fast SW-GSEMO3D | | $p_2$ | $p_3$ | Fast SW-GSEMO3D$_0$ | | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | Std | Mean | Std | | Mean | Std | | | Mean | Std | | | |
| ca-CSphd | 0.2 | 1176359 | 23453 | 1166190 | 32090 | 0.071 | 1053397 | 6005 | 0.000 | 0.000 | 1052796 | 5364 | 0.000 | 0.000 | 0.668 |
| | 0.1 | 1197763 | 27695 | 1190714 | 32418 | 0.322 | 1076425 | 6076 | 0.000 | 0.000 | 1075804 | 5419 | 0.000 | 0.000 | 0.657 |
| | 0.01 | 1243411 | 22570 | 1248957 | 33200 | 0.416 | 1131114 | 6256 | 0.000 | 0.000 | 1130446 | 5555 | 0.000 | 0.000 | 0.647 |
| | 1.0E-4 | 1318313 | 23041 | 1326592 | 34244 | 0.294 | 1204011 | 6514 | 0.000 | 0.000 | 1203281 | 5748 | 0.000 | 0.000 | 0.615 |
| | 1.0E-6 | 1370672 | 30267 | 1384255 | 35022 | 0.110 | 1258155 | 6718 | 0.000 | 0.000 | 1257380 | 5898 | 0.000 | 0.000 | 0.615 |
| | 1.0E-8 | 1411274 | 28448 | 1432117 | 35668 | 0.004 | 1303096 | 6895 | 0.000 | 0.000 | 1302282 | 6027 | 0.000 | 0.000 | 0.605 |
| | 1.0E-10 | 1465714 | 31864 | 1473889 | 36233 | 0.322 | 1342319 | 7054 | 0.000 | 0.000 | 1341472 | 6143 | 0.000 | 0.000 | 0.584 |
| | 1.0E-12 | 1494845 | 26265 | 1511414 | 36742 | 0.076 | 1377554 | 7202 | 0.000 | 0.000 | 1376676 | 6249 | 0.000 | 0.000 | 0.595 |
| | 1.0E-14 | 1539841 | 28989 | 1545767 | 37207 | 0.756 | 1409811 | 7339 | 0.000 | 0.000 | 1408905 | 6349 | 0.000 | 0.000 | 0.584 |
| ca-HepPh | 0.2 | 24940255 | 229915 | 24770247 | 328453 | 0.019 | 21925184 | 256481 | 0.000 | 0.000 | 21672753 | 170643 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 25129650 | 329488 | 25048454 | 330378 | 0.322 | 22184365 | 258158 | 0.000 | 0.000 | 21930197 | 171670 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 25756684 | 291735 | 25709164 | 334958 | 0.584 | 22799898 | 262146 | 0.000 | 0.000 | 22541605 | 174114 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 26478853 | 313950 | 26589855 | 341060 | 0.156 | 23620377 | 267470 | 0.000 | 0.000 | 23356586 | 177388 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 27073736 | 305766 | 27243988 | 345599 | 0.055 | 24229788 | 271431 | 0.000 | 0.000 | 23961915 | 179829 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 27647166 | 283416 | 27786927 | 349368 | 0.086 | 24735610 | 274722 | 0.000 | 0.000 | 24463448 | 181862 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 28101126 | 327539 | 28260785 | 352656 | 0.079 | 25177076 | 277597 | 0.000 | 0.000 | 24902857 | 183641 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 28523939 | 323332 | 28686461 | 355612 | 0.071 | 25573654 | 280182 | 0.000 | 0.000 | 25296778 | 185243 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 28937484 | 306489 | 29076155 | 358320 | 0.147 | 25936707 | 282550 | 0.000 | 0.000 | 25657400 | 186712 | 0.000 | 0.000 | 0.000 |
| ca-AstroPh | 0.2 | 51524407 | 570578 | 50681144 | 611971 | 0.000 | 64564376 | 6120887 | 0.000 | 0.000 | 45109042 | 578792 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 52060421 | 613178 | 51184838 | 615193 | 0.000 | 65131764 | 6149639 | 0.000 | 0.000 | 45579883 | 581545 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 53271848 | 477115 | 53371067 | 622852 | 0.000 | 66472261 | 6217924 | 0.000 | 0.000 | 46698084 | 588091 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 54408644 | 498038 | 53975385 | 633070 | 0.012 | 68273417 | 6309850 | 0.000 | 0.000 | 48188591 | 596839 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 55533826 | 541001 | 55159915 | 640666 | 0.015 | 69609515 | 6376561 | 0.000 | 0.000 | 49295668 | 603348 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 56254153 | 556558 | 56142930 | 646976 | 0.469 | 70716840 | 6432682 | 0.000 | 0.000 | 50214561 | 608758 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 57221946 | 419431 | 57000876 | 652487 | 0.165 | 71683280 | 6481663 | 0.000 | 0.000 | 51016543 | 613485 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 58115722 | 620857 | 57771583 | 657440 | 0.048 | 72551451 | 6525664 | 0.000 | 0.000 | 51736978 | 617736 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 58619146 | 587556 | 58477140 | 661975 | 0.433 | 73346234 | 6565947 | 0.000 | 0.000 | 52396514 | 621630 | 0.000 | 0.000 | 0.000 |
| ca-CondMat | 0.2 | 8757971 | 869378 | 8654792 | 742313 | 0.000 | 10369413 | 9559675 | 0.000 | 0.000 | 7593910 | 868161 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 8771312 | 806685 | 8727037 | 745665 | 0.028 | 10448253 | 9599957 | 0.000 | 0.000 | 7660990 | 872129 | 0.000 | 0.000 | 0.000 |
| | 0.01 | 8920700 | 639620 | 8898613 | 753635 | 0.132 | 10635491 | 9692249 | 0.000 | 0.000 | 7820298 | 881557 | 0.000 | 0.000 | 0.000 |
| | 1.0E-4 | 9085633 | 857580 | 9127316 | 764272 | 0.110 | 10885507 | 9816605 | 0.000 | 0.000 | 8032649 | 894132 | 0.000 | 0.000 | 0.000 |
| | 1.0E-6 | 9239208 | 865712 | 9297186 | 772182 | 0.017 | 11070448 | 9908973 | 0.000 | 0.000 | 8190373 | 903477 | 0.000 | 0.000 | 0.000 |
| | 1.0E-8 | 9367441 | 808337 | 9438181 | 778752 | 0.003 | 11224314 | 9985641 | 0.000 | 0.000 | 8321286 | 911237 | 0.000 | 0.000 | 0.000 |
| | 1.0E-10 | 9460492 | 600565 | 9561288 | 784491 | 0.000 | 11338604 | 10052555 | 0.000 | 0.000 | 8435542 | 918012 | 0.000 | 0.000 | 0.000 |
| | 1.0E-12 | 9614567 | 762399 | 9671817 | 789649 | 0.017 | 11479238 | 10112666 | 0.000 | 0.000 | 8538183 | 924101 | 0.000 | 0.000 | 0.000 |
| | 1.0E-14 | 9681407 | 636459 | 9772980 | 794374 | 0.000 | 11589676 | 10167696 | 0.000 | 0.000 | 8632147 | 929676 | 0.000 | 0.000 | 0.000 |

Table 7: Average maximum population size and standard deviation during the 30 runs of 1M iterations for Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ in the uniform random, degree-based setting for large graphs.

| Graph | Fast SW-GSEMO3D | | | | Fast SW-GSEMO3D$_0$ | | | |
|---|---|---|---|---|---|---|---|---|
| | uniform | | degree | | uniform | | degree | |
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| ca-CSphd | 665 | 40.555 | 670 | 37.727 | 225 | 16.829 | 230 | 15.616 |
| ca-HepPh | 2770 | 124.786 | 2713 | 166.804 | 125 | 15.561 | 128 | 20.372 |
| ca-AstroPh | 3608 | 167.880 | 3602 | 132.344 | 140 | 26.422 | 144 | 25.647 |
| ca-CondMat | 5196 | 130.968 | 5245 | 109.568 | 107 | 20.817 | 104 | 19.662 |

We recall the chance-constrained dominating set problem. The input is given as a graph $G = (V, E)$ with node set $V = \{v_1, \ldots, v_n\}$ and weights on the nodes. The goal is to compute a set of nodes $D \subseteq V$ of minimal weight such that each node of the graph is dominated by $D$, i.e. either contained in $D$ or adjacent to a node in $D$. In our setting the weight $w_i$ of each node $v_i$ is chosen independently of the others according to a normal distribution $N(\mu_i, \sigma_i^2)$ and the goal is to find a dominating set of minimal weight with respect to Equation (2) for a given confidence level of $\alpha$. The constraint function $c(x)$ counts the number of nodes dominated by the given search point $x$. As each node needs to be dominated in a feasible solution, $x$ is feasible iff $c(x) = n$ holds and therefore work with the bound $B = n$ in the algorithms. We start with an initial solution $x \in \{0,1\}^n$ chosen uniformly at random. We also investigate starting with $x = 0^n$ for Fast SW-GSEMO3D (denoted as Fast SW-GSEMO3D$_0$) in the case of large graphs as this could be beneficial for such settings. We try to give some explanation by considering how the maximal population size differs when starting with a solution chosen uniformly at random or with $0^n$.

As done in Neumann and Witt (2023a, 2025), we consider the graphs cfat200-1, cfat200-2, ca-netscience, ca-GrQc, and Erdos992 consisting of 200, 200, 379, 4158, and 6100 nodes respectively, together with the following categories for choosing the weights. In the uniform setting each weight $\mu(u)$ is an integer chosen independently and uniformly at random in $\{n, \ldots, 2n\}$. The variance $v(u)$ is an integer chosen independently and uniformly at random in $\{n^2, \ldots, 2n^2\}$. In the degree-based setting, we have $\mu(u) = (n + \deg(u))^5/n^4$ where $\deg(u)$ is the degree of node $u$ in the given graph. The variance $v(u)$ is an integer chosen independently and uniformly at random in $\{n^2, \ldots, 2n^2\}$. For these graphs, we use 10M (million) fitness evaluations for each run. We also use the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 1882, 11204, 17903, 21363 nodes already used in the experiments for the bi-objective formulation for maximum coverage. We examine the same uniform random and degree-based setting as described before. We consider 1M fitness evaluations for these graphs in order to investigate the performance on large graphs with a smaller fitness evaluation budget.

For our new sliding window algorithms we use $t_{frac} = 0.9$, $std = 10$, $a = 0.5$, $\epsilon = 0$ based on some preliminary experimental investigations. Furthermore, we consider $10M$ fitness evaluations for all algorithms and results presented in Table 3 and 4 and 1M fitness evaluations for the instances in Table 5 and 6. For each setting, each considered algorithm is run on the same set of 30 randomly generated instances. We use the rank-based Mann Whitney-U test to compute the $p$-value for algorithm pairs to check whether results are statistically significant, which we assume to be the case if the p-value is at most 0.05.

We first consider results for the instances already investigated in Neumann and Witt (2023a). We consider instances with uniform and degree-based weights. Results for the examined algorithms are shown in Table 3 and 4, respectively. We note that the results for the GSEMO2D and GSEMO3D have already been obtained in Neumann and Witt (2023a). Each run that does not obtain a dominating set gets allocated a fitness value of $10^{10}$. We note that this only applies to GSEMO3D for ca-GrQc and Erdos992 and GSEMO3D. It has already been stated in Neumann and Witt (2023a) that GSEMO3D has difficulties in obtaining feasible solutions for these graphs. In fact, it never returns a feasible solution for Erdos992 in both chance constrained settings and only in 1 out of 30 runs for ca-GrQc in both chance constrained settings. Comparing the results of GSEMO2D and GSEMO3D to our new approaches Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$, we can see that all approaches behave quite similar for cfat200-1 and cfat200-2. For ca-netscience, there is a slight advantage for our fast sliding window approaches that is statistically significant when compared to GSEMO2D and GSEMO3D, but no real difference on whether the sliding window approach starts with an initial solution chosen uniformly at random or with the search point $0^n$. Both Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ show their real advantage for the larger graphs ca-GrQc and Erdos992 where the 3-objective approach GSEMO3D was unable to produce feasible solutions. On these instance GSEMO2D is clearly outperformed by the sliding window 3-objective approaches.

Results for the instances based on the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 1882, 4158, 6100, 11204, 17903, 21363 nodes are shown in Table 5 for the case of uniform weights and in Table 6 for the case of degree-based weights. Note that the graphs (except ca-CSphd) have more than 10000 nodes and are therefore significantly larger than the ones examined in Neumann and Witt (2023a). As we

are dealing with larger graphs and a smaller fitness evaluation budget of 1M, we also consider the (1+1) EA approach presented in Neumann and Witt (2025). Here each run of the (1+1) EA tackles each value of $\alpha$ (see Equation (2)) separately with a budget of 1M fitness evaluations, which implies the single-objective approach uses a fitness evaluation budget that is ten times the one of the multi-objective approaches. We observe that Fast SW-GSEMO3D$_0$ overall produces the best results. For the smallest graph ca-CSphd, there is no significant difference on whether to start with an initial solution chosen uniformly at random or with the search point $0^n$. However, for the larger graphs ca-HepPh, ca-AstroPh, ca-CondMat consisting of more than 10000 nodes, starting with the initial search point $0^n$ in the sliding window approach is crucial for the success of the algorithm. In particular, we can observe that Fast SW-GSEMO3D starting uniformly at random is performing significantly worse than the (1+1) EA and GSEMO2D for the graphs ca-AstroPh, ca-CondMat consisting of 17903 and 21363 nodes, respectively. All observations hold for the uniform as well as the degree-based chance constrained settings.

As mentioned, starting with $0^n$ in our sliding window approach provides a clear benefit when dealing with large graphs. We have already seen in our analysis that the sliding window approach starts at the constraint value of 0 which gives a partial explanation of its benefit. In order to gain additional insights, we provide in Table 7 the maximum population sizes that the approaches Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ encounter for the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat. We can observe that the maximum population sizes when starting with the search point $0^n$ are significantly smaller than when starting with an initial solution chosen uniformly at random. For the graph ca-CondMat, the average maximum population size among the executed 30 runs for Fast SW-GSEMO3D is almost by a factor of 50 larger than for Fast SW-GSEMO3D$_0$ (5196 vs. 107). Given that large populations can significantly slow down the progress of the sliding window approach, we regard the difference in maximum population sizes as a clear explanation why Fast SW-GSEMO3D$_0$ clearly outperforms Fast SW-GSEMO3D on the graphs ca-HepPh, ca-AstroPh, and ca-CondMat.

## 5 Conclusions

Many optimization problems in the area of artificial intelligence can be stated in terms of a submodular function under a given set of constraints. Pareto optimization using GSEMO has widely been applied in the context of submodular optimization. We introduced the two-objective Sliding Window GSEMO algorithm which selects an individual due to time progress and constraint value in the parent selection step. Our theoretical analysis provides better runtime bounds for SW-GSEMO while achieving the same worst-case approxmation ratios as GSEMO. Our experimental investigations for the maximum coverage problem shows that SW-GSEMO outperforms GSEMO for a wide range of settings. We also provided additional insights into the optimization process by showing that SW-GSEMO computes significantly more trade-off then GSEMO for instances with random weights or uniform instances with large budgets.

Afterwards, we adapted the sliding window approach to 3-objective formulations. We have shown how to significantly speed and scale up the 3-objective approach for chance constrained problems introduced in Neumann and Witt (2023a) and presented a scalable sliding window approach for it. The new approach provides with high probability the same theoretical approximation quality as the one given in Neumann and Witt (2023a) but within a significantly smaller fitness evaluation budget. Our experimental investigations show that the new approach is able to deal with chance constrained instances of the dominating set problem with up to 20,000 nodes (within 1M iterations) whereas the previous approach given in Neumann and Witt (2023a) was not able to produce good quality (or even feasible) solutions for already medium size instances of around 4,000 nodes (within 10M iterations).

## Acknowledgments

This work has been supported by the Australian Research Council (ARC) through grant FT200100536 and by the Independent Research Fund Denmark through grant 10.46540/2032–00101B.

## References

Antipov, D., Kötzing, T., and Radhakrishnan, A. (2024). Greedy versus curious parent selection for multi-objective evolutionary algorithms. In Proc. of Parallel Problem Solving from Nature (PPSN '24), pages 86–101. Springer.

Bian, C. and Qian, C. (2022). Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In PPSN (2), volume 13399 of Lecture Notes in Computer Science, pages 428–441. Springer.

Crawford, V. G. (2021). Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In Proc. of International Joint Conference on Articial Intelligence (IJCAI '21), pages 1661–1667. ijcai.org.

Das, A. and Kempe, D. (2011). Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In Proc. of International Conference on Machine Learning (ICML '11), pages 1057–1064. Omnipress.

Doerr, B., Doerr, C., Neumann, A., Neumann, F., and Sutton, A. M. (2020). Optimization of chance-constrained submodular functions. In Proc. of AAAI Conference on Artificial Intelligence (AAAI '20), pages 1460–1467. AAAI Press.

Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative drift analysis. Algorithmica, 64:673–697.

Doerr, B. and Neumann, F. (2020). Theory of Evolutionary Computation – Recent developments in discrete optimization. Natural Computing Series. Springer.

Eiben, A. E. and Smith, J. E. (2015). Introduction to Evolutionary Computing, Second Edition. Natural Computing Series. Springer.

Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., and Witt, C. (2010). Approximating covering problems by randomized search heuristics using multi-objective models. Evolutionary Computation, 18(4):617–633.

Friedrich, T. and Neumann, F. (2015). Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evolutionary Computation, 23(4):543–558.

Giel, O. (2003). Expected runtimes of a simple multi-objective evolutionary algorithm. In Proc. of Conference on Evolutionary Computation (CEC '03), volume 3, pages 1918–1925. IEEE.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM, 42(6):1115–1145.

Gu, Y., Bian, C., and Qian, C. (2023). Submodular maximization under the intersection of matroid and knapsack constraints. In Proc. of AAAI Conference on Articial Intelligence (AAAI '23), pages 3959–3967. AAAI Press.

Ishii, H., Shiode, S., Nishida, T., and Namasuya, Y. (1981). Stochastic spanning tree problem. Discrete Applied Mathematics, 3(4):263–273.

Kempe, D., Kleinberg, J. M., and Tardos, É. (2015). Maximizing the spread of influence through a social network. Theory of Computing, 11:105–147.

Khuller, S., Moss, A., and Naor, J. (1999). The budgeted maximum coverage problem. Information Processing Letters, 70(1):39–45.

Knowles, J. D., Watson, R. A., and Corne, D. (2001). Reducing local optima in single-objective problems by multi-objectivization. In Proc. of International Conference on Evolutionary Multi-Criterion Optimization (EMO '93), pages 269–283. Springer.

Korf, R. E. (2010). Artificial intelligence search algorithms. In Atallah, M. J. and Blanton, M., editors, Algorithms and Theory of Computation Handbook: Special Topics and Techniques, page 22.1ff. Chapman & Hall/CRC, 2nd edition.

Kratsch, S. and Neumann, F. (2013). Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica, 65(4):754–771.

Krause, A. and Golovin, D. (2014). Submodular function maximization. In Tractability: Practical approaches to hard problems, pages 71–104. Cambridge University Press.

Laumanns, M., Thiele, L., and Zitzler, E. (2004). Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. IEEE Transactions on Evolutionary Computation, 8(2):170–182.

Ma, X., Huang, Z., Li, X., Qi, Y., Wang, L., and Zhu, Z. (2023). Multiobjectivization of single-objective optimization in evolutionary computation: A survey. IEEE Transactions on Cybernetics, 53(6):3702–3715.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. (2015). Lazier than lazy greedy. In Proc. of AAAI Conference on Artificial Intelligence (AAAI '15), pages 1812–1818. AAAI Press.

Nemhauser, G. L. and Wolsey, L. A. (1978). Best algorithms for approximating the maximum of a submodular set function. Mathematics of Operations Research, 3(3):177–188.

Neumann, A. and Neumann, F. (2025). Optimizing monotone chance-constrained submodular functions using evolutionary multiobjective algorithms. Evolutionary Computation, 33(3):363–393.

Neumann, F. and Wegener, I. (2006). Minimum spanning trees made easier via multi-objective optimization. Nat. Comput., 5(3):305–319.

Neumann, F. and Witt, C. (2010). Bioinspired computation in combinatorial optimization. Natural Computing Series. Springer.

Neumann, F. and Witt, C. (2023a). 3-objective Pareto optimization for problems with chance constraints. In Proc. of Genetic and Evolutionary Computation Conference (GECCO '23), pages 731–739. ACM.

Neumann, F. and Witt, C. (2023b). Fast Pareto optimization using sliding window selection. In Proc. of European Conference on Artificial Intelligence (ECAI '23), volume 372 of Frontiers in Artificial Intelligence and Applications, pages 1771–1778. IOS Press.

Neumann, F. and Witt, C. (2024). Sliding window 3-objective Pareto optimization for problems with chance constraints. In Proc. of Parallel Problem Solving from Nature (PPSN '24), pages 36–52. Springer.

Neumann, F. and Witt, C. (2025). Runtime analysis of single- and multiobjective evolutionary algorithms for chance-constrained optimization problems with normally distributed random variables. Evolutionary Computation, 33(2):191–214.

Osuna, E. C., Gao, W., Neumann, F., and Sudholt, D. (2020). Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. Theoretical Computer Science, 832:123–142.

Qian, C., Liu, D., Feng, C., and Tang, K. (2023). Multi-objective evolutionary algorithms are generally good: Maximizing monotone submodular functions over sequences. Theoretical Computer Science, 943:241–266.

Qian, C., Shi, J., Yu, Y., and Tang, K. (2017). On subset selection with general cost constraints. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI '17), pages 2613–2619. ijcai.org.

Qian, C., Yu, Y., Tang, K., Yao, X., and Zhou, Z. (2019). Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. Artificial Intelligence, 275:279–294.

Qian, C., Yu, Y., and Zhou, Z. (2015). Subset selection by Pareto optimization. In Proc. of Conference on Neural Information Processing Systems (NIPS '15), pages 1774–1782. MIT Press.

Roostapour, V., Neumann, A., Neumann, F., and Friedrich, T. (2022). Pareto optimization for subset selection with dynamic cost constraints. Artificial Intelligence, 302:103597.

Rossi, F., van Beek, P., and Walsh, T., editors (2006). Handbook of Constraint Programming, volume 2 of Foundations of Artificial Intelligence. Elsevier.

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In Proc. of AAAI Conference on Artificial Intelligence (AAAI '15), pages 4292–4293. AAAI Press.

Xie, Y., Harper, O., Assimi, H., Neumann, A., and Neumann, F. (2019). Evolutionary algorithms for the chance-constrained knapsack problem. In Proc. of Genetic and Evolutionary Computation Conference (GECCO '19), pages 338–346. ACM.

Xie, Y., Neumann, A., and Neumann, F. (2020). Specific single- and multi-objective evolutionary algorithms for the chance-constrained knapsack problem. In Proc. of Genetic and Evolutionary Computation Conference (GECCO '20), pages 271–279. ACM.

Zhang, H. and Vorobeychik, Y. (2016). Submodular optimization with routing constraints. In Schuurmans, D. and Wellman, M. P., editors, Proc. of AAAI Conference on Artificial Intelligence (AAAI '16), pages 819–826. AAAI Press.

Zhang, W., Dechter, R., and Korf, R. E. (2001). Heuristic search in artificial intelligence. Artificial Intelligence, 129(1–2):1–4.