Exact Minimum Weight Spanners via Column Generation

Fritz Bökler
Institute of Computer Science, Osnabrück University, Germany

Markus Chimani
Institute of Computer Science, Osnabrück University, Germany

Henning Jasper[1]
Institute of Computer Science, Osnabrück University, Germany

Mirko H. Wagner
Institute of Computer Science, Osnabrück University, Germany

Abstract

Given a weighted graph G, a minimum weight α-spanner is a least-weight subgraph $H \subseteq G$ that preserves minimum distances between all node pairs up to a factor of α. There are many results on heuristics and approximation algorithms, including a recent investigation of their practical performance [21]. Exact approaches, in contrast, have long been denounced as impractical: The first exact ILP (integer linear program) method [49] from 2004 is based on a model with exponentially many path variables, solved via column generation. A second approach [2], modeling via arc-based multicommodity flow, was presented in 2019. In both cases, only graphs with 40–100 nodes were reported to be solvable.

In this paper, we briefly report on a theoretical comparison between these two models from a polyhedral point of view, and then concentrate on improvements and engineering aspects. We evaluate their performance in a large-scale empirical study. We report that our tuned column generation approach, based on multicriteria shortest path computations, is able to solve instances with over 16 000 nodes within 13 min. Furthermore, now knowing optimal solutions for larger graphs, we are able to investigate the quality of the strongest known heuristic on reasonably sized instances for the first time.

2012 ACM Subject Classification Theory of computation → Mathematical optimization; Theory of computation → Network optimization

Keywords and phrases Graph spanners, ILP, algorithm engineering, experimental study

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.30

Related Version *appendix*: `https://arxiv.org/abs/2406.19164` [14]

1 Introduction

Let $G = (V, E)$ be an undirected graph with n nodes and m edges. The distance $d_G(u, v)$ is the length of a shortest path between two nodes u and v in G, possibly subject to positive edge weights $w_e > 0$ for all $e \in E$. A *spanner* is a subgraph $H \subseteq G$ that preserves these distances within some quality degree; see Ahmed et. al [1] for an overview on several variants. In this paper, we consider the original and most prominent variant of a *multiplicative α-spanner* (in the following just *(α-)spanner* for simplicity): for a given *stretch factor* $\alpha \geq 1$, we require that the *stretch constraint* $d_H(u, v) \leq \alpha \cdot d_G(u, v)$ holds for all node pairs $\{u, v\} \in \binom{V}{2}$. The *minimum weight spanner problem (MWSP)* is thus to find such a spanner of minimum total weight. For uniform edge weights, i.e., $w_e = 1$ for all $e \in E$, MWSP is equivalent to

[1] Corresponding author

© Fritz Bökler, Markus Chimani, Henning Jasper, and Mirko H. Wagner;
licensed under Creative Commons License CC-BY 4.0

finding a spanner of minimum size $|E(H)|$. Spanners were first introduced in the context of synchronization in distributed systems and communication networks [42, 43]. Their efficient computation is a highly relevant topic in many applications, e.g., routing problems [48], graph drawings [51], access control hierarchies [11, 35], or passenger assignment [33].

MWSP is known to be NP-hard [16]. Thus, most published algorithms are heuristics or approximations. However, their guarantees often primarily approximate, e.g., the ratio between the weight of the spanner and that of a minimum spanning tree (the so-called *lightness*) – see [1, 21] for an overview. One of the earliest MWSP-algorithms is the *Basic Greedy* (BG) algorithm by Althöfer et al. [5]. Several algorithms were developed in an attempt to improve over BG [8, 9, 26, 46]; some of them allow the spanners to violate the stretch constraint by a factor of $1 + \varepsilon$ [4, 18, 27, 28]. However, BG still is beneficial w.r.t. most guarantees and has been proven to be *existentially optimal* [1]; also, most of the newer algorithms lead to very complex, non-practical implementations. Recently, [21] investigated the practical performance of the most promising of those approaches. They conclude that in almost all cases, BG provides the sparsest and lightest spanners, typically even within the shortest running time.

There are exact algorithms for some special cases of MWSP. Cai and Keil [17] present a linear time algorithm for minimum 2-spanners in unweighted graphs with maximum degree at most four. Kobayashi [36] gives an FPT algorithm for unweighted MWSP that is parameterized in the number of edges that need to be removed to yield H.

For general MWSP, however, there are currently only two published exact algorithms, both solving an integer linear program (ILP): The first algorithm was proposed in 2004 by Sigurd and Zachariasen [49] and requires column generation: They use a *path-based* ILP formulation containing an exponential number of path variables, which are incrementally introduced by solving the *pricing problem*: a particular kind of the *(Resource) Constrained Shortest Path* problem (CSP) [29, 45]. Even in the case of only two resources (such that one is constrained, while the other is minimized), CSP is NP-hard [32], but can often be solved effectively [29, 45]. CSP is regularly used as a building block within column generation, e.g., in vehicle routing [6, 54], aircraft flight assignment [7], and crew scheduling [41]. The approach of [49] was tested on graphs with up to 64 nodes, but not every instance could be solved within a time limit of 30 minutes.

The second exact approach was proposed by Ahmed et al. [2] in 2019. Their model uses an *arc-based* multicommodity flow formulation and has polynomial size. While not directly comparing their approach to [49], they tested their formulation on graphs with up to 100 nodes, on which their solver needed up to 40 hours.

Contribution. We compare the known exact ILP approaches for the spanner problem for the first time, and improve on them. Our goal is to show that, despite the results suggested in literature, exact approaches for the spanner problem are in fact a worthwhile endeavor in practice. On the theory side, we investigate their relative polyhedral strength. From the practical point of view, the arc-based approach is relatively straight-forward to implement, but the path-based approach turns out to be much more interesting and fruitful w.r.t. boosting its performance: we propose several improvements by means of size reduction, new initialization strategies, and stronger pricing algorithms, facilitating concepts from multiobjective optimization. Our modifications allow us to solve instances orders of magnitudes larger than before, e.g., road networks with over 16 000 nodes within 13 minutes. Our path-based column-generation approach is significantly faster and often even yields smaller models than the polynomially-sized (and further tuned) arc-based model. Lastly,

our results allow us to further evaluate the quality of the strongest known spanner heuristic BG [5]. In contrast to previous works, we can now investigate the quality on reasonably-sized instances w.r.t. optimal objective values.

2 Original Column Generation Approach

We first summarize the column generation approach for MWSP [49], and discuss our improvements later in Section 3. The set of *terminal pairs* K contains all node pairs for which the stretch constraint is enforced. In [49], they use $K = V \times V$ consisting of all *ordered* node pairs. Herein, we prefer the sufficient $K = \binom{V}{2}$ of unordered pairs, to avoid redundancy.

The key concept of the model is to establish a binary *path variable* y_P for each path P in G, which is 1 if and only if P is contained in the solution H and at the same time is used to witness that its endpoints u, v satisfy the stretch constraint $d_H(u,v) \leq \alpha \cdot d_G(u,v)$. For all $\{u,v\} \in K$, let $\mathcal{P}_{uv}$ denote the set of all u-v-paths that are no longer than $\alpha \cdot d_G(u,v)$, and let $\mathcal{P} = \bigcup_{\{u,v\}\in K} \mathcal{P}_{uv}$. We can write the ILP model (PB), where decision variables x_e establish the solution H:

$$\text{(PB)} \qquad \min \sum_{e\in E} w_e x_e \tag{PB.a}$$

$$\text{s.t.} \sum_{P\in\mathcal{P}_{uv}} y_P \geq 1 \qquad \forall \{u,v\} \in K \tag{PB.b}$$

$$\sum_{P\in\mathcal{P}_{uv}: e\in P} y_P \leq x_e \qquad \forall e \in E,\ \forall \{u,v\} \in K \tag{PB.c}$$

$$x_e, y_P \in \{0,1\} \qquad \forall e \in E,\ \forall P \in \mathcal{P} \tag{PB.d}$$

To solve this ILP via branch-and-bound (B&B), we need to solve its LP relaxation (i.e., the binary requirements are relaxed to the interval $[0,1]$) at each B&B node. While there are only $(|E|+1)|K|$ constraints, the exponential number of path variables constitutes the main challenge, which is solved using *column generation* – see, e.g., [22, 53] for introductions into exact branch-and-price algorithms. We consider the *restricted master problem* (RMP), which considers all edge variables but only a subset $\mathcal{P}' = \bigcup_{\{u,v\}\in K} \mathcal{P}'_{uv}$ of the path variables, where $\emptyset \neq \mathcal{P}'_{uv} \subseteq \mathcal{P}_{uv}$. The original publication [49] does not describe the initialization of the individual sets P'_{uv}. However, since singletons suffice and there is no heuristic used to yield upper bounds (and corresponding candidate paths), we assume the natural initialization that P'_{uv} consists of a single shortest u-v-path in G, for each $\{u,v\} \in K$. We denote this initialization strategy **1**-SP in the following.

During the solving process, paths $P \in \mathcal{P} \setminus \mathcal{P}'$ are iteratively added to $\mathcal{P}'$, extending the RMP. This procedure is known as column generation. Consider the dual LP of the relaxation of the full primal model (PB), where σ_{uv} and π_e^{uv} are the non-negative dual variables corresponding to the primal constraints (PB.b) and (PB.c) (after canonicalization), respectively. We are mainly interested in the dual constraints

$$\sum_{e\in P} \pi_e^{uv} \leq \sigma_{uv} \qquad \forall P \in \mathcal{P}_{uv}, \forall \{u,v\} \in K. \tag{PB-D}$$

They are the only constraints that could be violated if we do not use a sufficiently large subset $\mathcal{P}'$ when finding a dual solution to the RMP. Consider a pair of optimal primal and dual solutions for the RMP. If the dual solution satisfies all constraints (PB-D), then by the strong duality theorem, the primal solution is optimal for the full LP relaxation of the ILP model (PB), instead of only for the RMP, and we can stop the column generation routine.

Otherwise, the dual solution violates at least one constraint (PB-D) for some path $P \in \mathcal{P}_{uv} \setminus \mathcal{P}'$ and $\{u, v\} \in K$. Our goal is thus to find such a P with negative *reduced cost* $r^{\pi}_{\sigma}(P) \coloneqq \sigma_{uv} - \sum_{e \in P} \pi^{uv}_e$, and add its path variable y_P to $\mathcal{P}'$. This is called the *pricing problem*. Typically, we ask for a path P with smallest possible reduced cost. We solve the pricing problem separately for every $\{u, v\} \in K$. We ask for a shortest u-v-path P in G w.r.t. edge costs π^{uv}_e, while ensuring that $P \in \mathcal{P}_{uv}$. Thus, we have a CSP (constrained shortest path) problem, since the latter property requires us to also consider the original edge weights w_e (distinct from the edge costs π^{uv}_e) and requiring that the identified path has a total weight of at most $B_{uv} = \alpha \cdot d_G(u, v)$. Also, we are only interested in paths of total cost strictly less than σ_{uv}, as those yield negative reduced costs.

BasicCSP. In [49], a very basic CSP algorithm similar to [24] is proposed, with the addition of early discarding of infeasible paths: A *label* is a tuple storing the cost and weight of a path. At each node of G, we store a list of labels (initially $\emptyset$) sorted by cost. We start with label $(0, 0)$ at u, and always proceed with the lowest-cost label: we propagate the label to adjacent nodes (increasing the cost and weight according to the cost and weight of the traversed edge). We discard labels that are dominated (i.e., there is another label element-wise smaller or equal) or guaranteed to exceed the cost or weight limit. This unidirectional search stops once v is reached, yielding a single path $P \in \mathcal{P}_{uv}$ to add to $\mathcal{P}'$. We call this algorithm BasicCSP.

Minor B&B considerations. To speed up the computation, [49] only adds constraints (PB.c) to the RMP once at least one corresponding path variable exists in the RMP. Further, whenever an edge variable x_e is fixed to 0 during branching, no further path containing e needs to be considered in this subproblem. By locally setting the cost of e to σ_{uv}, such paths will be automatically pruned by the CSP computation for $\{u, v\} \in K$.

3 Speedup Techniques and Algorithm Engineering

We are now ready to discuss our techniques to speed up the above approach.

General: Graph representation. All our shortest path implementations (**1**-SP and BasicCSP above, and ***k***-SP and μ-BiA* below) use an array-based forward-star representation [25] of the bidirected graph[2], which is especially suited for cache-efficient shortest path computations.

Preprocessing: Metrication. A minimum weight spanner will never include an edge $\{i, j\} \in E$ with $w_{\{i,j\}} > d_G(i, j)$. Thus, we can safely remove all such edges from G.

Model size and number of CSP calls: Terminal pairs. To observe the stretch constraint for every node pair in an undirected graph, it is natural to set $K = \binom{V}{2}$. However, it is long known that ensuring the stretch constraint for all *adjacent* node pairs already guarantees feasible α-spanners [42]. Thus, $K = E$ suffices. For sparse graphs, this decreases the size of K by a linear factor and, most importantly, also speeds up each pricing step, since we need to run a CSP algorithm for each node pair in K.

[2] Directed graphs are represented by an array of subarrays (one subarray per node); each subarray stores the outgoing edges for the respective node. Undirected graphs are encoded as bidirected graphs.

Initialization: Variable sets. Let H' be an α-spanner computed by a heuristic. For each $\{u, v\} \in K$, we can initialize P'_{uv} by a shortest u-v-path in H'. Clearly, this provides a feasible initial set of paths. Also, H' yields an upper bound for the B&B computation. Comparing several theoretically strong algorithms, [21] identify BG [5] as the by far practically strongest approach. The runner-ups Baswana and Sen [8] and Berman et al. [9] are only worthwhile in certain scenarios. Using the implementations of [21], pilot studies suggest that the latter two are clearly inferior to BG w.r.t. the initialization of the RMP.

Moreover, we generalize the **1**-SP initialization (used in [49]) such that multiple u-v-paths can be included: For all $\{u, v\} \in K$, our ***k***-SP initialization shall compute the k-shortest u-v-paths that are no longer than $\alpha \cdot d_G(u, v)$. We implement a k-shortest path A* algorithm similar to [38]. The goal-directing distance heuristics are also used to discard paths early that are guaranteed to violate the stretch constraints. Finding paths with ***k***-SP is more efficient than with CSP algorithms, but we cannot be certain that added paths locally improve the solution value of the RMP. Note that ***k***-SP can be modified to provide *all* u-v-paths that are no longer than $\alpha \cdot d_G(u, v)$, i.e., the entire set P_{uv}. We call this the *brute force* initialization. We compared combinations of BG with ***k***-SP for $k \in \{5, 10, 20, 50\}$, and brute force to compute initial paths. Pilot studies show that **10**-SP+BG is the best allrounder.

Initialization: Fixing variables. If there is only a single u-v-path P that is no longer than $\alpha \cdot d_G(u, v)$, for some $\{u, v\} \in K$, we can fix its corresponding path variable y_P (and associated edge variables) to 1. Such paths are detected if a **2**-SP algorithm yields only a single path. Thus, solvers using a ***k***-SP initialization with $k > 1$, can fix these variables with no overhead.

Pricing: μ-BiA*. In the pricing step, we ask for a feasible (in terms of the stretch constraint) u-v-path P with negative reduced cost, for each $\{u, v\} \in K$. Formally, any such path suffices to locally improve the current solution; a path with minimum cost yields the steepest descent w.r.t. the objective function. However, thinking about the pivot operation in the simplex algorithm, we know that another path with slightly higher cost may yield an overall better solution as it may allow the corresponding primal variable to be set to a higher value. Thus, we are interested in generalizing the pricing problem from CSP to what we call μ-CSP, for $1 \leq \mu \in \mathbb{N}$: the goal is to find a feasible path for each of the μ smallest among the negative reduced cost values, as long as they exist. The standard CSP is thus identical to 1-CSP.

For 1-CSP, we can use the bidirectional A* (BiA*) label-setting algorithm, as presented by Thomas et al. [50], instead of BasicCSP. We chiefly summarize its main ideas: We simultaneously conduct a forward search from u, and a backward search from $v \in V$. By computing traditional single-source shortest paths (both from u and from v) individually w.r.t. only costs or only weights, we obtain feasible lower bounds for early termination (similar to BasicCSP) and the goal direction. Label dominations can then be considered w.r.t. the induced lower bound estimations of the full u-v-paths. Unprocessed non-dominated labels are held in a min-heap, using a lexicographic comparison on this estimated cost and weight (in this order). Whenever a label is processed, we additionally try to join it with a suitable label from the other search direction at the respective node. The algorithm has identified a minimum-cost feasible path (if it exists) once a joined label is processed for the first time.

We generalize this 1-CSP BiA* algorithm to our scenario for general μ-CSP, and call this variant μ-BiA*. Thereby, we also integrate ideas from multiobjective optimization, in particular, the lexicographic-order-based label-setting algorithm of [40]. The core insight is that by allowing the algorithm to proceed after its standard termination criterion, we obtain a sequence of feasible pair-wise non-dominating u-v-paths with increasing reduced cost, and may stop only after extracting μ such paths (or once the label heap is emptied).

The set of all non-dominated labels of feasible paths is called *Pareto front*. Our μ-BiA* returns a feasible path corresponding to each of the μ lowest-cost labels in the Pareto front. For $\mu = \infty$, the entire Pareto front is represented. However, adding too many paths in a single μ-CSP call may overcrowd the LP with unnecessary variables. Furthermore, even if fewer than μ feasible paths exist, we still have to wait for the heap to be emptied, as deciding if all feasible paths are found is NP-hard [13]. In pilot studies, we compared solvers using μ-BiA*, for $\mu \in \{1, 2, 3, 5, \infty\}$. Generally, using $\mu = \infty$ performs worst, while the other μ-values yield very similar performances and there is no value that performs best across all instances. Overall, however, 3-BiA* appears to be the most promising approach.

Pricing: Pruning μ-CSP calls. During pricing, we generally solve a μ-CSP instance for each terminal pair $\{u, v\} \in K$. This instance is characterized by the edge costs π^{uv} and cost limit σ_{uv}. The vector π^{uv} contains the non-zero dual values for existing constraints (PB.c), and 0 otherwise. For each $\{u, v\} \in K$, we may store the last (π^{uv}, σ_{uv}) that was considered without yielding a feasible path. Subsequently, for this $\{u, v\}$, we only need to compute the current μ-CSP instance, if its new cost limit is larger or some new edge cost is lower than the old stored value. Otherwise, we can *prune* the call without needing to perform any computation. Clearly, the required bookkeeping is very memory consuming, but pilot studies show that it drastically reduces the number of μ-CSP calls. Typically, only a few terminal pairs require many pricing iterations to find all required paths.

4 Multicommodity Flow

The above path-based formulation (PB) can alternatively be understood to model a multicommodity flow problem (MCF, for short) with one commodity for each $\{u, v\} \in K$. A standard way to model MCF is an arc-based formulation, which is the basis for the only other ILP formulation (AB) for MWSP [2]. After its short description and some straight-forward speed-up techniques, we compare (AB) and (PB).

For each terminal pair $\{u, v\} \in K$, we route a unit flow from u to v, modeled via two directed *flow variables* $f^{uv}_{(i,j)}$ and $f^{uv}_{(j,i)}$, for each $\{i, j\} \in E$, and corresponding flow conservation constraints (AB.b). The spanner H is again described by decision variables x_e, for all $e \in E$, which have to be 1 if the edge carries some flow (AB.c). Constraints (AB.d) ensure that any integral u-v-flow has length at most $\alpha \cdot d_G(u, v)$. We use $\mathbb{1}_\varphi$ as an indicator function that is 1 if the boolean expression φ is true, and 0 otherwise.

$$\text{(AB)} \qquad \min \sum_{e \in E} w_e x_e \tag{AB.a}$$

$$\text{s.t.} \sum_{j:\{i,j\} \in E} (f^{uv}_{(i,j)} - f^{uv}_{(j,i)}) = \mathbb{1}_{i=u} - \mathbb{1}_{i=v} \qquad \forall \{u, v\} \in K, \forall i \in V \tag{AB.b}$$

$$f^{uv}_{(i,j)} + f^{uv}_{(j,i)} \leq x_e \qquad \forall \{u, v\} \in K, \forall e = \{i, j\} \in E \tag{AB.c}$$

$$\sum_{\{i,j\} \in E} w_{\{i,j\}} (f^{uv}_{(i,j)} + f^{uv}_{(j,i)}) \leq \alpha \cdot d_G(u, v) \qquad \forall \{u, v\} \in K \tag{AB.d}$$

$$\sum_{j:\{i,j\} \in E} f^{uv}_{(i,j)} \leq \mathbb{1}_{i \neq v} \qquad \forall \{u, v\} \in K, \forall i \in V \tag{AB.e}$$

$$x_e, f^{uv}_{(i,j)}, f^{uv}_{(j,i)} \in \{0, 1\} \qquad \forall \{u, v\} \in K, \forall e = \{i, j\} \in E \tag{AB.f}$$

The non-essential constraints (AB.e), with a right-hand side of 1, are proposed in [2] so that flow would correspond to simple paths; we use the slightly stronger right-hand side $\mathbb{1}_{i \neq v}$.

In [2], several size reduction techniques are used: Firstly, they also use metrication. Secondly, they fix *unreachable* variables: whenever, for any $\{u, v\} \in K$ and any edge direction (i, j), $\{i, j\} \in E$, $d_G(u, i) + w_{\{i,j\}} + d_G(j, v) > \alpha \cdot d_G(u, v)$, they set $f^{uv}_{(i,j)} = 0$. Thirdly, they fix *mandatory* variables: if, for some $\{u, v\} \in K$ and $\{i, j\} \in E$, the edge direction (i, j) is used in every u-v-path observing the stretch constraint, then the corresponding flow variable $f^{uv}_{(i,j)}$ (and consequently $x_{\{i,j\}}$) is fixed to 1.

Our modifications. We reduce the size of the (AB) model by using $K = E$, instead of $K = \binom{V}{2}$, analogous to the discussion in Section 3. We also provide the solver with an upper bound from the BG heuristic. Pilot studies show that our modified solver yields significantly smaller models and is able to solve larger instances than the original (AB) implementation.

Comparing (AB) and (PB). In contrast to (PB), (AB) has polynomial size and can be solved using standard B&B frameworks. Thus, (AB) is considerably easier to implement. Arc-based MCF ILPs can be formulated for a wide range of problems [47]; in practice, however, such models are often inferior to (typically also implementation-wise) more involved formulations, e.g., cut-based formulations [20, 30, 31, 39]. While the model size of (PB) is largely dependent on $|\mathcal{P}|$, and thus on α, (AB) is not. We can expect (PB)'s performance to degrade with increasing stretch factors, while (AB) should have similar performance across all stretches.

It is easy to see that every fractional LP-solution to (PB) can be mapped to a fractional (AB) solution with the same x-variable values (see, e.g., [3]), establishing that (PB) is at least as strong as (AB). The inverse is not true in general as, without fixing unreachable variables, already an unweighted 4-cycle for $\alpha = 2$ yields an LP-solution for (AB) with all $x_e = 0.5$ (yielding a dual bound of 2), while (PB) gives the optimum value of 3. For $\alpha = 2$ on unweighted graphs, fixing unreachable variables always suffices to avoid such a discrepancy. However, even on unweighted instances with variable fixing, we observe for general α:

▶ **Theorem 1.** *The LP-relaxation of* (AB) *is strictly weaker than* (PB) *in general.*

Proof. Let $\alpha = 5$. Consider an unweighted K_5, where we subdivide all edges along a Hamilton cycle once. Let C be the corresponding Hamilton cycle of length 10. For any pair of nodes $u \neq v$, every edge $e \in C$ lies on a u-v-path of length at most 5, and thus no flow variables can be fixed. Setting $x_e = 0.5$ for all $e \in C$, and $x_e = 0$ otherwise, allows a fractional LP-solution for (AB) of weight 5. Consider any fractional solution to (PB). For every degree-2 node u, the variables of its incident edges add up to at least 1 by (PB.b), but also to at most 1, if the objective value were to be at most 5. Thus, for every path P that uses both edges adjacent to some degree-2 node, we have $y_P \leq 0.5$. But for any pair of degree-4 nodes, there is only one path along C of length at most 5, requiring us to either increase the variable values along C or have non-zero variable values for edges in $E \setminus C$. In either case, any LP-solution to (PB) has an objective value > 5, proving the theorem. ◀

5 Experiments

All our implementations are freely available and will be part of the next release of the open source C++ library *Open Graph algorithms and Datastructures Framework* [19]. All instances and detailed data of all experiments are available at [15]. We use the Branch-and-Cut-and-Price framework SCIP 8.0.4 [10] with CPLEX 22.1.1 [23] as the LP solver. All experiments are performed on an Intel Xeon Gold 6134 with 256 GB RAM under Debian 10.2 using gcc 8.3.0-6 (`-O3`). We enforce a time limit of 30 minutes and a RAM limit of 32 GB per instance.

We consider several exact solvers. The re-implementation of the original path-based method by Sigurd and Zachariasen [49] is denoted $\mathsf{PB}_{\mathbf{Orig}}$. $\mathsf{PB}_{\mathbf{Top}}$ uses our improvements, were the most promising configuration was found during pilot studies: We metricize the graph, set $K = E$, and fix mandatory variables; the set of initial paths is created by **10**-SP+BG; we use 3-BiA* and prune μ-CSP calls. Furthermore, we consider our improved version of the arc-based approach, denoted AB^+.

Hypotheses. We formulate our central research questions as falsifiable hypotheses:

- $\mathcal{H}1$. Even with current hardware, $\mathsf{PB}_{\mathbf{Orig}}$ is unable to solve significantly larger instances compared to what was reported in 2004.
- $\mathcal{H}2$. Our speedup techniques used on $\mathsf{PB}_{\mathbf{Orig}}$ are effective and allow $\mathsf{PB}_{\mathbf{Top}}$ to solve instances orders of magnitude larger than previously possible.
- $\mathcal{H}3$. $\mathsf{PB}_{\mathbf{Top}}$ is faster and able to solve larger instances than AB^+.
  - $\mathcal{H}3'$. $\mathsf{PB}_{\mathbf{Top}}$ yields fewer variables than AB^+.
- $\mathcal{H}4$. Even for larger instances, BG produces spanners with near optimum weight.

Generated Instances. We place $n \in N \coloneqq \{20, 50, 100, 200, 500, 1000, 2000\}$ nodes uniformly at random in a unit square; previously, only graphs with $n \leq 100$ could be considered. Edges are introduced in different ways to obtain different graph classes, following the literature on evaluating spanner algorithms. In the generation processes below, we enforce relative densities $\varrho \in R \coloneqq \{2\ln(n)/n, 10\%, 50\%\}$ or constant average node degrees $\delta \in D \coloneqq \{4, 8\}$.

- **ER.** *Erdős-Rényi* graphs were previously used in [2, 21, 49]. For each $n \in N$ and $\varrho \in R$ ($\delta \in D$), each possible edge is included with uniform probability of ϱ ($\delta/(n-1)$) to get the desired relative density (average node degree, respectively).
- **WM.** *Waxman* graphs were previously used [49]. They generalize *random geometric graphs* [44] (where nodes are adjacent if their euclidean distance is below some threshold) and originate from applications in broadcasting [52]. Each edge is included with probability $\gamma e^{-d/(\beta L)}$, where d is the distance between its endnodes and L is the maximum distance between any two nodes. We use similar parameters as [49], but adapted for larger n: For each $n \in N$ and $\varrho \in R$ ($\delta \in D$), we keep $\beta = 0.14$ fixed and vary γ depending on n to achieve the desired relative density (average node degree). See [15] for specific γ values.
- **CMP.** *Complete* graphs (with relative density $\varrho = 100\%$), were previously used in [21, 49]. We consider them for all $n \in N \setminus \{2000\}$.

For each instance, we consider three different edge weight types $\mathcal{W} \coloneqq \{\mathrm{W}_1, \mathrm{W}_{\mathrm{euc}}, \mathrm{W}_{\mathrm{n}}\}$: W_1 denotes uniform weights, i.e., unweighted graphs. $\mathrm{W}_{\mathrm{euc}}$ considers euclidean distances. For W_{n}, edge weights are drawn uniformly at random from $\{1, 2, \ldots n\}$. We do not consider weights $1 \pm 1/3$ (a further of possibility considered in [21]), as the results therein suggest the same behavior as W_1. While W_1 and $\mathrm{W}_{\mathrm{euc}}$ yield metric weights, W_{n} does not in general.

We generate 10 ER and WM graphs for every combination in $N \times \mathcal{W} \times (R \cup D)$, respectively. Similarly, 10 CMP graphs are created for every combination in $(N \setminus \{2000\}) \times \mathcal{W}$. Note that ER are with high probability *expander graphs* [34], i.e., simultaneously sparse and highly connected, while WM generally are not.

Established Instances. We also consider preexisting (often real-world) instances.

- **SteinLib.** The well-known graph library [37], originally created for the Steiner tree problem. As many applications of Steiner trees and spanners overlap, it was used in [21] to evaluate spanner heuristics. We consider the 1017 graphs with at least 100 nodes. On average, they have 1398 nodes and 10 045 edges; 74% of graphs have a relative density $\varrho < 2.5\%$.

Table 1 Top: Share (%) of optimally solved WM graphs. Bold font marks the better of $\mathsf{PB}_{\mathsf{Top}}$ and AB^+. Bottom: median gap of BG to the optimum solution value (or the best found dual bound; cf. top). A "–" indicates that there are instances in that class with no non-trivial lower bound.

weight	W_n						W_{euc}						W_1 (=unweighted)			
density	any $\delta \in D$			any $\varrho \in R$			any $\delta \in D$			any $\varrho \in R$			any $\delta \in D$		any $\varrho \in R$	
α	1.2	2	3	1.2	2	3	1.2	2	3	1.2	2	3	2	3	2	3
$\mathsf{PB}_{\mathsf{Top}}$	**100**	**100**	**99**	**100**	**100**	**90**	**100**	**100**	**75**	**62**	**49**	25	41	10	19	8
$\mathsf{PB}_{\mathsf{Orig}}$	49	34	16	43	27	18	39	16	9	26	14	8	20	6	12	7
AB^+	93	93	86	86	81	66	93	93	74	57	46	**30**	**100**	**62**	**48**	**13**
$\mathsf{PB}_{\mathsf{noM}}$	100	100	98	76	76	62	100	100	74	62	49	24	41	10	19	8
$\mathsf{PB}_{\mathsf{noE}}$	64	57	44	57	57	40	39	19	12	37	17	13	30	9	15	5
n	Median gap of BG to best lower bound in %															
100	0.0	2.1	5.0	0.0	2.3	4.6	0.5	4.0	6.1	2.8	8.6	8.0	5.4	31.1	19.8	129.1
500	0.0	1.3	3.5	0.1	2.0	4.9	0.1	3.2	5.1	1.4	6.3	–	0.6	7.3	4.7	90.2
2000	0.0	0.6	3.2	0.0	1.1	3.6	0.0	1.1	2.3	0.7	–	–	0.1	1.3	3.6	56.4

Road. We consider 10 undirected US *road networks* [12] with $5\,000 < |V| < 17\,000$. Their average node degrees are 2.3–2.9. We consider two different weights: W_{len} and W_t consider the length and travel time (i.e., length/speed-limit) of the road segments, respectively.

Bundling. A recent graph drawing paper on *edge-path bundling* [51] uses spanners (computed via BG) as their central building block. We can now solve their *Airlines* instance ($|V| = 235$, $|E| = 1297$) optimally for $\alpha \in \{1.2, 1.5, 2\}$ within 4 s, 70 s, and 25 min, respectively.

For all above instance sets (both generated and established), we consider stretch factors $\alpha \in \{1.2, 1.5, 2, 3, 5\}$. In the unweighted case, we disregard $\alpha < 2$, since there G itself is the only feasible solution. We consider large stretch factors 3 and 5 mainly to get a broader picture. We stress that in practice, α should typically be assumed to be rather small: e.g. [33] considers detours in networks beyond $\alpha = 1.5$ to be typically too long; [51] are not interested in distortions beyond $\alpha = 2.5$ in graph drawings.

Hypothesis $\mathcal{H}1$. In 2004, [49] conducted experiments on a 933 MHz Intel Pentium III with a time limit of 30 minutes. For any $\alpha \leq 2$, the largest solved WM and ER graphs ($\delta \in \{4, 8\}$) have $n = 64$; CMP graphs are solved for up to $n = 50$. On our modern machine with $\alpha = 2$, $\mathsf{PB}_{\mathsf{Orig}}$ is still unable to solve WM or ER graphs with average node degree $\delta \in \{4, 8\}$ and $n > 100$, and CMP graphs with $n > 50$. Thus, we cannot reject $\mathcal{H}1$ and take it as confirmed.

Hypothesis $\mathcal{H}2$. We investigate the effect of our speedup techniques used in $\mathsf{PB}_{\mathsf{Top}}$. $\mathsf{PB}_{\mathsf{Top}}$ is able to solve many more instances than $\mathsf{PB}_{\mathsf{Orig}}$: in particular, while the former still has very high success rates for Road and SteinLib (Figs. 2a and 2b), the latter solves no such instance at all. On the generated instances, the picture is analogous (see, e.g., Tables 1 and 3): while $\mathsf{PB}_{\mathsf{Orig}}$ solves only the smallest instances as discussed above, $\mathsf{PB}_{\mathsf{Top}}$ typically also solves the largest instances ($n = 2000$) for stretches $\alpha \leq 2$. As expected, instances become harder for the (PB)-approaches with increasing α, as the variable set increases. But even for $\alpha = 3$, $\mathsf{PB}_{\mathsf{Top}}$ solves 70% of the W_n-weighted complete graphs on 1000 nodes. We also observe that unweighted instances are typically much more challenging than weighted instances.

We are thus interested, which of our speedup techniques enables these high success rates. Therefore, we consider variants of $\mathsf{PB}_{\mathsf{Top}}$, where individual features are turned off (they are otherwise identical to $\mathsf{PB}_{\mathsf{Top}}$): $\mathsf{PB}_{\mathsf{noM}}$ does not use metrication, $\mathsf{PB}_{\mathsf{noE}}$ uses $K = \binom{V}{2}$, $\mathsf{PB}_{\mathsf{noFix}}$ does not fix mandatory variables, $\mathsf{PB}_{\mathsf{simpleInit}}$ uses **1**-SP initialization, $\mathsf{PB}_{\mathsf{noPrune}}$ does not prune μ-CSP calls, $\mathsf{PB}_{\mathsf{noBiA}^*}$ uses the BasicCSP algorithm, and $\mathsf{PB}_{\mathsf{1\text{-}BiA}^*}$ uses 1-BiA*. For

Table 2 Average over the speedup factors of $\mathsf{PB}_{\textbf{Top}}$ relative to the specified algorithm, on WM instances with $n \geq 50$ solved by both solvers. Factors below 1 indicate that $\mathsf{PB}_{\textbf{Top}}$ is slower.

weight	W_n						W_{euc}						W_1 (=unweighted)			
density	any $\delta \in D$			any $\varrho \in R$			any $\delta \in D$			any $\varrho \in R$			any $\delta \in D$		any $\varrho \in R$	
α	1.2	2	3	1.2	2	3	1.2	2	3	1.2	2	3	2	3	2	3
$\mathsf{PB}_{\textbf{noFix}}$	1.85	1.85	2.05	1.96	1.79	1.68	1.84	2.14	1.94	1.73	1.22	1.14	2.91	1.79	1.52	1.00
$\mathsf{PB}_{\textbf{simpleInit}}$	0.83	1.18	1.43	0.97	1.53	1.38	0.91	1.63	1.35	1.69	2.93	1.57	0.98	1.30	0.94	1.28
$\mathsf{PB}_{\textbf{noPrune}}$	1.81	2.13	2.62	2.00	2.38	2.33	1.90	2.80	2.38	2.14	1.65	1.15	3.10	1.65	1.70	1.40
$\mathsf{PB}_{\textbf{noBiA}^*}$	0.98	1.07	1.96	1.03	1.12	5.24	0.97	1.72	2.15	1.20	4.07	2.45	1.36	1.20	1.27	0.93
$\mathsf{PB}_{\textbf{1-BiA}^*}$	1.00	1.05	1.05	1.04	1.01	1.09	1.00	1.01	0.85	1.03	0.77	0.66	1.00	1.01	1.01	1.26
AB^+	13.0	18.9	8.07	29.9	34.8	11.7	13.6	8.98	2.01	11.7	1.72	0.18	0.03	0.01	0.05	0.01

brevity, we will not discuss both ER and WM individually, as their behaviors are very similar. Seemingly, WM graphs are slightly more challenging to solve than ER [14, Table A1]. We define the *success rate* as the percentage of instances solved to proven optimality within the time and RAM limit. We start with considering Table 1.

Metrication and terminal pairs. For all W_n-weighted ER, WM, and CMP graphs, the average remaining density (average node degree) after metrication is in $[4.0\%, 9.2\%]$ ($[3.7, 7.9]$, respectively). Thus, after metrication, all W_n-weighted graphs are sparse. On originally (nearly) metric instances, $\mathsf{PB}_{\textbf{Top}}$ and $\mathsf{PB}_{\textbf{noM}}$ show similar performances; on dense W_n-weighted graphs with $\varrho \in R \cup \{100\%\}$, $\mathsf{PB}_{\textbf{Top}}$ is clearly superior. Table 1 also shows that $\mathsf{PB}_{\textbf{noE}}$ is far inferior to $\mathsf{PB}_{\textbf{Top}}$, a testament to the benefit of restricting K.

In the context of Table 1, the success rates for the other individually deactivated speedup techniques never deviate more than 5 ($\alpha \leq 2$) or 12 ($\alpha > 2$) percentage points from $\mathsf{PB}_{\textbf{Top}}$. Still, $\mathsf{PB}_{\textbf{Top}}$ achieves significant speedups against them, as shown in Table 2:

Fixing variables. As fixing mandatory variables generates no overhead, $\mathsf{PB}_{\textbf{noFix}}$ is never faster than $\mathsf{PB}_{\textbf{Top}}$. Overall, it typically allows 1.5–2 times faster computations, in particular on sparse graphs. The benefit decreases with increasing density or α, as the share of mandatory path variables drops; for $\alpha = 5$ almost no variables can be fixed.

Initial path variables. The speedup of the proposed initialization seems to largely depend on α. Clearly, the quality difference between the 1-spanner induced by the **1**-SP initialization to the α-spanner yielded by BG grows with α. Further, for larger α, **10**-SP is able to provide more paths. For $\alpha \leq 2$, there are some cases where the more elaborate initialization of $\mathsf{PB}_{\textbf{Top}}$ increases the overall running time, presumably due to the fact that there are only very few feasible paths. However, for the majority of the considered classes, the $\boldsymbol{k}$-SP+BG initialization is worthwhile and even yields close to three times faster computations than $\mathsf{PB}_{\textbf{simpleInit}}$ on dense W_{euc}-weighted WM graphs for $\alpha = 2$.

Pruning μ-CSP calls. On optimally solved instances with weights W_n, W_{euc}, and W_1, we can prune 90%, 86%, and 63% μ-CSP calls, respectively, in the median. This yields speedup factors of up to 3.1 compared to $\mathsf{PB}_{\textbf{noPrune}}$. The share of pruned calls slightly drops with increasing α. Surprisingly, this does not always translate into smaller speedups.

μ-BiA* pricing. Especially for $\alpha = 1.2$, the speedup against $\mathsf{PB}_{\textbf{noBiA}^*}$ is surprisingly low and sometimes even slightly below 1. Reasons may be that $\mathsf{PB}_{\textbf{1-BiA}^*}$ has a higher setup cost than BasicCSP, which cannot be recuperated if most necessary paths are already added

Table 3 Median running time (in sec) for WM (specified by δ or ϱ) and CMP graphs with $\alpha = 1.5$. "–" indicate success rates below 100%: 30% for $\mathsf{PB}_{\mathsf{Top}}$, and 0% for AB^+, respectively.

weight	W_n				W_{euc}		
density	$\delta = 4$	$\delta = 8$	$\varrho = 10\%$	CMP	$\delta = 4$	$\delta = 8$	$\varrho = 10\%$
n	1000 2000	1000 2000	1000 2000	500 1000	1000 2000	1000 2000	100 200
$\mathsf{PB}_{\mathsf{Top}}$	2.6 11.4	5.7 31.6	7.7 45.6	2.9 15.6	2.9 11.6	9.2 45.2	0.5 –
AB^+	35.9 172.9	672.0 –	1258.2 –	328.7 –	44.8 225.8	818.9 –	3.7 258.9

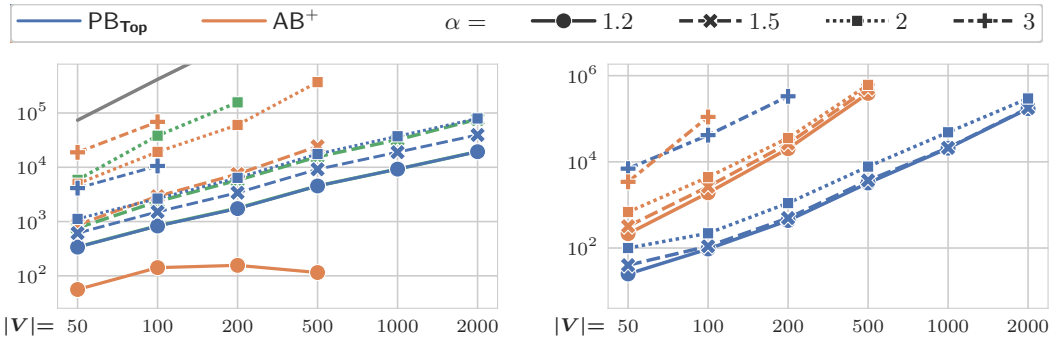
during initialization, and most μ-CSP calls are either pruned or quickly detected to be infeasible. Still, for larger stretches, using only BasicCSP instead of 3-BiA* can lead to up to 5-fold running times. – On most instances, the latter also slightly outperforms pricing via 1-BiA*, especially on instances that are sparse (after metrication) or for stretches $\alpha \leq 2$. $\mathsf{PB}_{\text{1-BiA}^*}$, however, seemingly has advantages for large (practically less relevant) stretches and (even after metrication) dense graphs. A possible explanation is based on the following observation on $\mathsf{PB}_{\mathsf{Top}}$: A majority of the dual solution values π (and thus edge costs in the CSP instance) are 0; thus in many cases the minimum cost feasible path is *free*, i.e., has total cost 0. On optimally solved instances, the median share of added free paths for W_n-, W_{euc}-, and W_1-weighted graphs are 95%, 98%, and 83%, respectively; their share grows with α and density. If there is a free path, there can be no other non-dominated solution for this μ-CSP call, and values $\mu > 1$ only inflict overhead.

The speedup of $\mathsf{PB}_{\mathsf{Top}}$ showcased against $\mathsf{PB}_{\mathsf{noM}}$, $\mathsf{PB}_{\mathsf{noE}}$, $\mathsf{PB}_{\mathsf{noFix}}$, $\mathsf{PB}_{\mathsf{simpleInit}}$, $\mathsf{PB}_{\mathsf{noPrune}}$, and $\mathsf{PB}_{\mathsf{noBiA}^*}$ give reason to not reject $\mathcal{H}2$. The decision between 1-BiA* or 3-BiA* is less clear: it seems non-crucial and instance-property dependent. We choose 3-BiA* in $\mathsf{PB}_{\mathsf{Top}}$, as it performs slightly better on more practically relevant instances. With no reason for rejection, we consider $\mathcal{H}2$ confirmed.

Hypothesis $\mathcal{H}3$. We compare $\mathsf{PB}_{\mathsf{Top}}$ and AB^+. Our experiments show a clear divide between (non-uniformly) weighted and unweighted (W_1) graphs, so we discuss them separately.

Weighted graphs. Tables 1 and 2 show the success rates and relative speeds on WM graphs (again, their behavior on ER graphs is analogous): $\mathsf{PB}_{\mathsf{Top}}$ can solve significantly more instances than AB^+ and achieves significant speedups, especially on graphs that are sparse (after metrication) and for stretches $\alpha \leq 2$. For W_{euc}, the advantage of $\mathsf{PB}_{\mathsf{Top}}$ seems to shrink with increasing α. For $\alpha = 5$, AB^+ is able to solve more instances than $\mathsf{PB}_{\mathsf{Top}}$ regardless of weight or density; however, neither algorithm can solve such instances with $n > 200$. Table 3 shows the median running times for different graph sizes and edge weights on WM and CMP graphs for $\alpha = 1.5$. Typically, $\mathsf{PB}_{\mathsf{Top}}$ is orders of magnitudes faster, and always requires less than 1 minute for any graph that is sparse (after metrication). AB^+ is only beneficial on dense ($\varrho = 10\%$) W_{euc}-weighted WM graphs; however, on those graphs, again neither algorithm can go beyond 200 nodes.

Fig. 1b shows the median running time of $\mathsf{PB}_{\mathsf{Top}}$ and AB^+ for W_{euc}-weighted ER graphs with $\varrho = 2\ln(n)/n$, for varying sizes and stretches. Running times of non-solved instances are interpreted as ∞; consequently, we only show data for success rates above 50%. For $\alpha = 5$, no algorithm can reliably solve graphs with more than 20 nodes. For all other α, $\mathsf{PB}_{\mathsf{Top}}$ solves significantly larger instances than AB^+. For $\alpha \leq 2$, $\mathsf{PB}_{\mathsf{Top}}$ reliably solves instances with 2000 nodes, while AB^+ can only solve instances with $n \leq 500$. For all $\alpha \leq 2$, median running times of $\mathsf{PB}_{\mathsf{Top}}$ are orders of magnitude lower than those of AB^+.



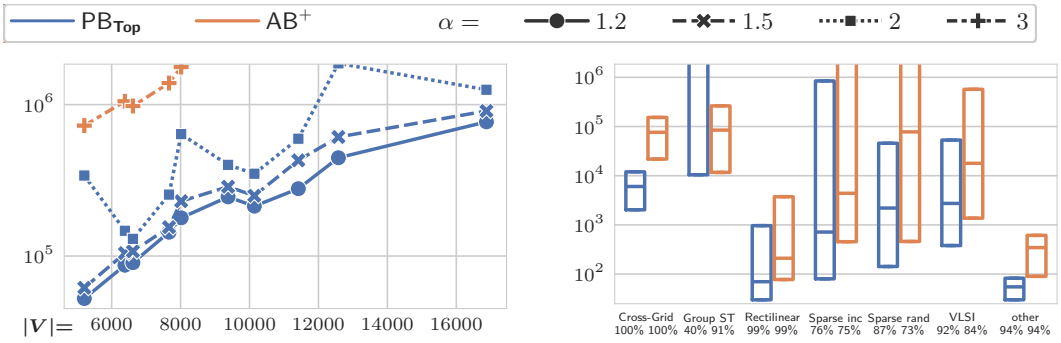
(a) Median number of non-fixed variables. **(b)** Median running time in ms.

Figure 1 Comparing PB$_{\text{Top}}$ and AB$^+$ on W$_{\text{euc}}$-weighted ER graphs with $\varrho = 2\ln(n)/n$. In **(a)**, the gray line represents the number of variables in the model (AB) (without fixing); the green lines (dependent on α) give the upper bound PB-UB of non-fixed variables for PB$_{\text{Top}}$, if one would consider all path variables in the full ILP (PB). For $\alpha = 1.2$, PB$_{\text{Top}}$ attains this upper bound.

Fig. 2a shows the running times for the real-world W$_{\text{t}}$-weighted Road instances; results for W$_{\text{len}}$ are similar. For $\alpha \leq 2$, PB$_{\text{Top}}$ solves all instances optimally, even the one with 16 874 nodes, and is orders of magnitudes faster than AB$^+$. In contrast to PB$_{\text{Top}}$, AB$^+$ – which has similar running times for all $\alpha \leq 3$ – is able to solve some small instances for $\alpha = 3$. Neither solved any Road graph for $\alpha = 5$. Consider the established SteinLib, see Fig. 2b for $\alpha = 1.5$. Generally, most groups behave similarly to sparse W$_{\text{n}}$- or W$_{\text{euc}}$-weighted graphs. Except for *Group ST* instances, PB$_{\text{Top}}$ performs better than AB$^+$ and never solves fewer instances. Interestingly, the *Cross-Grid* graphs are unweighted but – in contrast to most unweighted graphs, discussed below – PB$_{\text{Top}}$ solves them for $\alpha = 1.5$ over 10 time *faster* than AB$^+$.

Unweighted graphs. As witnessed for WM in Tables 1 and 3 (and analogously for ER), unweighted instances are generally much harder for both AB$^+$ and PB$_{\text{Top}}$, and we can solve only fewer and smaller graphs. On those graphs, however, AB$^+$ typically significantly outperforms PB$_{\text{Top}}$. PB$_{\text{Top}}$ cannot solve any generated instance with $n \geq 100$ for $\varrho \in R \cup \{100\%\}$. On unweighted instances, α needs to be integer, but only few instances were solved by either algorithm for $\alpha \geq 3$. So we concentrate on $\alpha = 2$ in the following: Let PB-UB denote the number of variables in the full (PB)-model, but after fixing variables. This yields an upper bound on PB$_{\text{Top}}$'s number of variables (which would, e.g., also be attained by the brute-force initialization discussed in Section 3). We observe that PB$_{\text{Top}}$ generates close to PB-UB many variables, whereas AB$^+$, thanks to fixing unreachable and mandatory variables, requires only comparably few. The former seems to struggle identifying the best of the large set of paths with identical length. Furthermore, we observe that SCIP's default cut generators successfully separate several additional constraints to the (AB)-model, whereas they find none for the (PB)-model. This leads to the effect that, for all densities except $\varrho \in \{50\%, 100\%\}$, AB$^+$ solves the majority of instances at the root B&B node. PB$_{\text{Top}}$ requires a median number of 36 000 B&B nodes already for $\delta = 8$ and $n = 50$.

Overall, we have to reject hypothesis $\mathcal{H}3$ on unweighted instances, very dense W$_{\text{euc}}$-weighted graphs, and large (arguably less practically relevant) stretches $\alpha \geq 5$. For all other instances, we cannot reject $\mathcal{H}3$, as PB$_{\text{Top}}$ is almost always significantly faster and able to solve larger instances than AB$^+$. The hypothesis looks particularly strong for stretches $\alpha \leq 2$.



(a) Road instances with W_t (i.e., travel times). AB$^+$ is visually indentical for all $\alpha \in \{1.2, 1.5, 2, 3\}$.

(b) SteinLib, grouped by type, for $\alpha = 1.5$.

Figure 2 Running time (in ms; always on the vertical log-scale axis) of PB$_{\text{Top}}$ and AB$^+$. The legend of Fig. 1 applies. On the horizontal axis of (b), we give the corresponding success rates. Observe the time limit of $1.8 \cdot 10^6$ms.

Hypothesis $\mathcal{H}3'$. The rationale for this hypothesis is that in the aforementioned cases where cut-based ILPs dominate arc-based MCF formulations, the former typically yield practically smaller models. Thus, it seems natural that PB$_{\text{Top}}$'s strength could be due to a smaller set of variables, compared to the $\mathcal{O}(n^4)$ of (AB). However, the picture seems much more complicated here: As mentioned above, on unweighted graphs (with $\alpha = 2$ as the only statistically significant case) the LPs of AB$^+$ hold fewer variables than PB$_{\text{Top}}$. Consider (non-uniformly) weighted graphs: As a representative example, Fig. 1a shows the median number of non-fixed variables per algorithm, whenever all respective W_{euc}-weighted ER instances with $\varrho = 2\ln(n)/n$ are solved; similar results hold for the other non-uniformly weighted generated instances. For both algorithms, this number grows with α. However, for AB$^+$ and $\alpha \leq 2$, these differences in model size are *not* reflected in the eventual running times, see Fig. 1b. In fact, for $\alpha = 1.2$, AB$^+$ yields the smallest median model size, but its median running time is still orders of magnitudes larger than PB$_{\text{Top}}$'s. AB$^+$ spends a significant amount of time fixing variables, which dominates the running time for $\alpha < 2$. However, only for $\alpha = 1.2$, it yields fewer unfixed variables than PB$_{\text{Top}}$; otherwise AB$^+$ typically requires significantly more variables. Sometimes, it even yields models larger than PB-UB. Consistently, on the very sparse Road instances [14, Fig. A1], PB$_{\text{Top}}$'s variable number is significantly larger than AB$^+$ (and surprisingly independent of α for all $\alpha \leq 2$). Also on SteinLib, AB$^+$ typically requires less variables than PB$_{\text{Top}}$. In any case, AB$^+$'s strive for a small variable set is typically too expensive to attain competitive running times – in particular since the reduction is very successful only for small α, where PB$_{\text{Top}}$ is still faster.

Overall, while there are several cases where PB$_{\text{Top}}$ requires less variables than AB$^+$, we overall reject $\mathcal{H}3'$ in its generality. Certainly, it cannot be used as the hypothesized simple explanation for PB$_{\text{Top}}$'s strong practical performance. Its performance benefit over AB$^+$ seems to be rooted in the latter's drawback: (AB)'s model size, and the very time-consuming preprocessing required to mitigate its effect (if successful at all).

Hypothesis $\mathcal{H}4$. The lower part of Table 1 lists the median gap (%) of BG to the optimum solution (or best lower bound), for WM graphs with varying $n \in N$. Results are more significant for high success rates. Generally, median gaps grow with α and are significantly larger on unweighted graphs. In [49], they observed decreasing gaps for increasing $n \leq 64$. Interestingly, our experiments on significantly larger graphs show that both the median gap and even the corresponding interquartile range decreases with increasing n [14, Table A2].

For $\alpha \in \{1.2, 1.5, 2\}$, the gaps on *Airlines* [51] are 0.8%, 5.2%, and 5.8%. On both W_t- and W_{len}-weighted Road graphs, the median gaps are 0.0%, 0.1%, and 0.5%, respectively [14, Table A3]. On all SteinLib groups in Fig. 2b, except for *other*, BG yields a median gap of 0.0% for $\alpha \in \{1.2, 1.5\}$. For $\alpha = 2$, the gap is below 0.9% on all groups except *VLSI* (15.7%) and unweighted *Cross-Grid* (84.8%). Overall, we consider $\mathcal{H}4$ confirmed for weighted graphs.

Summary. We improved both previously known exact MWSP algorithms; in particular, our speedup techniques for the path-based column generation approach enable us to solve instances orders of magnitude larger than previous attempts. On most instances, this approach is superior to the arc-based model. Exceptions are instances which are structurally challenging for both approaches (but, according to literature, less practically relevant): unweighted graphs and instances with very large stretch factors $\alpha \geq 5$. Lastly, we used the newly found lower bounds to evaluate the quality of the strongest and most prominent *basic greedy* heuristic: Even for large instances, it produces near-optimal spanners, with the quality surprisingly improving with instance size. Unsurprisingly, it is also challenged by unweighted instances, as its strategy to consider edges in order of increasing weight is ineffective.

In the future, one may try to further improve the μ-BiA* algorithm by exploiting the many 0-cost edges, or by identifying further strengthening constraint classes.

References

1 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37, 2020. doi:10.1016/j.cosrev.2020.100253.

2 Reyan Ahmed, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, Faryad Darabi Sahneh, and Richard Spence. Approximation algorithms and an integer program for multi-level graph spanners. In *Analysis of Experimental Algorithms: Special Event, SEA^2 2019, Kalamata, Greece, June 24-29, 2019*, pages 541–562. Springer, 2019. doi:10.1007/978-3-030-34029-2_35.

3 Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: Theory, applications and algorithms*. Englewood Cliffs, New Jersey, USA Arrow, KJ: Prentice-Hall, 1993.

4 Stephen Alstrup, Søren Dahlgaard, Arnold Filtser, Morten Stöckel, and Christian Wulff-Nilsen. Constructing light spanners deterministically in near-linear time. *Theoretical Computer Science*, 907:82–112, 2022. doi:10.1016/j.tcs.2022.01.021.

5 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.

6 Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(3):356–371, 2012. doi:10.1287/ijoc.1110.0456.

7 Cynthia Barnhart, Natashia L Boland, Lloyd W Clarke, Ellis L Johnson, George L Nemhauser, and Rajesh G Shenoi. Flight string models for aircraft fleeting and routing. *Transportation science*, 32(3):208–220, 1998. doi:10.1287/trsc.32.3.208.

8 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007. doi:10.1002/rsa.20130.

9 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed steiner forest. *Information and Computation*, 222:93–107, 2013. doi:10.1016/j.ic.2012.10.007.

10 Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. Enabling research through the SCIP optimization suite 8.0. *ACM Trans. Math. Softw.*, 49(2), 2023. doi:10.1145/3585516.

11 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

12 Geoff Boeing. Street network models and measures for every u.s. city, county, urbanized area, census tract, and zillow-defined neighborhood. *Urban Science*, 3(1), 2019. doi:10.3390/urbansci3010028.

13 Fritz Bökler, Matthias Ehrgott, Christopher Morris, and Petra Mutzel. Output-sensitive complexity of multiobjective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):25–36, 2017.

14 Fritz Bökler, Markus Chimani, Henning Jasper, and Mirko H. Wagner. Exact minimum weight spanners via column generation. *CoRR*, 2024. arXiv-Version including an appendix. arXiv:2406.19164.

15 Fritz Bökler, Markus Chimani, Henning Jasper, and Mirko H. Wagner. Experimental data. https://tcs.uos.de/research/spanner, 2024.

16 Leizhen Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics*, 48(2):187–194, 1994. doi:10.1016/0166-218X(94)90073-6.

17 Leizhen Cai and Mark Keil. Spanners in graphs of bounded degree. *Networks*, 24(4):233–249, 1994. doi:10.1002/net.3230240406.

18 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proc. SoCG 1992*, pages 192–201. ACM, 1992. doi:10.1145/142675.142717.

19 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, and Petra Mutzel. The Open Graph Drawing Framework (OGDF). In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 17. CRC press, 2014.

20 Markus Chimani, Maria Kandyba, Ivana Ljubić, and Petra Mutzel. Orientation-based models for {0,1,2}-survivable network design: Theory and practice. *Math. Program.*, 124(1):413–439, 2010. doi:10.1007/s10107-010-0375-5.

21 Markus Chimani and Finn Stutzenstein. Spanner approximations in practice. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *Algorithms–ESA 2022*, volume 244 of *LIPIcs*, pages 37:1–37:15, 2022. doi:10.4230/LIPIcs.ESA.2022.37.

22 Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming models*. Springer, 2014.

23 IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009. URL: https://www.ibm.com/analytics/cplex-optimizer.

24 Martin Desrochers and François Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR: Information Systems and Operational Research*, 26(3):191–212, 1988. doi:10.1080/03155986.1988.11732063.

25 Jürgen Ebert. A versatile data structure for edge-oriented graph algorithms. *Communications of the ACM*, 30(6):513–519, 1987. doi:10.1145/214762.214769.

26 Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. In *SODA 2017*, pages 652–669. ACM SIAM, 2017. doi:10.1137/1.9781611974782.41.

27 Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. *SIAM Journal on Discrete Mathematics*, 29(3):1312–1321, 2015. doi:10.1137/140979538.

28 Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(3):1–21, 2016. doi:10.1145/2836167.

29 Paola Festa. Constrained shortest path problems: state-of-the-art and recent advances. In *International Conference on Transparent Optical Networks (ICTON)*, pages 1–17. IEEE, 2015. doi:10.1109/ICTON.2015.7193456.

30 Matteo Fischetti. Facets of two Steiner arborescence polyhedra. *Math. Program.*, 51(1):401–419, 1991. doi:10.1007/BF01586946.

31 Matteo Fischetti, Juan-José Salazar-Gonzalez, and Paolo Toth. The Generalized Traveling Salesman and Orienteering Problems. In Gregory Gutin and Abraham P. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, pages 609–662. Springer US, 2007. doi:10.1007/0-306-48213-4_13.

32 Michael Robert Garey and David Stifler Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, 1979.

33 Irene Heinrich, Olli Herrala, Philine Schiewe, and Topias Terho. Using Light Spanning Graphs for Passenger Assignment in Public Transport. In Daniele Frigioni and Philine Schiewe, editors, *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*, volume 115 of *Open Access Series in Informatics (OASIcs)*, pages 2:1–2:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2023.2.

34 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

35 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42(2):700–731, 2013. doi:10.1109/FOCS.2011.13.

36 Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theoretical Computer Science*, 746:88–97, 2018. doi:10.1016/j.tcs.2018.06.031.

37 Thorsten Koch, Alexander Martin, and Stefan Voß. Steinlib: An updated library on Steiner tree problems in graphs. Tech Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000. URL: http://steinlib.zib.de/steinlib.php.

38 Gang Liu and KG Ramakrishnan. A* prune: an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, volume 2, pages 743–749. IEEE, 2001.

39 Ivana Ljubic, René Weiskircher, Ulrich Pferschy, Gunnar W. Klau, Petra Mutzel, and Matteo Fischetti. Solving the prize-collecting steiner tree problem to optimality. In *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments and the Second Workshop on Analytic Algorithmics and Combinatorics, Vancouver, BC, Canada*, pages 68–76. SIAM, 2005.

40 Ernesto Queiros Vieira Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

41 Aristide Mingozzi, Marco A Boschetti, Salvatore Ricciardelli, and Lucio Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6):873–888, 1999. doi:10.1287/opre.47.6.873.

42 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

43 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. In *Proc. PODC 1987*, pages 77–85. ACM, 1987. doi:10.1145/41840.41847.

44 Mathew Penrose. *Random geometric graphs*, volume 5. OUP Oxford, 2003.

45 Luigi Di Puglia Pugliese and Francesca Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013. doi:10.1002/net.21511.

46 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61:389–401, 2011. doi:10.1007/s00453-010-9401-5.

47 Khodakaram Salimifard and Sara Bigharaz. The multicommodity network flow problem: state of the art classification, applications, and solution methods. *Operational Research*, 22(2):1–47, 2022. doi:10.1007/s12351-020-00564-8.

48 Hanan Shpungin and Michael Segal. Near-optimal multicriteria spanner constructions in wireless ad hoc networks. *IEEE/ACM Transactions on Networking*, 18(6):1963–1976, 2010. doi:10.1109/INFCOM.2009.5061918.

49 Mikkel Sigurd and Martin Zachariasen. Construction of minimum-weight spanners. In *Algorithms–ESA 2004*, pages 797–808. Springer, 2004. doi:10.1007/978-3-540-30140-0_70.

50 Barrett W Thomas, Tobia Calogiuri, and Mike Hewitt. An exact bidirectional A* approach for solving resource-constrained shortest path problems. *Networks*, 73(2):187–205, 2019. doi:10.1002/net.21856.

51 Markus Wallinger, Daniel Archambault, David Auber, Martin Nöllenburg, and Jaakko Peltonen. Faster edge-path bundling through graph spanners. *Comput. Graph. Forum*, 42(6), 2023. doi:10.1111/CGF.14789.

52 Bernard M. Waxman. Routing of multipoint connections. *IEEE J. Sel. Areas Commun.*, 6(9):1617–1622, 1988. doi:10.1109/49.12889.

53 Laurence A Wolsey. *Integer Programming*. John Wiley & Sons, 2020.

54 Xiaoyan Zhu and Wilbert E Wilhelm. A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation. *Computers & Operations Research*, 39(2):164–178, 2012. doi:10.1016/j.cor.2011.03.008.