Backbones in Pseudo-Boolean Optimization: Extraction and Analysis

Matías Francia-Carramiñana[1][a], Bryan Alvarado-Ulloa[1][b], Dorit S. Hochbaum[2][c], Bistra Dilkina[3][d], Ricardo Ñanculef[1][e] and Roberto Asín-Achá[1][f]

[1]*Universidad Técnica Federico Santa María, Chile*

[2]*University of California, Berkeley, USA*

[3]*University of Southern California, USA*

{matias.francia, bryan.alvarado, jnancu, roberto.asin}@usm.cl, dhochbaum@berkeley.edu, dilkina@usc.edu

Keywords: Backbone, Pseudo-Boolean Optimization, Combinatorial Optimization.

Abstract: Backbone, the set of variables that are fixed across all optimal solutions, captures key structural properties of combinatorial optimization problems. The backbone provides interpretable indicators of problem hardness and, as demonstrated by recent research in SAT, valuable supervision targets for learning-based methods. Yet, the Pseudo-Boolean Optimization (PBO) domain has lacked dedicated tools for its extraction. Here we introduce three new backbone extractors for Pseudo-Boolean Optimization (PBO): RoundingBack, a PBO-native extractor built on top of RoundingSAT; GuroBack, an extractor built on top of the MILP solver Gurobi; and NapBack, a pipeline that converts PBO instances into SAT using NaPS and delegates the backbone extraction to CadiBack, a backbone extractor for SAT. Additionally, we propose two heuristic variants of RoundingBack: RB-WP (weighted-propagation ordering) and RB-PG (diversity-driven phase guidance). Each extractor demonstrates distinct strengths tailored to different domain applications. On the PBO Competition 2024 OPT-Lin benchmark, RB-PG achieves the highest extraction coverage (223 of 335 instances, 67%), outperforming GuroBack and NapBack. Our experimental evaluation also shows that RoundingBack, GuroBack and NapBack are complementary, and it is often the case that when an extractor fails, another is fit. Beyond algorithmic performance, a large-scale analysis over more than eight thousand instances shows a bimodal distribution distinguishing flexible and rigid problem classes. We also show that the correlation between backbone density and problem hardness is domain-specific. All extracted backbones are publicly released to foster future research in learning-based PBO and structural instance analysis.

1 INTRODUCTION

Combinatorial optimization problems permeate diverse domains (Mironov and Zhang, 2006; Arito et al., 2012). Pseudo-Boolean Optimization (PBO), which extends Boolean satisfiability (SAT) with objective functions and pseudo-Boolean constraints, represents a particularly significant class with applications ranging from circuit verification (Wille et al., 2011) to resource allocation (Ribas et al., 2012).

Developing efficient PBO solving techniques remains a central challenge in computer science. Recent years have witnessed substantial growth in solver diversity, as reflected in the PBO Competition 2024.[1] Unlike SAT competitions—where most solvers adopt Conflict-Driven Clause Learning (CDCL) variants (Biere et al., 2024; Eén and Sörensson, 2006)—PBO solvers employ a broad spectrum of strategies including Branch-and-Bound (Land and Doig, 2010), Branch-and-Cut (Padberg and Rinaldi, 1991), CDCL (Marques-Silva and Sakallah, 1999), PBO-to-SAT translation (Sakai and Nabeshima, 2015), and hybrid approaches (Elffers and Nordström, 2018; Jabs et al., 2024). This methodological diversity yields complementary performance profiles, with solvers excelling in different application domains depending on the structural properties of the instances.

One promising direction is the analysis of the *backbones* of the instances—the sets of variables that retain fixed values across all optimal solutions. The backbone of an instance represents its "frozen core"

[a] https://orcid.org/0009-0000-8680-7347
[b] https://orcid.org/0009-0008-7468-5723
[c] https://orcid.org/0000-0002-2498-0512
[d] https://orcid.org/0000-0002-6784-473X
[e] https://orcid.org/0000-0003-3374-0198
[f] https://orcid.org/0000-0002-1820-9019

[1] https://www.cril.univ-artois.fr/PB24/

and directly encodes its structural rigidity (Slaney et al., 2001). They offer an interpretable, domain-independent lens through which to study problem hardness and can serve as natural supervision targets for DL models, providing ground-truth signals aligned with fundamental properties of the solutions as demonstrated in SAT (Wang et al., 2024).

In this work, we present a large-scale empirical study of backbone properties across more than eight thousand PBO instances drawn from around forty domains. As illustrated in Figure 1, we find that backbone density often exhibits strong Spearman correlation (or inverse correlation) with solver difficulty across diverse domains. In some domains, denser backbones make instances harder, while in others its presence is favorable for the solving process.

Despite their central role in understanding optimization structure, backbone identification remains challenging. For NP-hard problems, exact backbone extraction is Co-NP-Hard (Janota et al., 2015). While efficient extraction methods exist for SAT, the PBO domain still lacks dedicated backbone extraction techniques—a significant gap given PBO's importance across many optimization applications.

This paper addresses this gap by introducing three new PBO backbone extractors: GUROBACK that relies on the MILP solver Gurobi, NAPBACK that is an straight forward pipeline that converts a PBO instance to SAT (using NaPs (Sakai and Nabeshima, 2015)) and extracts the backbone using Cadiback (Biere et al., 2023) (an extractor for SAT), and finally ROUNDINGBACK a native backbone extractor for PBO built on top of RoundingSat (Elffers and Nordström, 2018), a PBO solver that integrates conflict-driven search with cutting-plane reasoning and LP relaxations via SoPlex (Bolusani et al., 2024).

We extend the PBO native extractor ROUNDINGBACK (referred to as RB-BASE) with two heuristic variants that significantly improve performance: RB-WP, which prioritizes variables by their propagation activity, and RB-PG, which employs diversity-driven phase selection (Nadel, 2011) to explore the solution space more effectively.

Our comprehensive evaluation on 335 instances from the PBO Competition 2024 OPT-Lin track demonstrates that RB-PG achieves the highest coverage (223 instances, 67%), followed by GUROBACK and NAPBACK. This is notable, since Gurobi is the solver in the portfolio that performs the best when solving these PBO instances.

Our contributions are:

1. We introduce ROUNDINGBACK, the first native backbone extractor for PBO, along with two heuristic variants.
2. We introduce GUROBACK and NAPBACK, new PBO-backbone extractors based on Gurobi and established SAT-technologies respectively.
3. We demonstrate that ROUNDINGBACK variants outperform GUROBACK and NAPBACK on the PBO Competition 2024 OPT-Lin track benchmark set, while revealing meaningful domain-specific complementarity among all methods—suggesting future potential for hybrid or portfolio approaches.
4. We release a public repository of backbones for *8,351* PBO instances from recent competitions.

The remainder of this paper is organized as follows: Section 2 discusses related work and background. Section 3 details our extraction methods and heuristic variants. Section 4 presents our experimental evaluation, characterizes the PBO Competition 2024 dataset, and analyzes backbone distributions. Section 5 concludes with a summary of contributions and future research directions.

2 RELATED WORK

2.1 Backbone Extraction in SAT

For decisional problems, the backbone comprises pairs $\langle \text{variable}, \text{value} \rangle$ that appear in every *satisfying* assignment. In Boolean Satisfiability (SAT), this represents literals (a variable or its negation) that must be true in all models of a given instance. Early studies of backbones such as (Parkes, 1997; Monasson et al., 1999; Achlioptas et al., 2000) showed that backbone size correlates with propositional problem hardness. For many problem families, larger backbones indicate more tightly constrained problems, increasing solution difficulty (Codish et al., 2013). Determining a SAT instance's backbone is Co-NP-Complete (Kilby et al., 2005).

The backbone concept extends to optimization problems as pairs $\langle \text{variable}, \text{value} \rangle$ present in all *optimal* solutions. While some domains (e.g., graph coloring) show positive correlation between backbone size and hardness, others (e.g., blocks world planning, TSP) exhibit weaker or inverse relationships (Slaney et al., 2001). Nevertheless, backbone detection remains valuable: large backbones signal potential misassignments that can lead solvers into unproductive search paths (Slaney et al., 2001), and in some problems (e.g., TSP), early identification of some variables in the backbone enables the decomposition into more tractable subproblems (Schneider et al., 1996).

Janota et al. (Janota et al., 2015) established the algorithmic foundations for SAT backbone extraction, introducing iterative literal testing, chunk-based probing, model-based filtering, and core-based identification. The current state-of-the-art extractor CadiBack (Biere et al., 2023) combines these techniques—adaptive chunking with model filtering and improved model rotation via watched literals—within the CaDiCaL solver (Biere et al., 2024). It starts by identifying an initial model and initializing a set of candidate literals, filtering out those for which the polarity can be flipped (i.e., negating the literal also results in a satisfying assignment). The remaining candidates are examined in chunks using adaptive sizing: if all literals in a chunk are part of the backbone, the chunk size increases; otherwise, the solver returns a new model to filter the candidate list, and the chunk size resets.

2.2 Pseudo-Boolean Optimization Solvers

Pseudo-Boolean Optimization (PBO) serves as an effective intermediate formalism between SAT/MaxSAT and Mixed Integer Programming (MIP), offering powerful modeling capabilities for optimization problems. Standard SAT solving with Conflict-Driven Clause Learning (CDCL) faces limitations when reasoning over combinatorial constraints due to resolution proof system limitations (Haken, 1985). Pseudo-Boolean constraints—linear inequalities of the form $\sum_{i=1}^{n} c_i l_i \geq b$ over Boolean literals l_i with natural coefficients c_i and bound b—provide a more expressive alternative while enabling adaptation of SAT-based techniques.

PBO is NP-hard and no single solver consistently outperforms all others (Smith-Miles et al., 2014), and performance depends heavily on instance characteristics (Pezo et al., 2025). Different solving paradigms have emerged, each with distinct strengths. **RoundingSat** (Elffers and Nordström, 2018) integrates conflict-driven search with cutting planes reasoning, learning linear inequalities instead of clauses. It incorporates ILP techniques, including LP relaxations due to its integration with the SoPlex LP solver (Bolusani et al., 2024). This hybrid approach has influenced several leading PBO solvers (Jabs et al., 2024; Ihalainen et al., 2024; Devriendt, 2020) and provides the foundation for one of our backbone extraction methods. **Gurobi** (Gurobi Optimization, LLC, 2024), a commercial MIP solver, handles PBO through Branch & Bound, cutting planes (Gomory (Gomory, 1958), MIR (Wolsey and Nemhauser, 1999)), and parallel processing, and also constitutes the base for another of our extraction methods. **NaPS** (Sakai and Nabeshima, 2015) converts PBO constraints into CNF using Reduced Ordered Binary Decision Diagrams (ROBDDs) (Bryant, 1986), enabling the use of efficient SAT solvers like MiniSat+ (Eén and Sörensson, 2006). We use Naps inside the other backbone extraction method presented here. Unlike SAT, backbone detection for PBO remains largely unexplored, motivating the development of extraction tools using these diverse solver paradigms.

Data: PBO instance $\varphi = (C, f)$ with constraints C and objective f
Result: Backbone set B
$(sat, \sigma, S^*) \leftarrow \text{solve}(\varphi)$;
assert $sat = \text{true}$;
$C \leftarrow C \cup \{f(x) = S^*\}$;
$B \leftarrow \emptyset$, $k \leftarrow 1$;
$\Lambda \leftarrow [\ell \in \sigma]$; `# RB-WP sorts by propagation score`
while $\Lambda \neq \emptyset$ **do**
  $k' \leftarrow \min(k, |\Lambda|)$;
  $\Gamma \leftarrow$ pick k' literals from Λ;
  $\varphi \leftarrow (C \cup \{\sum_{\ell \in \Gamma} \bar{\ell} \geq 1\}, \emptyset)$;
  $(sat', \sigma') \leftarrow \text{solve}(\varphi)$; `# RB-PG applies pguide`
  if $sat' = true$ **then**
    $\Lambda \leftarrow \Lambda \setminus \{\ell \in \Lambda \mid \sigma(\ell) \neq \sigma'(\ell)\}$;
    $\Lambda \leftarrow \Lambda \setminus \{\ell \in \Lambda \mid \varphi \cup \{\bar{\ell}\}$ is satisfiable$\}$;
    $k \leftarrow 1$;
  else
    $B \leftarrow B \cup \Gamma$;
    $\Lambda \leftarrow \Lambda \setminus \Gamma$;
    $C \leftarrow C \cup \{\ell \mid \ell \in \Gamma\}$;
    $k \leftarrow 2 \cdot k$;
  end
end
return B;

Algorithm 1: General backbone extraction algorithm.

3 BACKBONE EXTRACTION FOR PBO

Notation. A *literal* ℓ is either a variable x or its negation $\neg x$; $\bar{\ell}$ denotes the complement of ℓ. An *assignment* σ maps variables to $\{0,1\}$; $\sigma(\ell)$ denotes the truth value of ℓ under σ. A literal ℓ is *flippable* in assignment σ if inverting its polarity satisfiability is maintained, verified by checking that constraints where ℓ appears with the same polarity as in σ remain satisfied when ℓ is negated.

Algorithm 1 generalizes CadiBack (Biere et al., 2023) for PBO. It transforms optimization problems into decision variants by solving the instance $minf(x)$ such that $C(x)$ (a set of constraints over x) is satis-

fied to optimality by first obtaining the optimal solution objective S^*, and reformulating the problem to find x such that $(C(x) \wedge \{f(x) = S^*\})$. The extraction employs adaptive chunking: for each chunk Γ, it tests whether negating at least one literal ($\sum_{\ell \in \Gamma} \bar{\ell} \geq 1$) remains satisfiable. If unsatisfiable, all literals in Γ are backbone members and chunk size doubles; otherwise, non-backbone literals are filtered and chunk size resets to 1. Although Λ is represented as a set for readability, implementations maintain it as an ordered list where ordering influences extraction efficiency (exploited by RB-WP).

3.1 ROUNDINGBACK: Native PBO Backbone Extractor

ROUNDINGBACK is our primary contribution: a native C++ backbone extractor built on RoundingSat (Elffers and Nordström, 2018) that extends CadiBack's algorithm (Biere et al., 2023) to the PBO domain. RoundingSat's hybrid architecture integrates conflict-driven search with cutting planes reasoning and LP relaxations via SoPlex (Bolusani et al., 2024), combining the strengths of CDCL and ILP techniques. Unlike pure LP-based approaches (e.g., Gurobi) or PBO-to-SAT conversions (e.g., NaPS), ROUNDINGBACK operates directly on pseudo-boolean constraints while leveraging this hybrid solver foundation.

Key advantages of the RoundingSat foundation include: (1) *hybrid reasoning* combining exact arithmetic over arbitrary-precision integers with LP relaxations, avoiding the numerical instability of pure floating-point approaches while benefiting from LP bounds; (2) *native pseudo-boolean propagation*, avoiding the overhead and information loss of CNF translation; (3) *cutting planes reasoning*, enabling stronger inference than pure clause learning. The source code is available at:https://github.com/matiasfrancia/roundingback.git.

We extended ROUNDINGBACK with two heuristic variants that significantly improve extraction coverage by guiding variable selection and search exploration. These variants modify different components of the extraction process while preserving correctness. Throughout the paper, we refer to the baseline (Algorithm 1) as RB-BASE, and the two heuristic variants as RB-WP (Weighted Propagation) and RB-PG (Phase Guidance), respectively.

3.1.1 ROUNDINGBACK-WP: Weighted Propagation Ordering

The baseline ROUNDINGBACK uses lexicographical variable ordering when selecting backbone candidates to test, treating all variables equally. ROUNDINGBACK-WP instead prioritizes variables by their *propagation activity* during the initial solution phase. When variable v participates in a reason constraint that propagates another variable, its score increases by:

$$\text{score}(v) \mathrel{+}= \frac{\text{coefficient}(v, \text{reason})}{\max \text{coefficient}(\text{reason})}$$

Variables are then ordered by descending score for backbone testing. This heuristic prioritizes checking variables whose values most frequently cause propagations, as these are more likely to be structural constraints (backbone members) rather than free choices.

3.1.2 ROUNDINGBACK-PG (PGuide): Diversity-Driven Phase Selection

While ROUNDINGBACK-WP reorders backbone candidates, ROUNDINGBACK-PG maintains lexicographical ordering but modifies RoundingSat's internal search behavior to explore the solution space more effectively. Since the backbone extraction algorithm repeatedly solves modified versions of the same PBO instance, ROUNDINGBACK-PG exploits this structure by applying the *pguide* heuristic from Nadel (Nadel, 2011), originally developed for generating diverse SAT solutions.

The pguide heuristic tracks the polarity (0 or 1) assigned to each variable across previously found solutions. For each variable u, it maintains a *potential* $\Pi_u = p_u - n_u$. When selecting a decision variable's polarity during CDCL search, pguide biases *against* the majority polarity: if $\Pi_u > 0$, assign 0; if $\Pi_u < 0$, assign 1. This increases solution diversity, enabling the solver to explore different regions of the search space.

In the backbone extraction context, this diversity-driven exploration reduces redundant solver calls when testing non-backbone variables, as the solver more effectively finds alternative satisfying assignments that demonstrate that a set of variables does not belong to the backbone. While Nadel et al. propose additional variants (e.g., pbcpguide with lookahead propagation), we use the base pguide heuristic as it provides strong performance gains without the computational overhead of more sophisticated alternatives.

We evaluate ROUNDINGBACK's performance against two additional backbone extractors that correspond to different solving paradigms: GUROBACK

and NAPBACK. Both implement sound adaptations of the CadiBack algorithm, and we provide publicly available implementations, enabling comprehensive comparative analysis across LP-based, SAT-conversion, and native PBO approaches. *The source code of all three extractors will be made public upon acceptance.*

3.2 GUROBACK: MILP-Based Extraction

GUROBACK implements Algorithm 1 using Gurobi's C++ API (Gurobi Optimization, LLC, 2024), leveraging the industry-standard commercial solver for mixed-integer programming. It transforms PBO instances into Gurobi's MIP representation, then applies the CadiBack extraction loop. For single-variable chunks, it adjusts variable bounds directly; for multi-variable chunks Γ with solution assignment σ, it partitions Γ into $\Gamma_1 = \{x \in \Gamma \mid \sigma(x) = 1\}$ and $\Gamma_0 = \{x \in \Gamma \mid \sigma(x) = 0\}$, adding the constraint $\sum_{x\in\Gamma_1}(1-x)+\sum_{x\in\Gamma_0} x \geq 1$ to enforce value changes. If the new model remains feasible, then we remove variables from the candidates list by checking against the original solution we got. We do not perform the additional flippable checking. In case the new model becomes infeasible, we update the backbone set, the candidates list, and the restrictions of the original problem accordingly, as shown in Algorithm 1

Gurobi's highly-optimized LP engine, advanced cutting planes (Gomory (Gomory, 1958), MIR (Wolsey and Nemhauser, 1999)), and parallel processing is expected to provide strong performance. On the contrary, Floating-point arithmetic can introduce numerical instability on instances with large coefficients. The source code is available at:https://github.com/bryan-alvarado-ulloa/guroback.

3.3 NAPBACK: SAT-based Extraction

NAPBACK is a pipeline composed by different SAT technologies: Given the optimal solution cost (computed either by Naps, Gurobi or RoundingSAT) (1) PBO instances are transformed to decisional instances as in ROUNDINGBACK and GUROBACK, (2) the decisional PB instance is converted to CNF using a ROBDD-based encoding (Bryant, 1986) with NaPS (Sakai and Nabeshima, 2015); (3) CadiBack (Biere et al., 2023) is run in the SAT instance to extract the SAT backbone; (4) the extracted backbone is converted back to original PBO variables using NaPS's variable correspondence. This pipeline is implemented via Bash scripts coordinating NaPS, CadiBack, and a Python mapping utility.

NAPBACK exploits the mature SAT ecosystem and CadiBack's proven efficiency. However, CNF translations introduce auxiliary variables and obscures PBO-specific structure; ROBDD construction can be costly for large constraints. The source code of NAPBACK is available at:https://github.com/bryan-alvarado-ulloa/napback

4 EXPERIMENTAL EVALUATION AND ANALYSIS

Experimental Setup. All experiments were conducted on compute nodes with Intel Xeon E5-2670 v3 processors (24 cores per node) and 64 GB RAM. We used RoundingSat (commit hash d34b6be), Gurobi v10.0.0rc2, NaPS 1.02b5, and CadiBack 0.2.1 (CaDiCaL version 2.0.0). Each extractor was allocated a one-hour wall-clock time limit per instance.

4.1 Computing the Backbones of the OPT-LIN PBO Dataset

To extract and share the backbones of as many instances as possible from the PBO Competition 2024 OPT-Lin track, we implemented a three-phase procedure on the complete dataset of 14,849 instances:

Phase 1 (Solving): We evaluated three solvers—Gurobi, NaPS, and RoundingSat—on all instances with a 10-minute time limit per instance. This phase identified 10,224 instances (69%) that were solved to optimality by at least one solver.

Phase 2 (Backbone Extraction): For each solved instance, we assigned the fastest solver to its corresponding extractor (GUROBACK for Gurobi, NAPBACK for NaPS, RB-BASE for RoundingSat), with a one-hour time limit. Our intuition was that the fastest solver would also be the most efficient extractor, since each extractor builds on its respective solver.

Phase 3 (Dataset Collection): We successfully extracted backbones for 8,351 instances (56% of the original dataset, 82% of solved instances) that completed within the time limit. The extracted backbones are publicly available at: https://github.com/matiasfrancia/dataset-pbo-backbones.

4.2 Backbone Structure and Problem Hardness Analysis

Figure 1 shows the relationship between backbone density—the proportion of variables fixed across all optimal solutions—and solving time across 31 problem domains. For each domain, we selected the

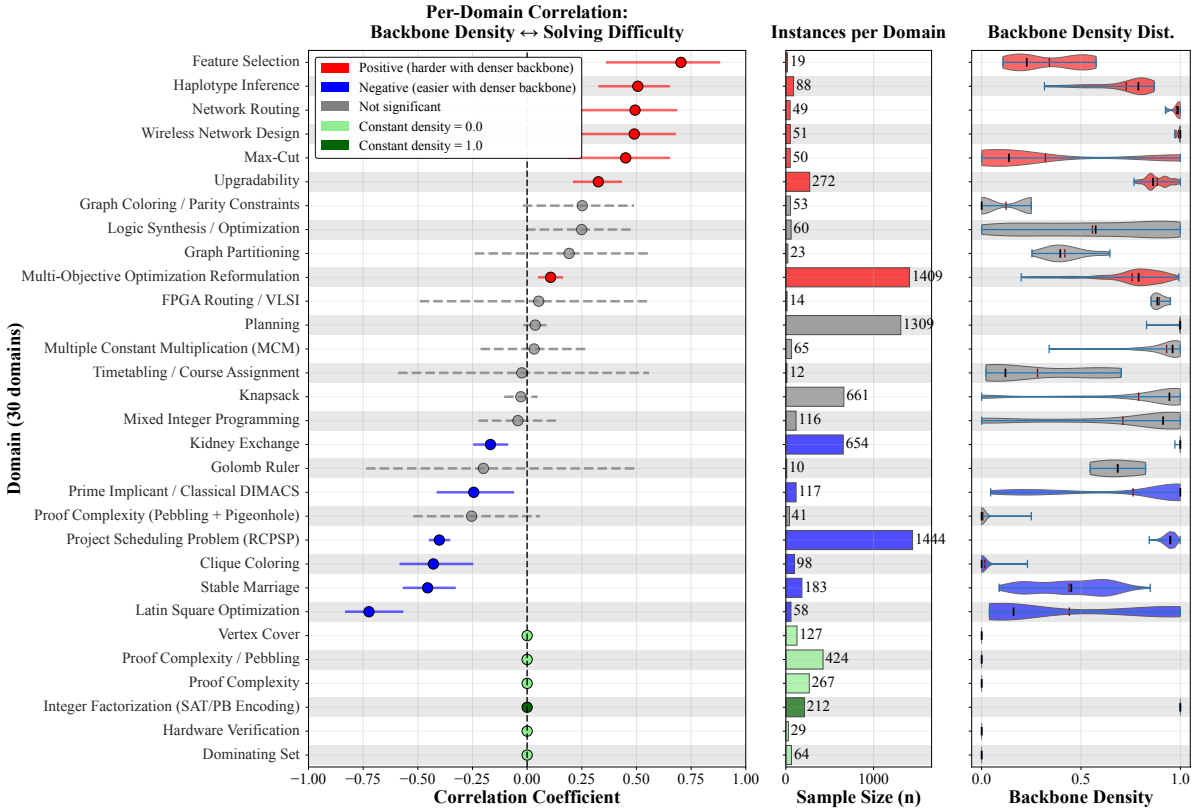

Figure 1: Per-domain correlation between backbone density and solving time. Red indicates positive correlations (larger backbones make instances harder), blue indicates negative correlations (larger backbones make instances easier), and gray indicates non-significant results ($p \geq 0.05$). Light green shows domains where all instances have empty backbones; dark green shows domains where all instances have complete backbones. Error bars represent 95% confidence intervals where the intervals do not cross zero are statistically significant. Sample sizes shown in the middle panel vary considerably across domains.

single best solver among RoundingSat, Gurobi, and NaPS based on average solving time. Correlations were computed using the instances solved by the best solver for each domain, yielding over 8,000 instances with both backbone and runtime data. When the best solver failed to solve an instance, we used results from alternative solvers when available. Domains with fewer than 10 instances were excluded from the analysis. Of the 31 domains examined, 13 show statistically significant correlations ($p < 0.05$), split between 7 positive and 6 negative.

We used Spearman correlation rather than Pearson due to the heavy-tailed distribution of solving times. Statistical significance was assessed via two-tailed tests with $\alpha = 0.05$. Significance depends on both correlation strength and sample size—weaker correlations require larger samples to achieve significance.

Several domains show strong positive correlations, where denser backbones correspond to longer solving times. Haplotype Inference exhibits a clear positive relationship ($\rho = 0.51$, $p < 0.001$), as does Network Routing ($\rho = 0.49$, $p < 0.001$). Conversely, Latin Square Optimization shows a strong negative correlation ($\rho = -0.72$, $p < 0.001$), where higher backbone density corresponds to faster solving. This suggests that, in this domain, fixed structural constraints help prune the search space more effectively.

The dataset exhibits a pronounced bimodal distribution in backbone density: approximately 11% of instances have completely empty backbones (density = 0.0), while another 14% are fully constrained (density = 1.0). This bimodality suggests two distinct problem classes. Under-constrained problems have high flexibility, while fully-constrained instances have unique optimal solutions. These findings establish that backbone density is a domain-sensitive indicator of problem hardness.

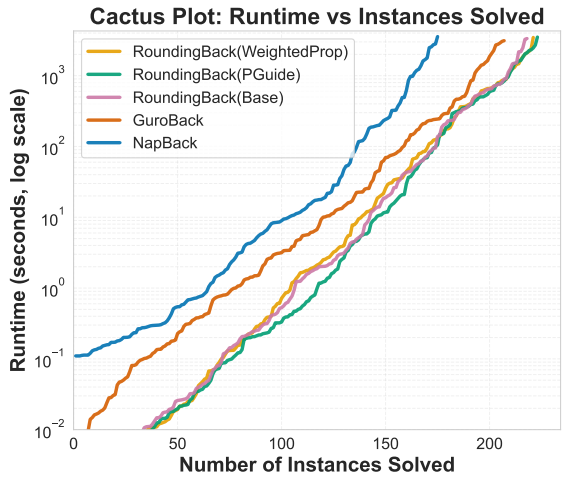

Figure 2: Cactus plot comparing extraction time and instances solved. RB-PG leads with 223 instances, followed by RB-WP (221), RB-BASE (218), GUROBACK (207), and NAPBACK (176).

4.3 Comparing Five Backbone Extractors

To evaluate performance and complementarity of all backbone extraction methods, we conducted experiments on 335 instances from the PBO Competition 2024 OPT-LIN evaluation benchmark for which the optimal objective value is known. This benchmark represents the official instances used to rank competition participants. We compared five extractors: three ROUNDINGBACK variants—RB-BASE, RB-WP, and RB-PG—alongside GUROBACK and NAPBACK. To isolate the backbone extraction phase, we provided each extractor with known optimal values at initialization.

Figure 2 presents a cactus plot summarizing extraction performance. The ROUNDINGBACK variants achieve the highest coverage: RB-PG leads with 223 instances, followed by RB-WP (221) and RB-BASE (218). GUROBACK (207) and NAPBACK (176) show lower coverage but reveal complementary strengths. This is remarkable since Gurobi is the best base solver for this benchmark, yet the by-default extractor is RB-PG.

Figure 3 reveals domain-specific performance patterns. The ROUNDINGBACK variants achieve superior performance in domains like *Upgradability*, *Latin Square Optimization*, and *Planning*. Meanwhile, GUROBACK achieves unique wins in *Max-Cut* and *Vertex Cover*, suggesting MILP-based relaxations uncover backbone implications that SAT-style propagation misses. Similarly, NAPBACK excels in SAT-friendly domains such as *Haplotype Inference*.

The differences among RB-BASE, RB-WP, and RB-PG remain marginal across domains. In contrast, the complementarity between ROUNDINGBACK, GUROBACK, and NAPBACK is substantial: when one method fails, another often succeeds. This suggests that a portfolio combining RB-PG with GUROBACK would cover nearly all observed strengths.

We also identify high-variance domains—such as *Stable Marriage*, *Feature Selection*, *Haplotype Inference*, *Planning* and *Multiple Constant Multiplication*, where extractors behave very differently, highlighting algorithm–domain interactions that warrant future investigation. Conversely, domains like *University of Bologna (MIP)*, *Proof Complexity (Pebbling + Pigeonhole)*, and *Graph Benchmarks from BHOSLIB* remain challenging for all methods, suggesting opportunities for new solving and extraction strategies.

5 CONCLUSIONS AND FUTURE WORK

We introduced ROUNDINGBACK, the first native backbone extractor for Pseudo-Boolean Optimization, along with two heuristic variants: RB-WP (propagation-weighted ordering) and RB-PG (diversity-driven phase guidance), and two further MIP-based and SAT-based extractors named GUROBACK and NAPBACK. On the PBO Competition 2024 OPT-Lin benchmark, RB-PG achieved the highest extraction coverage, outperforming GUROBACK and NAPBACK. Despite this overall superiority, our analysis reveals strong domain-specific complementarity. Beyond extractor design, our large-scale backbone analysis demonstrates that backbone density exhibits strong Spearman correlation (or inverse correlation) with solving time for some domains, and displays a pronounced bimodality separating under-constrained and over-constrained instances. These findings confirm that backbones can serve as robust indicators of problem hardness across heterogeneous PBO domains.

We release backbones for *8,351* competition instances to support learning-based methods and structural analysis. Future work includes portfolio models exploiting method complementarity, learning-based extractor selection, extended benchmark coverage, and integrating backbone prediction into solver pipelines.

Figure 3: Extractor performance across problem domains. Each dot shows the success rate of a backbone extractor on instances from a specific domain. Colors distinguish methods, and pie markers indicate ties. Domains sorted by weighted average success rate.

ACKNOWLEDGEMENTS

The research of the 1st, 2nd, 3rd, 4th and 6th authors was supported in part by the AI4OPT institute NSF award 2112533. The 2nd author is also supported by ANID-Subdirección de Capital Humano/Magíster Nacional/2025 - 22252175. R. Ñanculef acknowledges support from ANID CCTVal (CIA250027). ChatGPT and Gemini were used to improve grammar and spelling.

REFERENCES

Achlioptas, D., Gomes, C., Kautz, H., and Selman, B. (2000). Generating satisfiable problem instances. *AAAI/IAAI*, 2000:256–261.

Arito, F., Chicano, F., and Alba, E. (2012). On the application of sat solvers to the test suite minimization problem. In *International Symposium on Search Based Software Engineering*, pages 45–59. Springer.

Biere, A., Faller, T., Fazekas, K., Fleury, M., Froleyks, N., and Pollitt, F. (2024). Cadical 2.0. In *International Conference on Computer Aided Verification*, pages 133–152. Springer.

Biere, A., Froleyks, N., and Wang, W. (2023). Cadiback: Extracting backbones with cadical. In *26th International Conference on Theory and Applications of*

Satisfiability Testing (SAT 2023). Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., et al. (2024). The scip optimization suite 9.0. *arXiv preprint arXiv:2402.17702*.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691.

Codish, M., Fekete, Y., and Metodi, A. (2013). Backbones for equality. In *Hardware and Software: Verification and Testing: 9th International Haifa Verification Conference, HVC 2013, Haifa, Israel, November 5-7, 2013, Proceedings 9*, pages 1–14. Springer.

Devriendt, J. (2020). Exact: Evaluating a pseudo-boolean solver on maxsat problems. In *MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*, pages 13–14. Department of Computer Science, University of Helsinki. https://helda.helsinki.fi/bitstream/handle/10138/318684/mse20proc.pdf.

Eén, N. and Sörensson, N. (2006). Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26.

Elffers, J. and Nordström, J. (2018). Divide and conquer: Towards faster pseudo-boolean solving. In *IJCAI*, volume 18, pages 1291–1299.

Gomory, R. E. (1958). An algorithm for integer solutions to linear programs. In *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill.

Gurobi Optimization, LLC (2024). Gurobi Optimizer Reference Manual.

Haken, A. (1985). The intractability of resolution. *Theoretical computer science*, 39:297–308.

Ihalainen, H., Berg, J., and Järvisalo, M. (2024). Ipbhs in pb'24 competition. https://www.cril.univ-artois.fr/PB24/descr/IPBHS_Description.pdf.

Jabs, C., Berg, J., and Järvisalo, M. (2024). Pb-oll-rs and mixed-bag in pseudo-boolean competition 2024. Pseudo-Boolean Competition 2024.

Janota, M., Lynce, I., and Marques-Silva, J. (2015). Algorithms for computing backbones of propositional formulae. *Ai Communications*, 28(2):161–177.

Kilby, P., Slaney, J., Thiébaux, S., Walsh, T., et al. (2005). Backbones and backdoors in satisfiability. In *AAAI*, volume 5, pages 1368–1373.

Land, A. H. and Doig, A. G. (2010). *An automatic method for solving discrete programming problems*. Springer.

Marques-Silva, J. P. and Sakallah, K. A. (1999). Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521.

Mironov, I. and Zhang, L. (2006). Applications of sat solvers to cryptanalysis of hash functions. In *Theory and Applications of Satisfiability Testing-SAT 2006: 9th International Conference, Seattle, WA, USA, August 12-15, 2006. Proceedings 9*, pages 102–115. Springer.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., and Troyansky, L. (1999). Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137.

Nadel, A. (2011). Generating diverse solutions in sat. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 287–301. Springer.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100.

Parkes, A. J. (1997). Clustering at the phase transition. In *AAAI/IAAI*, pages 340–345. Citeseer.

Pezo, C., Hochbaum, D., Godoy, J., and Asín-Achá, R. (2025). Automatic algorithm selection for pseudo-boolean optimization with given computational time limits. *Computers & Operations Research*, 173:106836.

Ribas, B. C., Suguimoto, R. M., Montano, R. A., Silva, F., de Bona, L., and Castilho, M. A. (2012). On modelling virtual machine consolidation to pseudo-boolean constraints. In *Advances in Artificial Intelligence–IBERAMIA 2012: 13th Ibero-American Conference on AI, Cartagena de Indias, Colombia, November 13-16, 2012. Proceedings 13*, pages 361–370. Springer.

Sakai, M. and Nabeshima, H. (2015). Construction of an robdd for a pb-constraint in band form and related techniques for pb-solvers. *IEICE TRANSACTIONS on Information and Systems*, 98(6):1121–1127.

Schneider, J., Froschhammer, C., Morgenstern, I., Husslein, T., and Singer, J. M. (1996). Searching for backbones — an efficient parallel algorithm for the traveling salesman problem. *Computer Physics Communications*, 96(2):173–188.

Slaney, J., Walsh, T., et al. (2001). Backbones in optimization and approximation. In *IJCAI*, pages 254–259.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24.

Wang, W., Hu, Y., Tiwari, M., Khurshid, S., McMillan, K. L., and Miikkulainen, R. (2024). Neuroback: Improving CDCL SAT solving using graph neural networks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Wille, R., Zhang, H., and Drechsler, R. (2011). Atpg for reversible circuits using simulation, boolean satisfiability, and pseudo boolean optimization. In *2011 IEEE Computer Society Annual Symposium on VLSI*, pages 120–125. IEEE.

Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and combinatorial optimization*. John Wiley & Sons.