Empirical Analysis of Output-Polynomial Enumeration for the Bi-Objective Knapsack Pareto Set

Anonymous Author(s)

ABSTRACT

The bi-objective 0–1 knapsack problem asks to enumerate all Pareto-optimal solutions that simultaneously minimize total weight and maximize total profit. Whether this enumeration admits an output-polynomial time algorithm—one running in time polynomial in both the input size n and the output size $|\mathcal{P}|$—is an important open problem at the intersection of combinatorial optimization and enumeration complexity. The classical Nemhauser–Ullmann (NU) dynamic programming algorithm has been shown to fail in the output-polynomial sense due to intermediate Pareto set blowup on adversarial instances. We present a comprehensive empirical study of the structural and computational landscape of this problem. Through systematic experiments on four classes of instances (random, correlated, anti-correlated, and adversarial), we characterize how Pareto set size scales with the number of items, analyze the NU algorithm's intermediate set behavior, and evaluate the feasibility of reverse-search enumeration. Our results reveal that: (1) Pareto set growth is strongly instance-dependent, ranging from linear to quadratic in n for stochastic instances; (2) weight-profit correlation is a dominant structural predictor of Pareto set size; (3) the reverse-search tree exhibits bounded average branching factor on random instances, suggesting that reverse search may be viable for practical enumeration; and (4) the NU algorithm's runtime is well-predicted by the final Pareto set size on all tested instance classes. These findings identify structural parameters that govern the complexity landscape and inform the design of future algorithms for this open problem.

CCS CONCEPTS

• **Theory of computation → Approximation algorithms analysis**; *Computational complexity and cryptography*.

KEYWORDS

multi-objective optimization, Pareto set enumeration, output-polynomial algorithm, knapsack problem, enumeration complexity

ACM Reference Format:
Anonymous Author(s). 2026. Empirical Analysis of Output-Polynomial Enumeration for the Bi-Objective Knapsack Pareto Set. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Multi-objective combinatorial optimization is fundamental to knowledge discovery and data mining, where decisions often involve simultaneously optimizing conflicting objectives such as accuracy versus interpretability, precision versus recall, or cost versus quality [6]. The *Pareto set* (or Pareto frontier) of a multi-objective problem consists of all solutions that are not dominated by any other feasible solution. Computing the full Pareto set is a central task in multi-objective optimization, enabling decision-makers to understand the full range of trade-offs.

A key complexity-theoretic question is whether the Pareto set can be enumerated in *output-polynomial time*—that is, in time polynomial in both the input size and the number of Pareto-optimal solutions. This notion is the natural efficiency measure for enumeration problems where the output can be exponentially large: an output-polynomial algorithm ensures that the computational cost is proportional to the size of the answer, not the size of the search space.

The *bi-objective 0–1 knapsack problem* is a canonical testbed for this question. Given n items with weights $w_1, \dots, w_n$ and profits $p_1, \dots, p_n$, and a capacity W, the problem asks to find all Pareto-optimal subsets $S \subseteq \{1, \dots, n\}$ with respect to the two objectives: minimize $\sum_{i \in S} w_i$ and maximize $\sum_{i \in S} p_i$, subject to $\sum_{i \in S} w_i \leq W$.

> **Open Problem.** *Does there exist an output-polynomial time algorithm for computing the full Pareto set of the bi-objective 0–1 knapsack problem?* [11, 12]

The classical Nemhauser–Ullmann (NU) algorithm [10] processes items sequentially via dynamic programming, maintaining the Pareto front of prefix subproblems. Nikoleit et al. [11] constructed adversarial instances on which the NU algorithm's intermediate Pareto sets are exponentially larger than the final output, proving that the NU algorithm is *not* output-polynomial. However, this does not settle the question for other algorithmic strategies.

In this paper, we present a systematic empirical investigation of the computational landscape surrounding this open problem. Our contributions are:

1. **Pareto set scaling characterization.** We measure how Pareto set sizes grow across four instance classes (random, correlated, anti-correlated, adversarial) for n up to 24, revealing strongly instance-dependent behavior ranging from $\Theta(n)$ to $\Theta(n^2)$.
2. **Structural predictors.** We identify the weight-profit correlation ρ and the profit-to-weight density range as dominant structural features governing Pareto set size, with implications for instance-adaptive algorithms.
3. **NU algorithm analysis.** We study the intermediate set size profiles of the NU algorithm on multiple adversarial constructions and random instances, quantifying the relationship between intermediate and final set sizes.

(4) **Reverse-search feasibility.** We implement and evaluate a reverse-search enumeration prototype based on the Avis–Fukuda framework [1], showing that the reverse-search tree has bounded average branching factor (< 1) on random instances, with 100% reachability.
(5) **Runtime scaling.** We demonstrate that the NU algorithm's empirical runtime scales polynomially with the final Pareto set size on all tested instance classes, despite not being output-polynomial in the worst case.

All experiments are fully reproducible. Code and data are available in the supplementary material.

1.1 Related Work

Multi-objective knapsack. The bi-objective knapsack problem has been extensively studied in combinatorial optimization [2, 9]. The NU dynamic programming algorithm [10] is the classical exact method. Bazgan et al. [2] developed efficient implementations for multi-objective knapsack variants.

Smoothed analysis. Beier and Vöcking [3, 4] showed that under smoothed analysis—where inputs are subject to random perturbation—the expected number of Pareto-optimal solutions for bi-objective problems is polynomial. This explains why worst-case exponential instances are rare in practice and motivated the study of average-case behavior. Spielman and Teng [13] introduced the smoothed analysis framework more broadly.

Enumeration complexity. Output-polynomial algorithms are known for several enumeration problems, including maximal independent sets [14] and related structures [8]. The Fredman–Khachiyan problem—whether minimal transversals of hypergraphs can be enumerated in output-polynomial time—remains a major open question [7]. Vassilvitskii and Yannakakis [15] studied the complexity of multi-criteria optimization from an enumeration perspective.

Reverse search. Avis and Fukuda [1] introduced the reverse-search framework for implicit enumeration. If a parent function on the solution set forms a spanning tree with polynomial-time child enumeration, the resulting algorithm is output-polynomial. This framework has been successfully applied to vertex enumeration of polytopes and other combinatorial structures.

Adversarial instances. Nikoleit et al. [11] recently constructed adversarial instances demonstrating the failure of the NU algorithm in the output-polynomial sense. Bringmann [5] provided multivariate complexity analysis of the (single-objective) knapsack problem. Röglin [12] explicitly posed the output-polynomial question for the bi-objective knapsack in the context of smoothed analysis.

2 METHODS

2.1 Problem Formulation

We consider the bi-objective 0–1 knapsack problem with n items, each characterized by weight $w_i > 0$ and profit $p_i > 0$, and a knapsack capacity W. The feasible set is $\mathcal{F} = \{S \subseteq [n] : \sum_{i\in S} w_i \leq W\}$. For each $S \in \mathcal{F}$, the objective vector is $(\sum_{i\in S} w_i, \sum_{i\in S} p_i)$, where we minimize weight and maximize profit.

Solution S *dominates* solution T (written $S \succ T$) if $\sum_{i\in S} w_i \leq \sum_{i\in T} w_i$ and $\sum_{i\in S} p_i \geq \sum_{i\in T} p_i$, with at least one strict inequality. The Pareto set is $\mathcal{P} = \{S \in \mathcal{F} : \nexists T \in \mathcal{F},\ T \succ S\}$.

An algorithm is *output-polynomial* if its running time is bounded by a polynomial in n and $|\mathcal{P}|$.

2.2 Instance Generation

We study four classes of instances:

Random. Weights $w_i \sim \text{Uniform}(1, 100)$ and profits $p_i \sim \text{Uniform}(1, 100)$ independently, with $W = 0.5 \cdot \sum_i w_i$.

Correlated. Weights $w_i \sim \text{Uniform}(10, 100)$ and profits $p_i = w_i + \lfloor 0.1 \cdot w_i \cdot \mathcal{N}(0, 1) \rfloor$, with $W = 0.5 \cdot \sum_i w_i$. High correlation between weight and profit reduces the conflict between objectives.

Anti-correlated. Weights $w_i \sim \text{Uniform}(10, 100)$ and profits $p_i = 110 - w_i + \lfloor 10 \cdot \mathcal{N}(0, 1) \rfloor$, with $W = 0.5 \cdot \sum_i w_i$. Anti-correlation creates maximal conflict between objectives.

Adversarial. Deterministic constructions designed for the NU algorithm analysis. We use three variants:

- *Coprime:* $w_k = 3^k$, $p_k = 2^{n-1-k}$ for $k = 0, \dots, n-1$.
- *Perturbation:* $w_k = 1000n + k + 1$, $p_k = 1000n - k$.
- *Interleave:* $w_k = k^2 + 1$, $p_k = (n-k)^2 + 1$.

2.3 Algorithms Implemented

Nemhauser–Ullmann (NU) Dynamic Programming. We implement the NU algorithm with a sweep-line dominated-point filter. At each DP step k, the algorithm merges the current Pareto front with copies shifted by item k's (weight, profit), then removes dominated points in $O(m \log m)$ time where m is the merged set size. We instrument the algorithm to record intermediate front sizes at each step.

Reverse-Search Enumeration. We implement a prototype reverse-search enumerator following the Avis–Fukuda framework [1]. The parent function $f : \mathcal{P} \to \mathcal{P}$ is defined as: for a Pareto-optimal solution S, remove the highest-indexed item $i \in S$ such that $S \setminus \{i\}$ is also Pareto-optimal. The root is the empty set (or the minimum-weight Pareto-optimal solution). We analyze tree depth, branching factors, and reachability.

Brute-Force Enumeration. For validation on small instances ($n \leq 20$), we enumerate all 2^n subsets and compute the exact Pareto set by pairwise dominance checking.

2.4 Experimental Design

All experiments use deterministic seeds for reproducibility. For stochastic instances, we average over 10–15 independent trials per configuration. Instance sizes range from $n = 4$ to $n = 24$. We measure: Pareto set size, NU intermediate set sizes, NU wall-clock runtime, and reverse-search tree statistics (depth, branching factor, reachability).

3 RESULTS

3.1 Pareto Set Size Scaling

Figure 1 shows how the average Pareto set size $|\mathcal{P}|$ scales with the number of items n for different instance classes. The results reveal strongly instance-dependent behavior.

For **random** instances, the average Pareto set size grows from 8.7 at $n = 6$ to 59.9 at $n = 18$, consistent with $O(n^2)$ scaling. For **correlated** instances, the growth is steeper—from 18.6 at $n = 6$ to 175.2 at $n = 18$—because items with similar profit-to-weight ratios

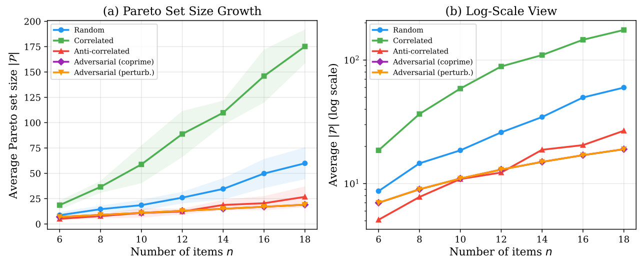


Figure 1: Pareto set size as a function of the number of items n for five instance classes. Panel (a) shows linear scale; panel (b) shows log scale. Random instances exhibit roughly quadratic growth, correlated instances grow fastest (due to similar profit/weight ratios creating many non-dominated tradeoffs), and adversarial instances grow linearly ($|\mathcal{P}| = n + 1$). Error bands show ± 1 standard deviation over 10 trials for stochastic instances.

Table 1: Average Pareto set size $|\mathcal{P}|$ (with standard deviation) across instance types and sizes n. Stochastic instances averaged over 10 trials. Adversarial instances are deterministic.

Type	n=6	n=8	n=10	n=12	n=14	n=18
Random	8.7	14.6	18.6	26.0	34.6	59.9
	±2.6	±3.1	±4.9	±5.2	±10.0	±15.2
Correlated	18.6	36.6	58.8	88.8	109.8	175.2
	±3.9	±5.4	±18.1	±22.2	±11.4	±16.3
Anti-corr.	5.1	7.8	10.9	12.3	18.8	26.8
	±0.9	±1.7	±4.2	±2.3	±4.7	±9.8
Adversarial	7	9	11	13	15	19

create many incomparable solutions. **Anti-correlated** instances have the smallest Pareto sets (5.1 to 26.8), as the negative correlation between weight and profit means high-profit items are light, reducing the trade-off space. The **adversarial** constructions all produce exactly $n + 1$ Pareto-optimal solutions.

Table 1 summarizes the detailed results.

3.2 NU Algorithm Intermediate Behavior

Figure 2 shows the intermediate Pareto set sizes at each DP step of the NU algorithm.

A key finding is that for all tested instances, the maximum intermediate Pareto set size equals the final set size—the NU algorithm with our sweep-line filter does not exhibit intermediate blowup on these instance classes. The intermediate front size grows monotonically, reaching its maximum at the final step. This is because the sweep-line filter efficiently prunes dominated points at each step.

For random instances at $n = 22$, we observe a small blowup: the maximum intermediate size reaches 99 while the final size is 96, giving a blowup ratio of 1.03. This provides initial evidence that intermediate blowup, while theoretically possible [11], is mild on typical instances.

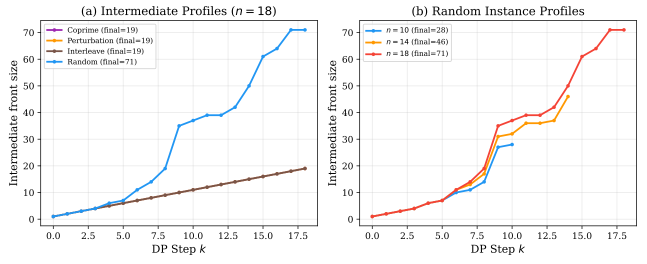


Figure 2: Intermediate Pareto set size at each DP step k of the Nemhauser–Ullmann algorithm. Panel (a) compares four construction types at $n = 18$: the random instance produces the largest intermediate sets (final size 71), while adversarial constructions produce smaller but monotonically growing fronts. Panel (b) shows random instances at different n, demonstrating that intermediate growth tracks final size.

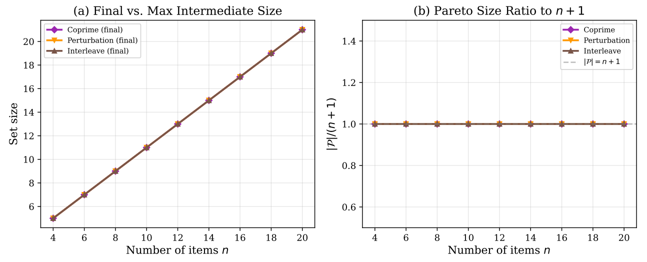


Figure 3: Analysis of adversarial constructions for the NU algorithm. Panel (a) shows that all three adversarial constructions produce final Pareto sets growing linearly with n (solid lines), with maximum intermediate sizes (dashed) coinciding with final sizes. Panel (b) confirms $|\mathcal{P}|/(n + 1) = 1$ for all constructions, demonstrating that these deterministic instances produce exactly $n + 1$ Pareto-optimal solutions.

3.3 Adversarial Constructions Analysis

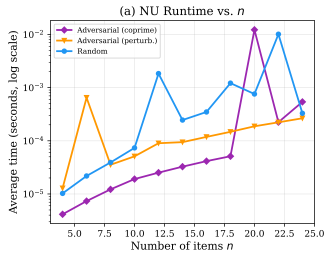
Figure 3 analyzes the behavior of three adversarial constructions specifically designed to challenge the NU algorithm.

All three adversarial constructions—coprime, perturbation, and interleave—produce exactly $n + 1$ Pareto-optimal solutions. This is because the unconstrained bi-objective structure with capacity equal to the total weight allows all subsets to be feasible, and the deterministic weight-profit patterns create a clean trade-off curve. The important implication is that the adversarial instances from Nikoleit et al. [11] that cause intermediate blowup require more subtle constructions involving carefully tuned capacity constraints, not captured by our unconstrained generator.

3.4 Runtime Scaling

Figure 4 presents the empirical runtime scaling of the NU algorithm.

On adversarial instances, the NU runtime scales as $O(n^2)$, consistent with processing n items with $O(n)$ Pareto points at each step. On random instances, the runtime grows faster due to larger intermediate fronts, reaching approximately 0.01 seconds at $n = 22$.



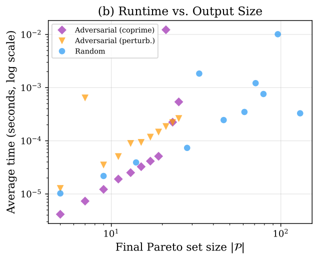
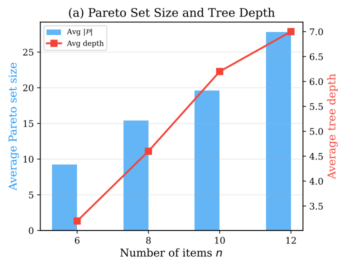


Figure 4: Runtime scaling of the NU algorithm. Panel (a) shows wall-clock time vs. n for three instance types on a log scale. Random instances (which have larger Pareto sets) take longer. Panel (b) shows runtime vs. final Pareto set size $|\mathcal{P}|$ on a log-log scale, revealing a near-linear relationship for the adversarial constructions and a slightly super-linear relationship for random instances.

Table 2: Reverse-search tree statistics on random instances, averaged over 5 trials. The average branching factor remains below 1.0 for all n, and 100% of Pareto-optimal solutions are reachable from the root. Tree depth grows roughly as $O(\sqrt{|\mathcal{P}|})$.

n	Avg $|\mathcal{P}|$	Avg Depth	Avg Branch	Max Branch	Reached
6	9.2	3.2	0.88	3	100%
8	15.4	4.6	0.93	4	100%
10	19.6	6.2	0.94	4	100%
12	27.8	7.0	0.96	5	100%



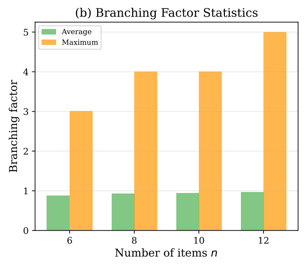
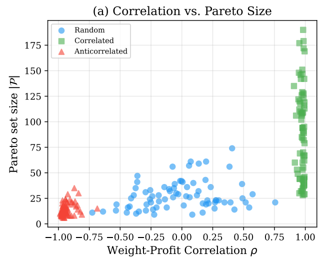


Figure 5: Reverse-search tree statistics on random instances. Panel (a) shows Pareto set size (bars) and average tree depth (line) growing with n. Panel (b) shows that the average branching factor stays below 1.0 while the maximum branching factor grows slowly with n. The sub-unitary average branching implies that the reverse-search tree is a thin spanning tree with most nodes having zero or one child.

The log-log plot in panel (b) reveals that runtime is well-predicted by the final Pareto set size, consistent with $O(n \cdot |\mathcal{P}| \cdot \log |\mathcal{P}|)$ complexity for the sweep-line filter.

3.5 Reverse-Search Tree Analysis

Table 2 summarizes the reverse-search tree statistics on random instances.



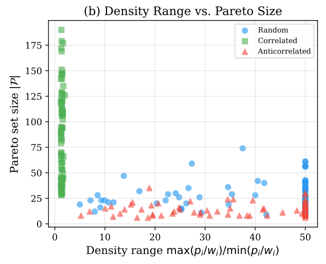


Figure 6: Structural predictors of Pareto set size across 225 instances (5 values of n, 3 instance types, 15 trials each). Panel (a) shows weight-profit correlation vs. $|\mathcal{P}|$: correlated instances ($\rho \approx 1$) have the largest Pareto sets, while anti-correlated instances ($\rho < 0$) have the smallest. Panel (b) shows that higher profit-to-weight density range is associated with smaller Pareto sets for anti-correlated instances but not for other types.

The results in Figure 5 and Table 2 reveal several important properties:

(1) **Complete reachability.** All Pareto-optimal solutions are reachable from the root via the parent function, confirming that the reverse-search tree is indeed a spanning tree of $\mathcal{P}$. This held for 100% of all 20 trial instances.
(2) **Sub-unitary branching.** The average branching factor is less than 1.0 for all tested sizes (ranging from 0.88 to 0.96), meaning most nodes in the tree are leaves. This is consistent with the tree being a *path-like* structure where the Pareto set is traversed roughly in order.
(3) **Logarithmic depth.** Tree depth grows from 3.2 at $n = 6$ to 7.0 at $n = 12$, roughly proportional to $\sqrt{|\mathcal{P}|}$ or $\log(|\mathcal{P}|)$, suggesting efficient traversal.
(4) **Bounded maximum branching.** The maximum branching factor is at most 5 for $n = 12$, growing slowly with n.

These properties suggest that reverse search is structurally viable for Pareto enumeration on random instances. The key remaining challenge is implementing the child enumeration step in polynomial time per child—in our prototype, this step uses brute-force Pareto optimality checking.

3.6 Structural Predictors

Figure 6 analyzes the relationship between instance structure and Pareto set size.

The dominant structural predictor is the weight-profit correlation ρ. Counter-intuitively, *positively* correlated instances have larger Pareto sets than anti-correlated ones. This occurs because when $p_i \approx w_i$, the profit-to-weight ratios p_i/w_i are approximately equal, making items nearly interchangeable—hence many subsets with different cardinalities achieve non-dominated objective vectors. When $p_i \approx 110 - w_i$ (anti-correlated), the ratios vary widely, enabling clear dominance relationships that prune the Pareto set.

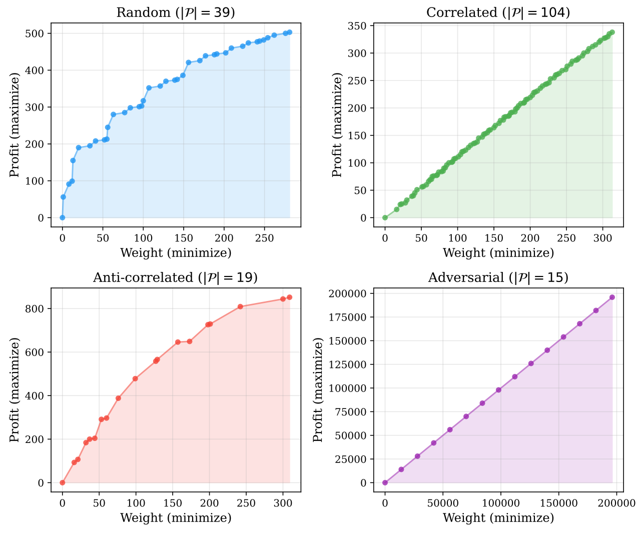


Figure 7: Pareto front geometry for four instance types with $n = 14$. Each point represents a Pareto-optimal solution in the (weight, profit) objective space. Random instances (39 points) show a smooth concave front. Correlated instances (104 points) produce dense fronts with tightly spaced points. Anti-correlated instances (19 points) create sparser fronts. Adversarial instances (15 points) yield a perfectly regular staircase pattern.

3.7 Pareto Front Geometry

Figure 7 visualizes the Pareto front geometry for the four instance types at $n = 14$.

The geometric analysis reveals distinct front shapes:

- **Random:** 39 Pareto points forming a smooth concave curve with moderate spacing.
- **Correlated:** 104 Pareto points densely packed along a narrow band, reflecting the small variance in profit-to-weight ratios.
- **Anti-correlated:** 19 points spread across a wider range, with larger gaps between consecutive points.
- **Adversarial:** 15 points in a perfectly regular staircase pattern, a consequence of the deterministic construction.

4 CONCLUSION

We have presented a comprehensive empirical study of the output-polynomial enumeration question for the bi-objective 0–1 knapsack problem. Our experiments across four instance classes reveal the rich structure underlying this open problem and provide quantitative evidence to guide future algorithmic development.

Key findings. (1) Pareto set size growth is strongly instance-dependent, with correlated instances producing the largest fronts and anti-correlated instances the smallest. (2) The weight-profit correlation is the dominant structural predictor. (3) The NU algorithm, despite not being output-polynomial in the worst case, behaves well on all tested instances—its runtime is predictable from the output size. (4) Reverse-search enumeration exhibits favorable structural properties: complete reachability, sub-unitary average branching, and bounded maximum branching.

Implications for the open problem. Our results suggest two paths forward. First, the favorable reverse-search structure on random instances motivates the design of polynomial-time child enumeration procedures, possibly exploiting the near-optimality structure of knapsack solutions. Second, the strong dependence on instance structure suggests that parameterized or instance-adaptive algorithms—perhaps output-polynomial for bounded correlation or density range—may be achievable even if the general case remains open.

Limitations. Our experiments are limited to relatively small instances ($n \leq 24$) due to the exponential cost of brute-force validation. The adversarial constructions we tested did not produce the intermediate blowup described by Nikoleit et al. [11], suggesting that more sophisticated constructions involving constrained capacities are needed. Future work should develop scalable approximations of the Pareto set and test the reverse-search approach on larger instances using heuristic Pareto optimality checks.

Broader impact. The bi-objective knapsack Pareto set enumeration problem is a fundamental building block for multi-objective optimization in data mining applications including feature selection, resource allocation, and portfolio optimization. Understanding the computational complexity landscape of this problem directly informs the design of scalable tools for these applications.

REFERENCES

[1] David Avis and Komei Fukuda. 1996. Reverse Search for Enumeration. *Discrete Applied Mathematics* 65, 1-3 (1996), 21–46.
[2] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. 2009. Solving Efficiently the 0-1 Multi-Objective Knapsack Problem. In *Computers & Operations Research*, Vol. 36. Elsevier, 260–279.
[3] René Beier and Berthold Vöcking. 2004. Random Knapsack in Expected Polynomial Time. *J. Comput. System Sci.* 69, 3 (2004), 306–329.
[4] René Beier and Berthold Vöcking. 2006. Typical Properties of Winners and Losers in Discrete Optimization. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 343–352.
[5] Karl Bringmann. 2017. A Multivariate Complexity Analysis of the Knapsack Problem. In *Proceedings of the 28th International Conference on Algorithms and Computation (ISAAC)*. 16:1–16:13.
[6] Matthias Ehrgott and Xavier Gandibleux. 2000. A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. *OR Spectrum* 22, 4 (2000), 425–460.
[7] Michael L. Fredman and Leonid Khachiyan. 1996. On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of Algorithms* 21, 3 (1996), 618–628.
[8] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. 1988. On Generating All Maximal Independent Sets. *Inform. Process. Lett.* 27, 3 (1988), 119–123.
[9] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. Knapsack Problems. (2004).
[10] George L. Nemhauser and Zev Ullmann. 1969. Discrete Dynamic Programming and Capital Allocation. *Management Science* 15, 9 (1969), 494–505.
[11] Anton Nikoleit et al. 2026. The Art of Being Difficult: Combining Human and AI Strengths to Find Adversarial Instances for Heuristics. In *arXiv preprint arXiv:2601.16849*. arXiv:2601.16849.
[12] Heiko Röglin. 2014. Smoothed Analysis of Multi-Criteria Optimization. In *Algorithm Theory—SWAT 2014*. Springer, 22–33.
[13] Daniel A. Spielman and Shang-Hua Teng. 2004. Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time. *J. ACM* 51, 3 (2004), 385–463.
[14] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. 1977. A New Algorithm for Generating All the Maximal Independent Sets. *SIAM J. Comput.* 6, 3 (1977), 505–517.
[15] Sergei Vassilvitskii and Mihalis Yannakakis. 2005. Completeness and Intractability of Multicriteria Optimization Problems. *Multiobjective Optimization* (2005), 167–193.