Algorithm Selection with Zero Domain Knowledge via Text Embeddings

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Vienna, Austria

Abstract

We propose a feature-free approach to algorithm selection that replaces hand-crafted instance features with pretrained text embeddings. Our method, ZeroFolio, proceeds in three steps: it reads the raw instance file as plain text, embeds it with a pretrained embedding model, and selects an algorithm via weighted k-nearest neighbors. The key to our approach is the observation that pretrained embeddings produce representations that distinguish problem instances without any domain knowledge or task-specific training. This allows us to apply the same three-step pipeline (serialize, embed, select) across diverse problem domains with text-based instance formats. We evaluate our approach on 11 ASlib scenarios spanning 7 domains (SAT, MaxSAT, QBF, ASP, CSP, MIP, and graph problems). Our experiments show that this approach outperforms a random forest trained on hand-crafted features in 10 of 11 scenarios with a single fixed configuration, and in all 11 with two-seed voting; the margin is often substantial. On the three scenarios with published AutoFolio results from the 2015 ASlib competition, ZeroFolio with two-seed voting matches or outperforms AutoFolio without per-scenario configuration tuning. Our ablation study shows that inverse-distance weighting, line shuffling, and Manhattan distance are the key design choices.

2012 ACM Subject Classification Computing methodologies → Machine learning; Theory of computation → Design and analysis of algorithms

Keywords and phrases algorithm selection, pretrained text embeddings, feature-free, ASlib, k-nearest neighbors, combinatorial problems

Digital Object Identifier 10.4230/LIPIcs...

Supplementary Material *Software (Python package)*: `zerofolio on PyPI`

Funding *Stefan Szeider*: Supported by the Austrian Science Fund (FWF) within the projects 10.55776/COE12 and 10.55776/P36420.

1 Introduction

Algorithm selection is the fundamental problem of choosing the most suitable solver from a portfolio for a given problem instance. Rice [11] formalized this as a mapping from instance features to algorithm performance, establishing a framework that has guided decades of research.

From problem instances, modern algorithm selection systems obtain strong performance by extracting hand-crafted features and training machine learning models to predict solver runtimes. SATzilla [15] pioneered feature-based selection for SAT, later automated by AutoFolio [5]. The ASlib benchmark library [1] provides a standardized evaluation framework with pre-computed features for scenarios across multiple domains.

The central challenge of this pipeline is domain-specific feature engineering. Each new domain requires expert knowledge to design informative features. Feature computation can itself be costly: SATzilla's probing features run SAT solvers internally and can time out or crash on hard instances. Shavit and Hoos [13] recently updated the SATzilla feature extractor, reporting that the original tool failed to extract features from over 20% of modern SAT competition formulas. Moreover, features designed for one domain (e.g., clause-variable ratios for SAT) do not carry over to other domains (e.g., constraint satisfaction or answer set

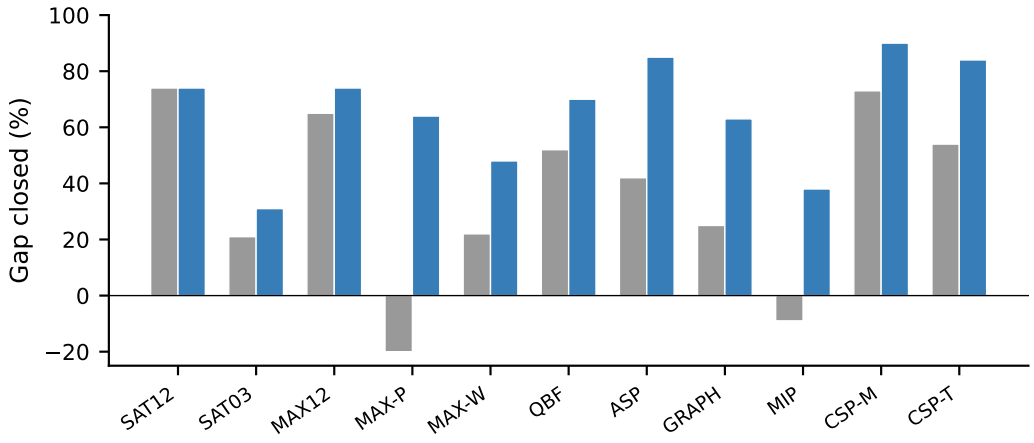

Figure 1 Percentage of the SBS–VBS gap closed. RF on handcrafted features; ZeroFolio. ZeroFolio closes a larger fraction on all 11 scenarios; RF falls below SBS on two.

programming). This leaves practitioners with a trade-off between investing in domain-specific feature engineering and accepting weaker selection performance.

We propose an approach that does not require feature engineering. The basic idea is to read the raw instance file as plain text, embed it with a pretrained text embedding model, and subsequently select an algorithm via weighted k-nearest neighbors. Our results show that pretrained embeddings produce representations that distinguish instance structures well enough for effective selection, without any task-specific training.

It is not obvious a priori that one fixed pipeline, applied to a 10,000-character window of the raw instance file with no per-domain customization, would generalize across formats as structurally diverse as DIMACS CNF, weighted CNF, QDIMACS, MiniZinc models, MPS files, and graph adjacency lists. Our experiments show that, with appropriate serialization and selector choices, it does.

Our contributions are:

- ZeroFolio,[1] a feature-free framework for algorithm selection that replaces hand-crafted features with pretrained text embeddings, requiring zero domain knowledge.
- An evaluation on 11 scenarios across 7 domains, showing that embedding-based k-NN outperforms random forest on hand-crafted features.
- A comparison against AutoFolio on the three scenarios with published 10-fold CV PAR10 numbers, showing that ZeroFolio matches or outperforms AutoFolio without per-scenario configuration tuning.
- An ablation study characterizing the design choices that determine performance.

Figure 1 previews the main result: on all 11 scenarios, ZeroFolio closes a larger fraction of the SBS–VBS gap than the feature-based baseline.

[1] ZeroFolio is available as a Python package: `https://pypi.org/project/zerofolio/`.

2 Related Work

Classical algorithm selection.

Rice [11] formalized algorithm selection as a mapping from a feature space to algorithm performance. SATzilla [15] instantiated this framework for SAT, extracting structural features (clause-variable ratios, variable interaction graphs, probing statistics [9]) and training regression models to predict solver runtimes. AutoFolio [5] automated the configuration of the selection pipeline through algorithm configuration. The ASlib benchmark library [1] standardized the evaluation of algorithm selection systems across domains. We refer the interested reader to Kotthoff [4] for a comprehensive survey. Each of these systems requires hand-crafted, domain-specific instance features as input.

Feature-free algorithm selection.

Loreggia et al. [8] proposed the first feature-free approach to algorithm selection. In their method, problem instances are rendered as grayscale images and classified with a CNN. Salinas-Pinto et al. [12] replaced image rendering with raw text input to a custom Transformer encoder. While this approach improves over the CNN method, it still falls short of the single best solver on SAT Industrial instances. Pellegrino et al. [10] fine-tune a BERT encoder on high-level Essence specifications for constraint programming, obtaining competitive performance within that single domain. In contrast, both Salinas-Pinto et al. [12] and Pellegrino et al. [10] train task-specific neural models on algorithm selection labels. They eliminate features but retain supervised training. Our approach does not require task-specific training and uses a frozen pretrained embedding model.

Neural and LLM-based algorithm selection.

Zhang et al. [17] introduce GraSS, a GNN-based SAT solver selector that represents CNF instances as tripartite literal-clause graphs with hand-designed node features. GraSS achieves competitive results on SAT benchmarks but is inherently domain-specific, requiring CNF parsing, expert-designed graph features, and a trained GNN. Wu et al. [14] use LLMs to embed algorithm source code, combining the resulting representations with hand-crafted instance features from ASlib. Their approach addresses a different aspect of the problem: they embed the algorithm side while still relying on hand-crafted instance features. Gao et al. [3] propose neural solver selection for combinatorial optimization. All these approaches require either domain-specific instance representations or supervised training on runtime data; our method requires neither.

Serialization for language models.

Fatemi et al. [2] study the encoding of graph-structured data as text for large language models and show that the serialization format is a key determinant of model performance. Yin et al. [16] benchmark serialization strategies for structured entity matching and find that random attribute ordering improves robustness. Our line-shuffling strategy is based on this observation. It randomly permutes the lines of an instance file before truncation, so that the embedding model receives a more representative sample of the instance under truncation. Our ablation study shows that line shuffling improves PAR10 by 11% compared to the fixed sequential order.

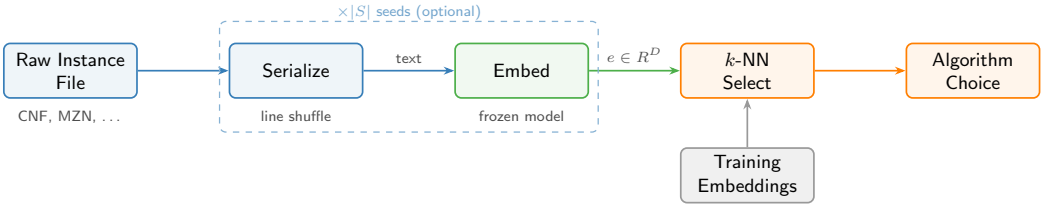

Figure 2 The ZeroFolio pipeline: serialize, embed, select. The same three steps apply unchanged across all problem domains. Multi-seed voting (dashed) is optional.

3 Method

We call our method *ZeroFolio*, reflecting its zero domain knowledge requirement. It proceeds in three steps: serialize the raw instance file as text, embed it with a pretrained model, and select an algorithm via weighted k-NN. Figure 2 illustrates the pipeline.

3.1 Zero Domain Knowledge Serialization

The input to our pipeline is the raw instance file as stored on disk. We treat this file as plain text, without parsing, preprocessing, or feature extraction. Our approach does not require domain-specific knowledge: the same procedure handles CNF formulas, WCNF (MaxSAT), QDIMACS (QBF), MiniZinc models, answer set programs, MPS files, and adjacency lists. For formats with separate model and data files (e.g., MiniZinc `.mzn` + `.dzn`), we concatenate them into a single text.

To handle the token limit of the embedding model, we truncate the file to a character budget b (we use $b = 10{,}000$ throughout). Because truncation discards everything beyond the budget, a fixed file order means that later parts of the instance are never seen. We address this with *line shuffling*: we randomly permute the lines of the file before truncation. This way, different random seeds expose different parts of the instance to the embedding model.

Formally, let f denote an instance file with lines $\ell_1, \ldots, \ell_n$. Given a random seed s, the serialization is

$$\text{serialize}(f, s) = \text{truncate}(\sigma_s(\ell_1, \ldots, \ell_n),\ b)$$

where σ_s is the random permutation determined by s and truncate retains the first b characters. For formats where line order carries no semantic meaning (e.g., CNF clauses), this is a lossless transformation up to truncation.

3.2 Embedding

We pass the serialized text to a pretrained embedding model to obtain a fixed-dimensional vector $e \in \mathbb{R}^D$ via a single API call. We use the embedding model as a black box, with no fine-tuning and no domain adaptation.

3.3 Algorithm Selection via k-NN

Given a test instance with embedding e, we compute a weighted score for each algorithm a from its k nearest training instances under Manhattan distance:

$$\text{score}(a) = \sum_{i=1}^{k} w_i \cdot t_{i,a}, \qquad w_i = \frac{1}{\delta(e, e_i)}$$

Table 1 The 11 ASlib benchmark scenarios used in our evaluation. *Inst.*: number of instances with available raw files. *Algo.*: number of algorithms in the portfolio. *Cutoff*: per-instance time limit in seconds.

Scenario	Domain	Inst.	Algo.	Cutoff (s)	Format
SAT12-ALL	SAT	1367	31	1200	CNF
SAT03-16_INDU	SAT	1887	10	5000	CNF
MAXSAT12-PMS	MaxSAT	875	6	2100	WCNF
MAXSAT-PMS-2016	MaxSAT	450	19	1800	WCNF
MAXSAT-WPMS-2016	MaxSAT	630	18	1800	WCNF
QBF-2016	QBF	825	24	1800	QDIMACS
ASP-POTASSCO	ASP	848	11	600	LP
GRAPHS-2015	Graph	5725	7	10^5	SIP
MIP-2016	MIP	218	5	7200	MPS
CSP-MZN-2013	CSP	3940	11	1800	MZN/XML
CSP-MZN-Time-2016	CSP	100	20	1200	MZN

where δ denotes Manhattan distance, e_i is the embedding of the i-th neighbor, and $t_{i,a}$ is the PAR10 runtime of algorithm a on that neighbor. We select the algorithm with the lowest weighted score. Our ablation study (Section 5.2) shows that this is the most consequential design choice, improving PAR10 by 33% over uniform weighting.

Multi-seed voting.

Line shuffling with different seeds yields distinct embeddings for the same instance. We exploit this by averaging algorithm scores across seeds:

$$\text{score}_{\text{vote}}(a) = \frac{1}{|S|} \sum_{s \in S} \text{score}_s(a)$$

where S is a set of random seeds and $\text{score}_s(a)$ is computed from the embedding produced with seed s. This soft voting reduces the variance introduced by random truncation.

4 Experimental Setup

4.1 Benchmark Scenarios

We consider 11 scenarios from the Algorithm Selection Library (ASlib) [1], spanning 7 domains. Table 1 lists the scenarios with their key characteristics. We start from the union of scenarios used in the Algorithm Selection Competitions [6] and the sunny-as2 evaluation [7]. Specifically, we require a runtime metric, a text-based instance format, publicly available instances, and at least 5 algorithms in the portfolio.

We obtained instance files from public repositories: SAT/MaxSAT competition archives, QBFLIB, and MiniZinc benchmarks. The instance counts reported in the table reflect available files. Algorithm counts and cross-validation splits come from the ASlib data.

4.2 Evaluation Protocol

We use the 10-fold cross-validation splits provided by ASlib. We report PAR10 as the performance metric (lower is better): the runtime if the selected algorithm solves the instance within the cutoff, and 10 times the cutoff otherwise [1].

We compare against two baselines:

- The *single best solver* (SBS): the algorithm with the lowest average PAR10 across all instances.
- A *random forest* (RF) trained on hand-crafted ASlib features, a standard supervised baseline for algorithm selection [1]. We use 100 trees, median imputation for missing features, and standard scaling, with no per-scenario tuning. The RF predicts the best algorithm per instance (classification).

Our standard configuration uses k=10, Manhattan distance, inverse-distance weighting, and line-shuffle serialization with a 10,000-character budget. We fix this configuration across all 11 scenarios and perform no per-scenario tuning.

4.3 Embedding Model

We use `gemini-embedding-2` as the primary embedding model, a proprietary model by Google with 3072 output dimensions. We additionally compare three alternatives across all 11 scenarios: its predecessor `gemini-embedding-001` (3072 dimensions), OpenAI's `text-embedding-3-large` (3072 dimensions, 8k token context), and the open-source `qwen3-embedding-8b` (4096 dimensions, 32k token context). On SAT12-ALL we additionally evaluate `bge-m3` by BAAI (1024 dimensions); it is included only in the ablation. We access Gemini via the Vertex AI API and all other models via OpenRouter. We use all models without fine-tuning. Section 5.3 presents the model comparison.

5 Results

5.1 Main Results

The results are summarized in Table 2. With a single fixed configuration (k=10, Manhattan distance, inverse-distance weighting, single seed), ZeroFolio outperforms the RF in 10 of the 11 scenarios. SAT12-ALL is the only scenario where ZeroFolio falls short of the RF (1156 vs. 1010). Adding two-seed voting closes this gap, yielding 1002 on SAT12-ALL. This gives improvements across all 11 scenarios.

On ASP-POTASSCO, we obtain 539 versus the RF's 775; on QBF-2016, we obtain 1979 versus 2387; on the two CSP scenarios, the improvements exceed 16%. Even on GRAPHS-2015, where the performance gap between SBS and VBS is small (8.8 vs. 8.0), our ZeroFolio closes more of the gap than the RF (8.3 vs. 8.6).

On the three scenarios with published AutoFolio PAR10 from the 2015 ASlib competition (Table 2, AF*), ZF-v2 is within 1% of AutoFolio on ASP-POTASSCO and MAXSAT12-PMS and improves on SAT12-ALL (1002 vs. 1066), despite using smaller instance subsets on two of the three scenarios and performing no per-scenario tuning.

A paired Wilcoxon signed-rank test across the 10 CV folds confirms statistical significance (p<0.05) for 6 of the 10 scenarios where ZeroFolio outperforms RF. The remaining 4 show consistent but non-significant improvements, partly due to high PAR10 variance within folds.

5.2 Ablation Study

We conduct an ablation study on SAT12-ALL to characterize the contribution of each design choice. We use the standard configuration (k=10, Manhattan distance, inverse-distance weighting, line shuffle) and vary the setting of one dimension at a time. Table 4 summarizes the results.

Table 2 Main results: mean PAR10 across 11 ASlib scenarios. ZF: ZeroFolio (k-NN on embeddings, k=10); ZF-v2: ZeroFolio with two-seed voting. AF*: AutoFolio PAR10 from the 2015 ASlib competition (Lindauer et al. [6], Appendix D.6, 10-fold CV); "–" indicates no published number. AF* uses the full scenario (1294 and 1614 instances for ASP-POTASSCO and SAT12-ALL, respectively), whereas our evaluation uses the subset with available raw files (848 and 1367); MAXSAT12-PMS instance counts are nearly identical (876 vs. 875). Bold indicates the best result among RF, ZF, and ZF-v2 per scenario. Gap%: percentage of SBS–VBS gap closed. Significance of ZF vs. RF assessed via paired Wilcoxon signed-rank test across 10 folds ($^{***}p<0.001$, $^{**}p<0.01$, $^{*}p<0.05$).

							Gap%	
Scenario	SBS	RF	AF*	ZF	ZF-v2	VBS	RF	ZF
SAT12-ALL	3066	1010	1066	1156	**1002**	271	**74**	**74**
SAT03-16_INDU	10097	9483	–	**9218**	9187	7152	21	**31**
MAXSAT12-PMS	4899	3748	3559	3658	**3587**	3131	65	**74**
MAXSAT-PMS-2016	2965	3186	–	**2254****	2240	1833	−20	**64**
MAXSAT-WPMS-2016	3893	3614	–	3376	**3292**	2630	22	**48**
QBF-2016	3667	2387	–	1979*	**1958**	1209	52	**70**
ASP-POTASSCO	1015	775	525	539**	**526**	440	42	**85**
GRAPHS-2015	8.8	8.6	–	**8.3*****	8.3	8.0	25	**63**
MIP-2016	3008	3258	–	**1977**	1989	282	−9	**38**
CSP-MZN-2013	9461	5415	–	4524***	**4479**	3950	73	**90**
CSP-MZN-Time-2016	3612	2781	–	**2309**	2418	2062	54	**84**

Inverse-distance weighting has the greatest impact on results: switching to uniform weighting reduces PAR10 from 1156 to 1540 (+33%). Line shuffling is the second most important factor: the raw file order yields 1287 versus 1156 for the shuffled order (+11%). Manhattan distance has a smaller but consistent advantage over cosine (1195 vs. 1156, +3.4%). The naive baseline (raw text, cosine distance, uniform weighting) reaches only 1670, confirming that each design choice contributes.

Varying k shows that k=5 is optimal on SAT12-ALL (1088), while k=10 works well as a cross-domain default. With two-seed voting and k=5, the PAR10 improves to 1002, matching the RF baseline (1010).

To test whether pretrained embeddings are necessary, we replace them with TF-IDF character n-gram features of the same dimensionality (3072) and apply the same k-NN selector. The best TF-IDF variant achieves 1524 on SAT12-ALL. This is better than SBS (3066) but substantially worse than Gemini embeddings (1156). This gap suggests that pretrained representations capture information beyond what surface-level text statistics alone provide, though differences in feature-space geometry may also contribute.

To verify that these choices generalize, we repeat the key ablation dimensions on three additional scenarios (Table 3). Inverse-distance weighting remains important on QBF-2016 but has a negligible impact on ASP-POTASSCO and CSP-MZN-2013. Manhattan and cosine yield nearly identical results on all three scenarios. The standard configuration transfers without modification.

5.3 Embedding Model Comparison

Table 5 compares four embedding models across all 11 scenarios. Gemini 2 (proprietary, 3072 dimensions) outperforms the RF baseline on 9 of 11 scenarios. Its predecessor Gemini 1 (same dimensions) beats RF on 8 of 11; OpenAI text-embedding-3-large (also 3072 dimensions) also beats RF on 8 of 11. Among open-source models, Qwen3-8B (4096 dimensions, 32k

Table 3 Cross-domain ablation (k=10, single seed). Each cell shows PAR10 when one dimension departs from the standard configuration (Manhattan, $1/\delta$). Δ: relative change from standard.

		Cosine		Uniform wt.	
Scenario	Std	PAR10	Δ	PAR10	Δ
SAT12-ALL	1156	1195	+3%	1540	+33%
QBF-2016	1979	1959	−1%	2221	+12%
ASP-POTASSCO	537	537	0%	524	−2%
CSP-MZN-2013	4523	4523	0%	4526	+0%

Table 4 Ablation on SAT12-ALL. Standard configuration: k=10, Manhattan, $1/\delta$ weighting, line shuffle, single seed. Each row varies one dimension from the standard.

Dimension	Variant	PAR10
Standard configuration		1156
Naive baseline (raw + cosine + uniform)		1670
Serialization	raw (no shuffle)	1287
Distance	cosine	1195
Weighting	uniform	1540
k	k=5	1088
k	k=20	1262
Voting (k=5)	2 seeds	1002

token context) beats RF on 6 of 11.

The comparison between Gemini 2, Gemini 1, and OpenAI is informative because all three share the same output dimensionality (3072). Performance differences thus reflect model-specific factors such as training data and architecture rather than dimensionality or context length.

5.4 Hybrid Selection

We combine ZeroFolio with an RF (on ASlib features) via soft voting. For each algorithm, we average the normalized scores from both selectors with equal weight (α=0.5).

On SAT12-ALL, with single-seed ZeroFolio at k=5 (PAR10 1088) competitive with RF (969 in this evaluation), the hybrid achieves 844, improving over both selectors individually. However, on the three scenarios where ZeroFolio already outperforms RF (ASP-POTASSCO, QBF-2016, CSP-MZN-2013), the hybrid outperforms RF but falls short of ZeroFolio alone (e.g., ASP: 685 hybrid vs. 537 ZeroFolio). Hybrid fusion is most beneficial when both selectors are competitive. When one selector dominates, the weaker selector degrades the overall result. Concatenating the two feature vectors into a single representation performs substantially worse than soft voting (PAR10 $\approx$ 1027 on SAT12-ALL). This suggests that keeping the feature spaces separate is more effective than concatenation, possibly because the 3072-dimensional embeddings dominate the 115 hand-crafted features in a joint space.

Table 5 Embedding model comparison (k=10, Manhattan, $1/\delta$, single seed per model). Gem. 2 and Gem. 1: Google, 3072d. OAI: OpenAI, 3072d. Qwen3: open-source, 4096d. Within each row all models use the same embedding seed; the seed used here may differ from that of the ZF column in Table 2, which accounts for small differences in the Gem. 2 column. Bold: best embedding result per scenario.

Scenario	RF	Gem. 2	Gem. 1	OAI	Qwen3
SAT12-ALL	1010	**1156**	1194	1339	1532
SAT03-16_INDU	9483	**9387**	10452	10548	12291
MAXSAT12-PMS	3748	3926	3709	**3635**	3806
MAXSAT-PMS-2016	3186	**2319**	2324	2478	2640
MAXSAT-WPMS-2016	3614	**3378**	3457	3542	3859
QBF-2016	2387	2263	**2217**	2566	3178
ASP-POTASSCO	775	595	**574**	576	646
GRAPHS-2015	8.6	**8.3**	8.5	8.5	8.4
MIP-2016	3258	**2318**	3313	2981	2002
CSP-MZN-2013	5415	4524	4537	**4511**	4680
CSP-MZN-Time-2016	2781	2309	2536	**2188**	2316
Beats RF	—	9/11	8/11	8/11	6/11

6 Discussion

Why does it work?

Pretrained text embedding models are trained on broad web corpora that likely include source code, configuration files, and structured data formats. While a CNF formula is not natural language, it is a structured text file with regularities (token distributions, clause length patterns, comment headers) that such models can plausibly capture. Our results show that embedding-based similarity suffices to identify instances with similar solver performance profiles. Figure 3 visualizes this: a t-SNE projection of SAT12-ALL embeddings reveals clusters dominated by individual solvers, suggesting that the embedding space captures solver-relevant instance structure.

Scope.

Our method applies to any problem domain with text-based instance formats, but binary or image-based formats would require a different serialization step. Within this scope, the same pipeline can handle CNF, WCNF, QDIMACS, MiniZinc, ASP, and MPS files without modification, and no domain expertise is required. Unlike hand-crafted feature extractors, our serialization has no failure modes beyond I/O errors: it reads the file as plain text without parsing. Note that, like all supervised selection approaches, our method still requires labeled runtime data per scenario.

Character budget.

The character budget b has negligible impact on performance: on SAT12-ALL, budgets of 5,000, 10,000, and 30,000 characters yield PAR10 scores of 1083, 1087, and 1087, respectively (k=5). This plateau is consistent with an effective token limit of approximately 2,048 tokens for Gemini Embedding, beyond which additional input text does not affect the embedding.

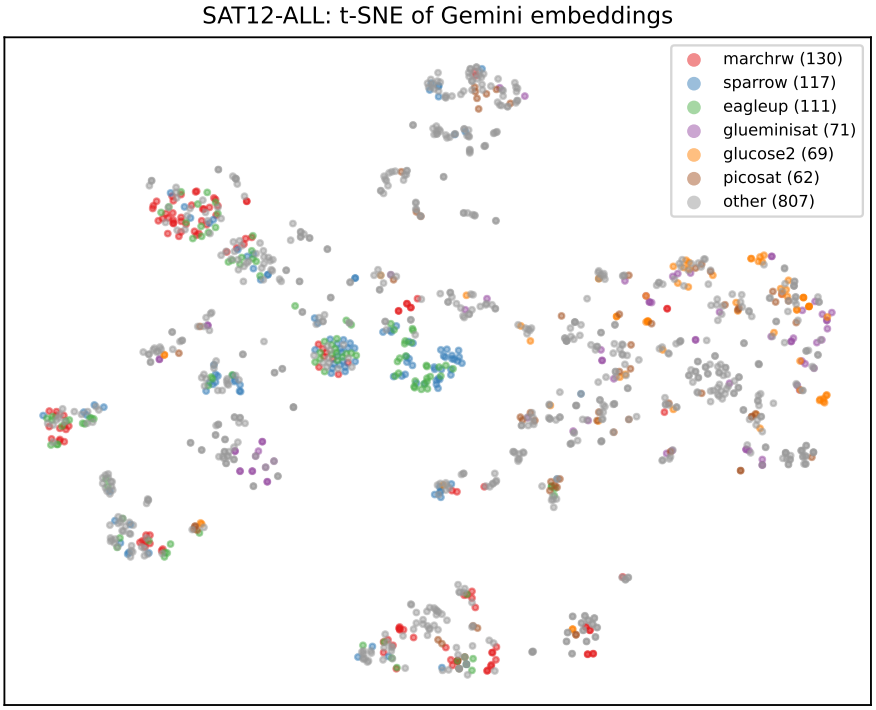

Figure 3 t-SNE projection of Gemini embeddings for SAT12-ALL (1367 instances), colored by best solver. Instances with similar solver preferences cluster together in embedding space.

Limitations.

Our best results rely on proprietary embedding models (Gemini, OpenAI); as noted in Section 5.3, open-source alternatives lag. As open embedding models improve, this gap may narrow. A second limitation is the API cost: embedding all instances for one scenario costs approximately $10 with Gemini. While this is modest compared to the computational cost of running solvers, the reliance on a commercial service introduces risks to availability and reproducibility. AutoFolio's best reported PAR10 on SAT12-ALL is 890 with extensive per-scenario SMAC tuning [5], 11% lower than ZeroFolio's 1002. Under the standard tuning budget of the ASlib competitions, AutoFolio scores 1066 on the same scenario (Table 2), which ZeroFolio improves upon. Per-scenario configuration tuning is compatible with our approach and could, in principle, be layered on top. Finally, we use embedding models out of the box, without fine-tuning; whether task-specific fine-tuning would help, at the cost of the zero-domain-knowledge framing, is outside the scope of this paper.

Feature computation cost.

A practical advantage of our approach is that it avoids the costs and failure modes associated with feature extraction. Recall that the SATzilla feature extractor fails on over 20% of modern formulas (Section 1); feature computation can take minutes per instance [13]. Our entire pipeline (serialize, embed, select) takes approximately one second per instance: serialization and file I/O dominate the runtime, the embedding API call takes 200–500 ms, and k-NN retrieval is negligible. For domains where no established feature extractor exists, our method offers an immediate off-the-shelf alternative.

7 Conclusion

We present ZeroFolio, a feature-free approach to algorithm selection that replaces hand-crafted instance features with pretrained text embeddings, without feature extraction or task-specific training.

Our experiments show that ZeroFolio outperforms a random forest trained on hand-crafted features across 11 scenarios spanning 7 domains using a single fixed configuration, and extends this to every scenario with two-seed voting. On the three scenarios with published AutoFolio competition results, ZeroFolio matches or outperforms AutoFolio without per-scenario tuning. The ablation identifies inverse-distance weighting, line shuffling, and Manhattan distance as the most consequential design choices. ZeroFolio applies unchanged to any problem domain with text-based instance formats.

References

1. Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. Aslib: A benchmark library for algorithm selection. *Artif. Intell.*, 237:41–58, 2016. URL: https://doi.org/10.1016/j.artint.2016.04.003, doi:10.1016/J.ARTINT.2016.04.003.
2. Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=IuXR1CCrSi.
3. Chengrui Gao, Haopu Shang, Ke Xue, and Chao Qian. Neural solver selection for combinatorial optimization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL: https://proceedings.mlr.press/v267/gao25l.html.
4. Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. In Christian Bessiere, Luc De Raedt, Lars Kotthoff, Siegfried Nijssen, Barry O'Sullivan, and Dino Pedreschi, editors, *Data Mining and Constraint Programming - Foundations of a Cross-Disciplinary Approach*, volume 10101 of *Lecture Notes in Computer Science*, pages 149–190. Springer, 2016. doi:10.1007/978-3-319-50137-6_7.
5. Marius Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. Autofolio: An automatically configured algorithm selector. *J. Artif. Intell. Res.*, 53:745–778, 2015. URL: https://doi.org/10.1613/jair.4726, doi:10.1613/JAIR.4726.
6. Marius Lindauer, Jan N. van Rijn, and Lars Kotthoff. The algorithm selection competitions 2015 and 2017. *Artif. Intell.*, 272:86–100, 2019. URL: https://doi.org/10.1016/j.artint.2018.10.004, doi:10.1016/J.ARTINT.2018.10.004.
7. Tong Liu, Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. sunny-as2: Enhancing SUNNY for algorithm selection. *J. Artif. Intell. Res.*, 72:329–376, 2021. URL: https://doi.org/10.1613/jair.1.13116, doi:10.1613/JAIR.1.13116.
8. Andrea Loreggia, Yuri Malitsky, Horst Samulowitz, and Vijay A. Saraswat. Deep learning for algorithm portfolios. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1280–1286. AAAI Press, 2016. URL: https://doi.org/10.1609/aaai.v30i1.10170, doi:10.1609/AAAI.V30I1.10170.
9. Eugene Nudelman, Kevin Leyton-Brown, Holger H. Hoos, Alex Devkar, and Yoav Shoham. Understanding random SAT: beyond the clauses-to-variables ratio. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference,*

CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings, volume 3258 of *Lecture Notes in Computer Science*, pages 438–452. Springer, 2004. doi:10.1007/978-3-540-30201-8_33.

10 Alessio Pellegrino, Özgür Akgün, Nguyen Dang, Zeynep Kiziltan, and Ian Miguel. Transformer-based feature learning for algorithm selection in combinatorial optimisation. In Maria Garcia de la Banda, editor, *31st International Conference on Principles and Practice of Constraint Programming, CP 2025, Glasgow, Scotland, August 10-15, 2025*, volume 340 of *LIPIcs*, pages 31:1–31:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.CP.2025.31, doi:10.4230/LIPICS.CP.2025.31.

11 John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pages 65–118. Elsevier, 1976. doi:10.1016/S0065-2458(08)60520-3.

12 Amanda Salinas-Pinto, Bryan Alvarado-Ulloa, Dorit S. Hochbaum, Matías Francia-Carramiñana, Ricardo Ñanculef, and Roberto Javier As'in Ach'a. Text-based feature-free automatic algorithm selection. In Frans Coenen, Ana Fred, and Jorge Bernardino, editors, *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2024, Volume 1: KDIR, Porto, Portugal, November 17-19, 2024*, pages 267–274. SCITEPRESS, 2024. doi:10.5220/0012913700003838.

13 Hadar Shavit and Holger H. Hoos. Revisiting satzilla features in 2024. In Supratik Chakraborty and Jie-Hong Roland Jiang, editors, *27th International Conference on Theory and Applications of Satisfiability Testing, SAT 2024, Pune, India, August 21-24, 2024*, volume 305 of *LIPIcs*, pages 27:1–27:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.SAT.2024.27, doi:10.4230/LIPICS.SAT.2024.27.

14 Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, and Kay Chen Tan. Large language model-enhanced algorithm selection: Towards comprehensive algorithm representation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 5235–5244. ijcai.org, 2024. URL: https://www.ijcai.org/proceedings/2024/579.

15 Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res.*, 32:565–606, 2008. URL: https://doi.org/10.1613/jair.2490, doi:10.1613/JAIR.2490.

16 Haoteng Yin, Jinha Kim, Prashant Mathur, Krishanu Sarker, and Vidit Bansal. How to talk to language models: Serialization strategies for structured entity matching. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pages 7836–7850. Association for Computational Linguistics, 2025. URL: https://doi.org/10.18653/v1/2025.findings-naacl.437, doi:10.18653/V1/2025.FINDINGS-NAACL.437.

17 Zhanguang Zhang, Didier Chételat, Joseph Cotnareanu, Amur Ghose, Wenyi Xiao, Hui-Ling Zhen, Yingxue Zhang, Jianye Hao, Mark Coates, and Mingxuan Yuan. Grass: Combining graph neural networks with expert knowledge for SAT solver selection. In Ricardo Baeza-Yates and Francesco Bonchi, editors, *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 6301–6311. ACM, 2024. doi:10.1145/3637528.3671627.