Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 86 (2025) 290–297

Transportation Research Procedia

www.elsevier.com/locate/procedia

26th Euro Working Group on Transportation Meeting (EWGT 2024)

Non-pool-based line planning: a practical analysis of the FPT algorithm TW-LINES

Oliver Bachtler[a,*], Irene Heinrich[b], Philine Schiewe[c], Constantin Seebach[a]

[a]*RPTU Kaiserslautern-Landau, Postfach 3049, 67653 Kaiserslautern, Germany*
[b]*TU Darmstadt, Präsidium der Technischen Universität Darmstadt, 64277 Darmstadt, Germany*
[c]*Aalto University, P.O. Box 11000 (Otakaari 1B), FI-00076 Aalto, Finland*

Abstract

Line planning, which is the selection of lines to be operated to meet the demands of a transport network, is a crucial component in public transport optimization. Our line planning algorithm TW-LINES for complete line pools (see Heinrich et al. (2023b)) is proven to be fast in theory whenever treewidth, maximum degree, and maximum upper frequency of the input network are bounded. So far, experiments and a detailed analysis of the algorithm's practical performance were lacking. In this paper we close this gap. For two natural families of line planning instances we show that TW-LINES finds (close-to) optimal solutions quickly and markedly reduces the line concept costs compared to traditional line-pool-based solution approaches.

Keywords: line planning; fixed parameter tractability; treewidth

1. Introduction

Public transport optimization (see Guihaire and Hao (2008)) is gaining ever more importance in light of the sustainable development goals formulated in UN General Assembly (2015). The crucial step of line planning is the selection of simple paths (*lines*) in a public transport network (*PTN*) together with the frequencies at which the lines are operated. Since there are exponentially many possible lines in even highly structured and thin PTNs (e.g., a square grid of dimension d inherits more than 2^d possible lines), line planning is traditionally executed in two phases: first, the potential lines within the public transport network are significantly reduced to a manageable *line pool*, typically via manual or heuristic methods, see Gattermann et al. (2017). Then a (near-)optimal solution is determined, employing only the lines from this predefined pool, see Schöbel (2012) and Schmidt and Schöbel (2024). With this approach, however, there is a significant risk of dismissing the optimal solution in the first phase. One way of overcoming this

[*] Corresponding author. Tel.: +49-631-205-3878.
E-mail address: o.bachtler@math.rptu.de

10.1016/j.trpro.2025.04.037

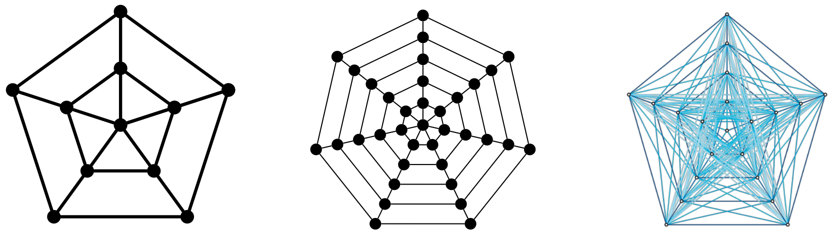

Fig. 1: Circular cities with $r = 2$ and $s = 5$ (left), $r = 5$ and $s = 7$ (middle), and $r = 4$, $s = 5$ and demand data (right). Vertices represent stops and edges represent streets or tracks.

dilemma is the column-generation approach of Borndörfer et al. (2007). However, considering all possible lines yields an intractable problem, see Heinrich et al. (2022). We developed the fixed parameter tractable algorithm TW-LINES for finding cost-optimal line concepts in Heinrich et al. (2023b), involving the three parameters maximum degree, maximum upper frequency bound, and treewidth of the given network. Particularly charming about this approach is that real-work PTNs inherently come with a bounded maximum degree, i.e., a maximum number of streets or tracks meeting at an intersection, and a maximum upper frequency imposed by the local infrastructure. treewidth (see Bodlaender (1998)) is another natural parameter in this context since cities which developed along main roads, rivers, or valleys often can be described as networks of small treewidth. Also circular cities (see Fig. 1 and Stokkink et al. (2023)) which incorporate multiple properties of real word PTNs (planarity, having a city center around which the city expands in all directions) are of bounded treewidth whenever the center vertex is of bounded degree, which is imposed by the above mentioned natural bound on the maximum number of intersections at a crossing.

2. Preliminaries and Problem Description

For two functions f and f' on the same domain D we write $f \leq f'$ to indicate that $f(x) \leq f'(x)$ for each element x of D. Analogously, we use "$\geq$" and "$=$" for functions to indicate that these relations apply point-wise.

Graph theory. Whenever we consider a graph $G = (V, E)$, we use $n := |V|$ to denote its number of vertices. A graph $P = (V, E)$ of the form $V = \{v_1, v_2, \dots, v_m\}$ and $E = \{v_1v_2, v_2v_3 \dots, v_{m-1}v_m\}$ where all the v_i are distinct is a v_1-v_m-*path* (or just *path*). If a graph is obtained from a v_1-v_m path by adding the edge v_1v_m, then it is called a *cycle*. We write $E(v)$ for the edges incident to a vertex $v \in V$ and the *degree* $\deg(v)$ of a vertex v is $|E(v)|$.

Circular cities. For two positive numbers r and s the *circular city* $C_{r,s}$ consists of r *rings* and s *spokes* around a center vertex. Formally, $C_{r,s}$ is the union of r cycles $C_i = (v_1^i, \dots, v_s^i, v_1^i)$ for $1 \leq i \leq r$ and s paths $P_j = (c, v_j^1, \dots v_j^r)$ for $1 \leq j \leq s$. We note that the treewidth of $C_{r,s}$ is at most $\min\{2r + 1, s\}$. We also compute a lower and an upper bound on the number of paths in a circular city:

Lemma 2.1. For every two positive numbers r and s the circular city $C_{r,s}$ contains at least $(2s - 1)^r$ distinct paths.

Proof. Fix two positive numbers r and s. Let $p(r)$ be the number of paths of the form $(c, v_{j_1}^{i_1}, v_{j_2}^{i_2}, \dots, v_{j_l}^{i_l})$ which satisfy $i_l = r$ and $i_k \leq i_{k+1}$ for each $i \in [l - 1]$ (i.e., paths from the center to the outer ring where the distance to the center increases monotonically when traversing the path). We have that $p(1) = s(2s - 1)$ and $p(r) = p(r - 1) \cdot (2s - 1)$, yielding $p(r) = (2s - 1)^r s$. □

Line planning. A *public transport network (PTN)* is a graph $G = (V, E)$ whose vertices and edges represent stations and direct connections between them (e.g., streets or tracks), respectively. A *line planning instance* is a tuple $(G, c_{\text{fix}}, c, f^{\min}, f^{\max})$ where G is a PTN, $c_{\text{fix}} \in \mathbb{R}_{\geq 0}$ represents *frequency-dependent fixed costs*, $c\colon E \to \mathbb{R}_{\geq 0}$, $e \mapsto c_e$ describes the *edge-dependent costs*, and $f^{\min}\colon E \to \mathbb{N}$ and $f^{\max}\colon E \to \mathbb{N}$ are *integer frequency restrictions* on the edges, satisfying $f^{\min} \leq f^{\max}$.

A *line* ℓ is a path in G and a *line concept* $(\mathcal{L}, f)$ is a set of lines $\mathcal{L}$ with a *frequency vector* $f = (f_\ell)_{\ell \in \mathcal{L}} \in \mathbb{N}^{|\mathcal{L}|}$, that is, f_ℓ is the *frequency* of line ℓ. At each edge $e \in E$, the lines sum up to a total frequency $F_e^{(\mathcal{L},f)} = \sum_{\ell \in \mathcal{L}:\, e \in E(\ell)} f_\ell$. A line concept is *feasible* if for each edge $e \in E$ the frequency restrictions are satisfied, i.e., if $f_e^{\min} \leq F_e^{(\mathcal{L},f)} \leq f_e^{\max}$.

We use frequency-dependent *line costs* $\text{cost}_\ell = c_{\text{fix}} + \sum_{e \in E(\ell)} c_e$ which consist of fixed costs c_{fix} and edge-dependent costs c_e, $e \in E$. The *costs of a line concept* $(\mathcal{L}, f)$ are defined as $\text{cost}(\mathcal{L}, f) = \sum_{\ell \in \mathcal{L}} \text{cost}_\ell \cdot f_\ell$. An equivalent representation is $\sum_{e \in E} c_e F_e^{(\mathcal{L},f)} + c_{\text{fix}} \cdot \sum_{\ell \in \mathcal{L}} f_\ell$ and we call $\sum_{e \in E} c_e F_e^{(\mathcal{L},f)}$ the *edge-cost-part* of the line concept and $c_{\text{fix}} \cdot \sum_{\ell \in \mathcal{L}} f_\ell$ the *fixed-cost-part* of the line concept.

With this notation, we can define the line planning on all lines problem considered in this paper.

Definition 2.2. Given a line planning instance, the *line planning on all lines problem* (LPAL) seeks to find a feasible line concept of minimal cost.

We obtain the following combinatorial bounds on the cost of an optimal line concept.

Lemma 2.3. If $(G, c_{\text{fix}}, c, f^{\min}, f^{\max})$ is a line planning instance, $(\mathcal{L}, f)$ is a feasible line concept, and $(\mathcal{L}^*, f^*)$ is a cost-optimal line concept, then

$$\text{cost}(\mathcal{L}, f) \geq \sum_{e \in E} c_e f_e^{\min} + c_{\text{fix}} \cdot \max_{v \in V} \tfrac{1}{2} \sum_{e \in E(v)} f_e^{\min} =: \text{LB and}$$

$$\text{cost}(\mathcal{L}^*, f^*) \leq \sum_{e \in E} c_e f_e^{\min} + c_{\text{fix}} \cdot \sum_{e \in E} f_e^{\min} =: \text{UB}.$$

Proof. For the lower bound, note that $F_e^{(\mathcal{L},f)} \geq f_e^{\min}$. Observe that an edge incident to a vertex v can only be used by a line that contains v and, hence, every line uses at most two edges of $E(v)$. As a result, to cover all edges incident to v, we need at least half as many lines as their combined minimum frequencies. It follows that $\max_{v \in V} \frac{1}{2} \sum_{e \in E(v)} f_e^{\min}$ is a lower bound on the number of lines in $\mathcal{L}$.

For the upper bound, notice that this value is obtained by using individual edges as lines and setting the frequency of line ℓ containing only e to $f_e^{\min}$, making this value at least as good as the optimum. □

3. Experiment description

We evaluate the performance of TW-LINES, as presented in Heinrich et al. (2023b). TW-LINES uses a nice tree decomposition (cf. Bodlaender (1998)) of a PTN together with a restricted set of operations to construct all possible lines of the PTN. These operations are used while traversing the tree decomposition in a bottom-up fashion. The optimal choices of which operations should be applied are determined by a linear integer program (IP), which can be solved in polynomial time, assuming that the maximum degree, the maximum upper frequency, and the treewidth of the PTN are constant.

In our experiment, TW-LINES is executed on a server with an Intel Xeon Gold 6126 CPU and the IP is solved using Gurobi 11.0.0 (Gurobi Optimization, LLC (2023)). Note that while TW-LINES is a fixed parameter tractable algorithm, solving the IPs with Gurobi in our practical implementation does not guarantee this property.

To evaluate the benefit of using the complete line pool, we compare TW-LINES with two heuristics implemented in the open source software library LinTim, see Sections 3.2.1 and 3.2.3 in Schiewe et al. (2023). Both use cost-optimal line planning and heuristically generate a line pool. Therefore, in contrast to TW-LINES, they do not have access to all possible lines. The first line pool generation heuristic uses a tree-based approach of Gattermann et al. (2017) that generates minimum spanning trees and uses these to generate lines (between leaves or vertices of high degree). The second heuristic adds the k shortest paths between each OD pair, where we use $k = 1$ and $k = 2$, and deletes nested lines. The line pools generated this way are small compared to the complete line pool as is illustrated in Table 1, which contains the values for our test instances. Note that the entries in the last row are obtained by the lower bound of Lemma 2.1.

The computation time of TW-LINES is split into two stages: constructing an IP and then solving the IP, which amounts to the majority of the computation time. We measure the running times of these stages separately. For the IP solving

Algorithm	min	max	median
tree-based	31	3026	724.0
1-SP	12	555	181.5
2-SP	21	978	296.0
TW-LINES	75	≈2.8e21	≈5.5e10

Table 1: Line pool sizes of TW-LINES and the three heuristics on circular cities. The values for TW-LINES are the lower bounds described in Lemma 2.1.

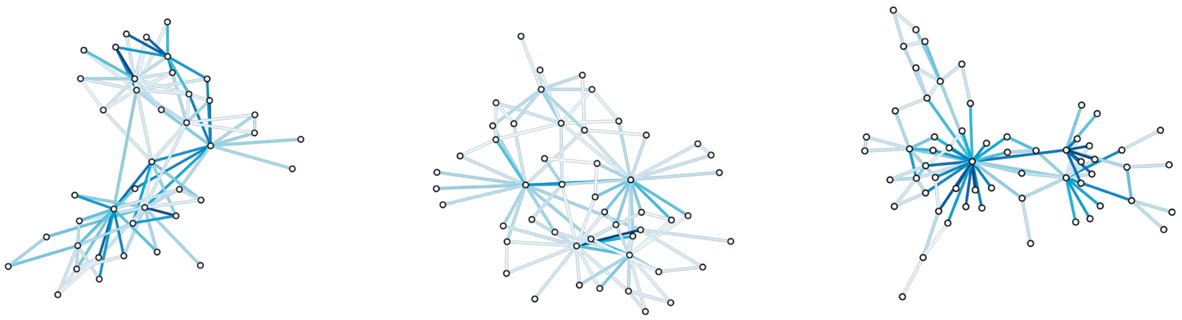
Fig. 2: Examples of our randomly generated low-treewidth graphs, with darker edges having higher values of $f_e^{\min}$. These have 41 vertices and treewidth 4 (left); 50 vertices and treewidth 3 (middle); 60 vertices and treewidth 2 (right).

we impose a runtime limit of one hour. We also evaluate the performance of TW-LINES when stopping the IP solver upon reaching an integrality gap of 2 % or less. In this case, TW-LINES is still able to produce a feasible line concept, albeit a sub-optimal one. For the heuristics we neither measure runtimes, nor impose time limits.

We evaluate the algorithms on several procedurally generated line planning instances $(G, c_{\text{fix}}, c, f^{\min}, f^{\max})$. There are three major steps in generating instances: First, choose a graph G, second, create a load on the edges $(f^{\min})$, and, third, apply a cost model (c and c_{fix}). All considered instances have $f^{\max} = 20$.

As benchmark instances we generate the two families of instances described in the following two paragraphs.

Circular cities. These instances are of the form $(C_{r,s}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ for all $r \geq 2$ and $3 \leq s \leq 5$ with $rs \leq 90$. In particular, in each of the instances the treewidth of $C_{r,s}$ is at most 5. We generate the passenger demand using the formula from Section 6 of Heinrich et al. (2023a), setting $d_r = d_s = 1$ (see Fig. 1 for an example). From the resulting demand matrix we compute the minimal frequency bounds $f^{\min}$ by routing passengers along shortest paths to their destinations. tune the demand matrix by a factor to ensure that $f^{\min} \leq 5$. We determine the edge costs by scaling the edge lengths which are induced by a natural embedding of $C_{r,s}$ into the real plane (for more details, see Heinrich et al. (2023a)). For each circular city we consider the three options $c_{\text{fix}} = 10$, $c_{\text{fix}} = 25$, and $c_{\text{fix}} = 50$.

Random graphs of low treewidth. For $k \in \{2, 3, 4\}$ we generate random order-n graphs (V, E) of treewidth at most k. First, we generate a random tree T: we start with a single node (the *root*) and iteratively add a child to a random existing node. We repeat this until T has $n - k$ nodes. Then we convert T into a tree decomposition by associating a *bag* B_u with each node $u \in T$; the root has the bag $\{1, \dots, k + 1\}$ and all other bags inherit their parents' bag, but randomly *forget* an existing vertex and *introduce* a new unique vertex from the vertex set $V = \{1, \dots, n\}$. We pick each $e \in \bigcup_{u \in T} \binom{B_u}{2}$ to be in E independently with probability $\frac{1}{2}$. If $G = (V, E)$ is not connected, then we randomly add more edges from $\bigcup_{u \in T} \binom{B_u}{2}$ to E. Finally, we embed (V, E) into the plane using a force-directed layout algorithm. The demands are computed according to the *gravity model* of Rodrigue et al. (2016) (Chapter 10, Section 9), assuming equal weight of all vertices. In the same fashion as for the previous family, we compute $f^{\min}$ from the demand matrix. The edge costs c are length-proportional and we set $c_{\text{fix}} := \frac{10}{|E|} \sum_{e \in E} c_e$. See Fig. 2 for examples.

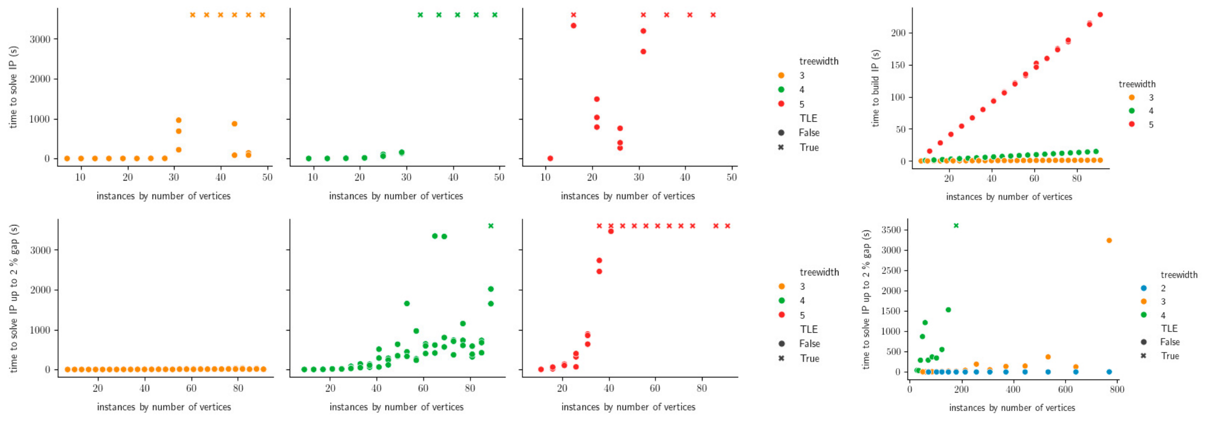

Fig. 3: The runtime of TW-LINES on different instances. Those instances, where TW-LINES did exceeded the time limit (TLE) are marked by an x. The plots depict the IP-time on circular cities (top left), the build time on circular cities (top right), the IP-time up to 2 % gap on circular cities (bottom left), and the IP-time on low-treewidth graphs (bottom right).

4. Results

Running time. First, we analyze the running time of TW-LINES in order to show its applicability. We measure the building time and the solving time of the IP separately. The building and solving time of the IPs for circular cities can be seen at the top of Fig. 3, respectively. Note that the building time seen in the top right Fig. 3 exhibits the expected linear behavior when we fix the treewidth. The top left details the time it takes to solve the IPs, which dominates the runtime of TW-LINES, where we sort the instances by their number of spokes (which equals the treewidth) and rings, instead of their vertex count. Not all instances can be solved to optimality within the time limit of an hour and the graphic shows that the treewidth has a higher influence on the runtime than the number of vertices as can be expected from the theoretical analysis of TW-LINES in Heinrich et al. (2023b). Note that good solutions are obtained fairly quickly and most of the solver time is spent on proving optimality. Among the instances that are not solved within the time limit, the maximum integrality gap is less than 10 % while the median gap is below 1 %.

The runtimes when we accept a gap of at most 2 % instead of trying to solve the problem to optimality are shown in the bottom left of Fig. 3. We directly see that the runtimes are significantly shorter now and, since we ran the experiments in parallel for the same amount of time, this figure has more data points. Note that when stopping the solver at 2 % gap, the maximum gap to an optimal solution is, indeed, almost 2 %. However, over all instances where we obtained an optimal solution, the median gap is just 0.7 % and over a third of the instances were solved optimally.

We also see that all circular city instances of treewidth 3 only require a marginal amount of time to solve to an optimality gap of 2 %. This behavior can also be observed for the random instances of small treewidth, where we expect our algorithm to truly shine: the results (see the bottom right of Fig. 3) show that treewidth 3 still solves quickly with over 600 vertices and the instance with 770 still solves in less than an hour. On the other hand, the algorithms dependence on the treewidth is clearly visible: for treewidth 2 the instances still all solve in less than four seconds while, for treewidth 4, the instance with 179 vertices reaches the runtime limit of an hour. In summary, for small treewidth values, the algorithm can optimally solve very large instances.

Solution quality. We now compare TW-LINES to the three described line pool generation heuristics in terms of solution quality. Note that, since TW-LINES returns solutions that are (almost) optimal, this comparison can also be interpreted as a quality assessment for the heuristics.

We start by looking at the results of our experiments on circular cities. For this we compare the ratios between the cost of the solution found by TW-LINES and the cost of heuristic solutions on different instances. The results can be seen in Fig. 4. We remark that the solution of TW-LINES we used is only the one with the 2 % gap if we did not manage to compute the optimal solution, and that the shortest path heuristics are missing entries since their execution failed for larger instances.

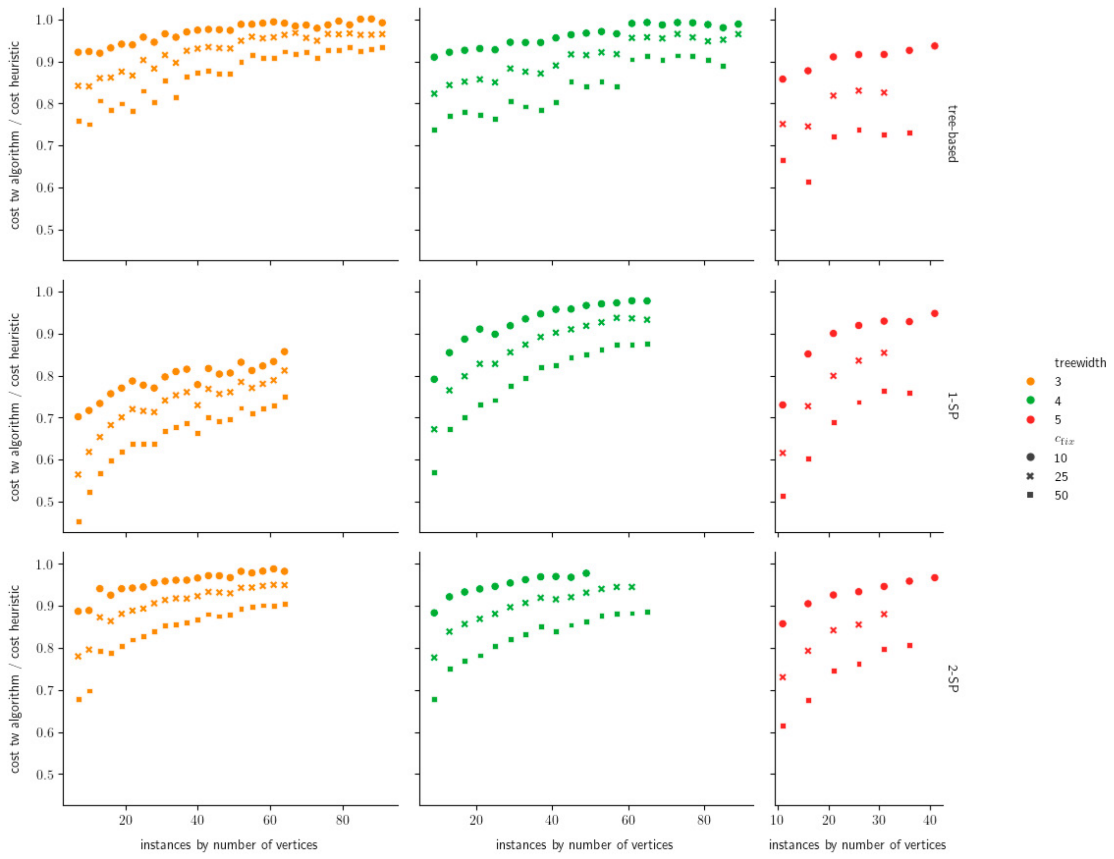

Fig. 4: The ratio between the cost of the solution obtained by TW-LINES and the heuristic solution on circular cities.

We see that it is possible to achieve relevant cost reductions using TW-LINES. To be precise, the ratio ranges from 0.454 to 1.001, where the values greater than 1 stem from the fact that we stop at a 2 % gap for some instances. The median value is 0.883.

Moreover, we observe that the heuristics perform worse for higher c_{fix}. This suggests that they use an excessive number of lines compared to the optimal solutions obtained by TW-LINES. To analyze this in more detail, we separate the solution cost into the fixed-cost-part and the edge-cost-part, then compute reduction ratios of these two parts individually (see the left of Fig. 5). This shows that the main improvement of TW-LINES comes from the fixed-cost-part, confirming that it manages to use fewer lines while still achieving a feasible solution. Hence the total cost reduction is better for higher values of c_{fix}. Thus, on circular cities TW-LINES's access to the full line pool allows it to use more efficient lines that are not part of the heuristics' line pools to achieve better solutions.

We also observe that the heuristics get better for larger circular cities due to the structure of our instances. For a fixed number of spokes and i rings, the costs of an outer ring edge is up to $2i$ times the costs of a spoke edge. As a result the edge-cost-part, which is at least $\sum_{e\in E} c_e f_e^{\min}$ (see Lemma 2.3), grows quickly with the number of rings. To see that the edge-cost-part dominates, we can use our bounds UB and LB from Lemma 2.3. Then UB − LB is the additional fixed-cost-part of our (crude) upper bound and we look at how large this value is with respect to LB in Fig. 6. As expected, this value decreases when increasing the number of rings, and using shorter (and more) lines is punished less and less.

On the family of randomly generated low-treewidth instances we observe a different behavior in Fig. 7. Here, the heuristics perform significantly worse than on circular cities, achieving only a median ratio of 0.444 compared to TW-LINES, with a minimum ratio of 0.236 as detailed on the right of Fig. 5.

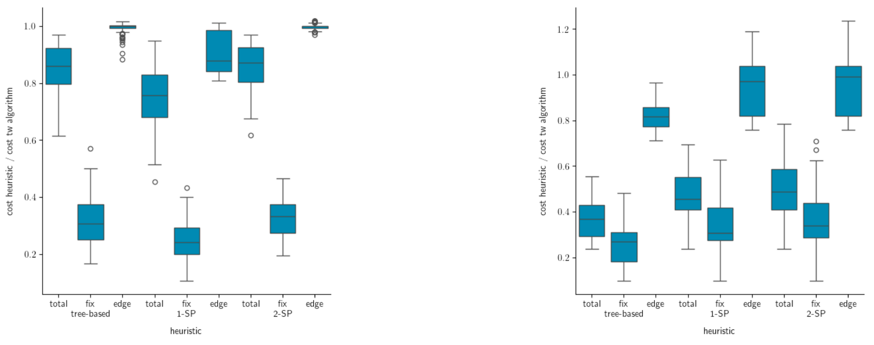

Fig. 5: The cost reduction values of TW-LINES compared to the different heuristics on circular cities (left) and random low-treewidth graphs (right). The horizontal lines in the boxes show the median value, the boxes are drawn from the first to the third quartile, and the whiskers go to the minimum and maximum value, excluding the outliers. An outlier is any point that is more than 1.5 times the interquartile range away from the box. Note that, to make the heuristics comparable, these plots are restriced to the instances where we have data from all three heuristics.

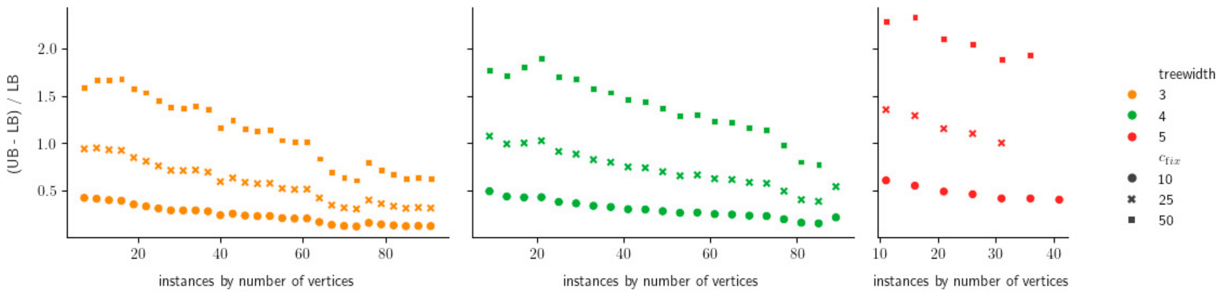

Fig. 6: Comparing the combinatorial upper bound (UB) and lower bound (LB) of Lemma 2.3: the plot shows the ratio $\frac{\text{UB}-\text{LB}}{\text{LB}}$.

In contrast to the circular cities, the performance of the heuristics does not improve for larger graphs, as evidenced by Fig. 7. Especially for treewidth 3, all heuristics perform very poorly for graphs of more than 200 vertices. This emphasizes the importance of considering all possible lines when finding a cost-optimal line concept.

5. Conclusion and outlook

In this paper, we evaluate the TW-LINES algorithm for finding cost-optimal line concepts for complete pools on two structurally different data sets. For the highly structured circular cities, we observe that TW-LINES outperforms traditional line-pool-generation-based solution approaches, especially when the fixed costs of a line are high. This effect is significantly more pronounced for randomly generated graphs of low treewidth. Here, the costs of the line concept computed by TW-LINES are as low as 23.6 % of the costs for traditional solution approaches. Additionally, we can find close-to-optimal solutions for graphs of up to 770 nodes within the time limit of one hour. Thus, we conclude that considering a complete line pool in cost-optimal line planning is crucial and that TW-LINES is a suitable solution approach for graphs of low treewidth.

We plan to adapt TW-LINES to further real-world requirements. By restricting the vertices where lines can start and end to a set of so-called terminals, we can incorporate vehicles' turn-around restrictions. Furthermore, this reduces the solution space considerably and might speed up TW-LINES further. Additionally, we intend to incorporate the passengers' perspective into TW-LINES by adding passenger routing in an iterative or integrated manner.

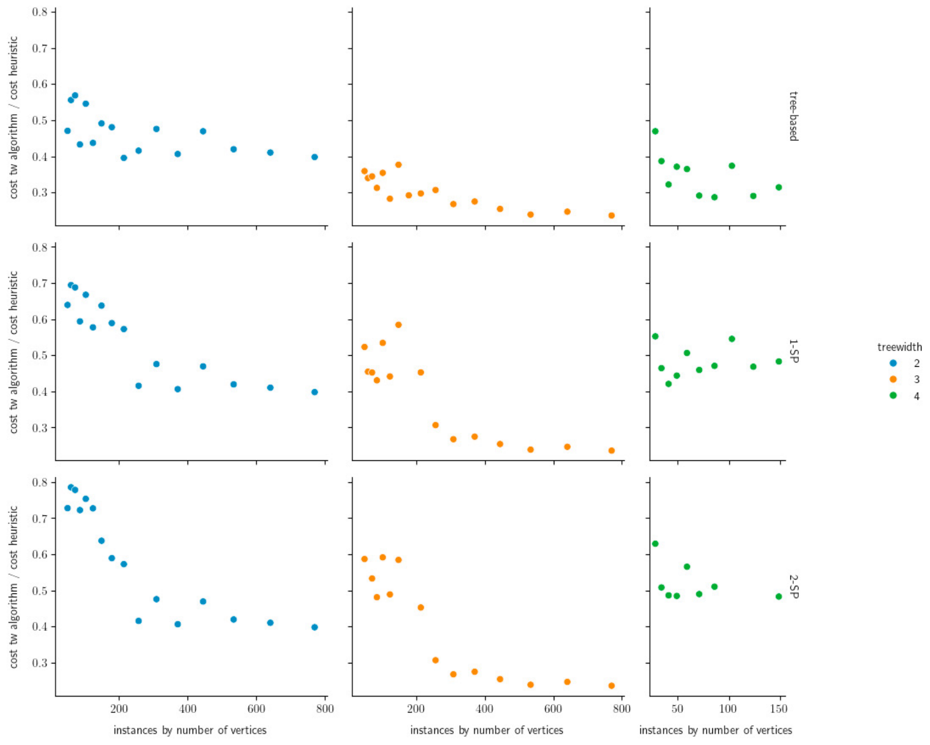

Fig. 7: The ratio between the cost of the solution obtained by TW-LINES and the heuristic solution on randomly generated graphs.

References

Bodlaender, H.L., 1998. A partial *k*-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. 209, 1–45.

Borndörfer, R., Grötschel, M., Pfetsch, M.E., 2007. A column-generation approach to line planning in public transport. Transp. Sci. 41, 123–132.

Gattermann, P., Harbering, J., Schöbel, A., 2017. Line pool generation. Public Transport 9, 7–32.

Guihaire, V., Hao, J.K., 2008. Transit network design and scheduling: A global review. Transportation Research Part A: Policy and Practice 42, 1251–1273.

Gurobi Optimization, LLC, 2023. Gurobi Optimizer Reference Manual.

Heinrich, I., Herrala, O., Schiewe, P., Terho, T., 2023a. Using light spanning graphs for passenger assignment in public transport, in: Frigioni, D., Schiewe, P. (Eds.), 23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2023, September 7-8, 2023, Amsterdam, The Netherlands, Schloss Dagstuhl - Leibniz-Zentrum für Informatik. pp. 2:1–2:16.

Heinrich, I., Schiewe, P., Seebach, C., 2022. Algorithms and hardness for non-pool-based line planning, in: D'Emidio, M., Lindner, N. (Eds.), 22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2022, Schloss Dagstuhl - Leibniz-Zentrum für Informatik. pp. 8:1–8:21.

Heinrich, I., Schiewe, P., Seebach, C., 2023b. Non-pool-based line planning on graphs of bounded treewidth, in: Frigioni, D., Schiewe, P. (Eds.), 23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2023, Schloss Dagstuhl - Leibniz-Zentrum für Informatik. pp. 4:1–4:19.

Rodrigue, J.P., Comtois, C., Slack, B., 2016. The Geography of Transport Systems.

Schiewe, P., Schöbel, A., Jäger, S., Albert, S., Baumgart, U., Biedinger, C., Grafe, V., Roth, S., Schiewe, A., Spühler, F., Stinzendörfer, M., Urban, R., 2023. LinTim: An integrated environment for mathematical public transport optimization. Documentation for version 2023.12. Technical Report. Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau.

Schmidt, M., Schöbel, A., 2024. Planning and optimizing transit lines. arXiv preprint arXiv:2405.10074 .

Schöbel, A., 2012. Line planning in public transportation: models and methods. OR Spectr. 34, 491–510.

Stokkink, P., de Palma, A., Geroliminis, N., 2023. Carpooling with transfers and travel time uncertainty, in: 11th Symposium of the European Association for Research in Transportation (hEART 2023).

UN General Assembly, 2015. Transforming our world : the 2030 Agenda for Sustainable Development. A/RES/70/1.