Strategies for Iteratively Refining Layered Graph Models*

Martin Riedler[1](✉), Mario Ruthmair[2], and Günther R. Raidl[1]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{riedler,raidl}@ac.tuwien.ac.at
[2] University of Vienna, Department of Statistics and Operations Research, Vienna, Austria
mario.ruthmair@univie.ac.at

Abstract. We consider a framework for obtaining a sequence of converging primal and dual bounds based on mixed integer linear programming formulations on layered graphs. The proposed iterative algorithm avoids the typically rather large size of the full layered graph by approximating it incrementally. We focus in particular on this refinement step that extends the graph in each iteration. Novel path-based approaches are compared to existing variants from the literature. Experiments on two benchmark problems—the traveling salesman problem with time windows and the rooted distance-constrained minimum spanning tree problem—show the effectiveness of our new strategies. Moreover, we investigate the impact of a strong heuristic component within the algorithm, both for improving convergence speed and for improving the potential of an employed reduced cost fixing step.

Keywords: Iterative Refinement · Layered Graphs · Integer Programming · Traveling Salesman Problem with Time Windows

1 Introduction

Layered graphs (LGs) are a well-known technique in mathematical programming to deal with specific constraints and restrictions in problems expressed on graphs. The basic idea is to construct an extended model that considers some problem dimension explicitly to make it easier to formulate certain constraints or even impose them implicitly. Picard and Queyranne [12] were among the first to consider such an approach. They modeled the time-dependent traveling salesman problem by introducing for each original node copies for all sequence positions at which it might be feasibly reached. Another typical application is related to distance restrictions in graphs. In such cases one can create node copies w.r.t. the feasible distances at which the original nodes can be reached. By omitting copies beyond the distance limit it is implicitly ensured that all paths in the extended graph adhere to the limit. If the dimension among which the original

* Supported by the Vienna Science and Technology Fund through project ICT15-014.

graph is extended corresponds to time, resulting LGs are sometimes called time-expanded networks. Such approaches are frequently considered for scheduling problems in which time is discretized to obtain so-called time-indexed models. For further details on LGs and associated mixed integer linear programming (MILP) formulations see the extensive survey by Gouveia et al. [8].

The main advantage of LG formulations is that they provide a convenient modeling option while usually leading to strong linear programming (LP) bounds. In many cases the LG is even acyclic and allows pseudo-polynomial formulations. However, there is also an important drawback involved: LGs and the associated models are typically much larger than simpler formulations on the original input. Frequently, this leads to models which are computationally impractical for reasonable problem sizes. However, often it is the case that already a subgraph of the full LG would suffice to encode an optimal solution. Several researchers used this observation to construct iterative algorithms that successively approximate the full LG until an optimal solution is found. This is usually done by omitting node copies and redirecting arcs. Among the first were Wand and Regan [16] who consider LG formulations for a pickup and delivery problem with time windows. In particular, they propose a relaxed formulation and a heuristic component that considers a subset of the feasible solutions. Those two formulations are successively extended until their bounds match, proving optimality. Ruthmair and Raidl [15] suggested such an iterative approach for the rooted distance-constrained minimum spanning tree problem (DCMST). Another successful application of a similar algorithm was proposed by Dash et al. [6] for solving the traveling salesman prblem with time windows (TSPTW). Their approach differs slightly from the former two as it refines the reduced LG only based on solving LP relaxations in a first stage. The final reduced LG is then used for solving an MILP in which the remaining infeasibilities are tackled by cutting planes. Further iterative refinement approaches in the network design area were considered by Macedo et al. [11], Boland et al. [3,4], and Clautiaux et al. [5].

Algorithms of this type that contain a component for obtaining heuristic solutions provide an eventually converging sequence of primal and dual bounds. Therefore, such an algorithm can also be terminated prematurely to obtain a high-quality primal solution together with a dual bound.

All previous works in this area have in common that they consider only a single strategy for extending the reduced LG in each iteration without evaluating alternatives. The employed techniques reach from rather simple approaches to more complex algorithms. In our previous work [13] we presented an extensive evaluation of different refinement techniques for a resource-constrained project scheduling problem (RCPSP). However, scheduling problems are somewhat special when it comes to the refinement step. In an aggregated set of time instants it is usually not clear what is the best/most promising option to reveal infeasibilities. Network design problems appear to be more accessible in this respect. An LG relaxation is typically obtained by redirecting arcs due to omitted layers which causes the arcs to no longer represent correct lengths. This makes it

straightforward how an arc can be corrected to the full extent. Moreover, we can use the knowledge regarding the true length of the arcs to get further insight from intermediate solutions. This is a crucial part of the algorithm and evaluating promising alternatives seems to be worthwhile.

In the following we start by introducing the necessary terminology of LGs in terms of an example problem: the TSPTW. In particular, we discuss how reduced LGs can be obtained, whose associated MILP models provide either primal or dual bounds. Then, we propose a generic refinement algorithm including several enhancements. Afterwards, we explain the specific refinement strategies and evaluate them in our computational experiments. In addition to existing strategies from the literature, we suggest new ones that aim at extracting more information from intermediate solutions. In the computational study we evaluate the discussed refinement strategies on several benchmark sets for the TSPTW. To show how the strategies behave on a structurally different problem we also conduct experiments for the rooted DCMST.

2 Mathematical Formalization

In the following we describe the construction of an LG and an associated MILP model. The process is exemplified in terms of the TSPTW. Afterwards, we characterize reduced LGs that serve as basis for the refinement algorithm introduced in the next section.

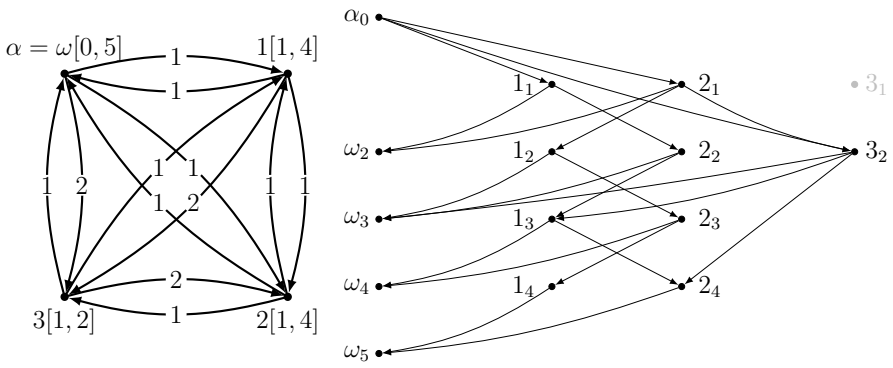
Notational remarks. For a graph $G = (V, A)$ and node subset $S \subseteq V$ let $\delta^+(S) = \{(i,j) \in A \mid i \in S,\, j \notin S\}$ be the set of outgoing and $\delta^-(S) = \{(j,i) \in A \mid i \in S,\, j \notin S\}$ be the set of incoming arcs. To simplify notation we omit the set braces for singletons S. Variable vectors are denoted in bold face. Solution vectors are indicated by a superscript "$*$".

2.1 Traveling Salesman Problem with Time Windows

The traveling salesman prblem with time windows (TSPTW) is defined on a directed graph $G = (V, A)$ with node set $V = \{\alpha, 1, \ldots, n, \omega\}$, associated arc costs $c\colon A \to \mathbb{Z}_{\geq 0}$, and travel times $t\colon A \to \mathbb{Z}_{>0}$. As in [6], we represent the depot by two distinct nodes α and ω in order to model a tour starting and ending at the depot as path. Each node $i \in V$ is associated with a time window $[r_i, d_i]$ with $r_i \leq d_i$. Service times for the nodes can be incorporated into the travel times and are therefore not considered separately. The goal is to find a least cost Hamiltonian path through V starting at α and ending at ω s.t. all nodes are visited within their time windows. Waiting at nodes is allowed in case of early arrival.

2.2 Layered Graph Model

We consider the layered digraph $G_{\mathrm{L}} = (V_{\mathrm{L}}, A_{\mathrm{L}})$. Initially, node set $V_{\mathrm{L}} = \{i_l \mid i \in V,\, l \in [r_i, d_i]\}$ contains all node copies that are feasible w.r.t. the time windows.



(a) Input graph (b) Full layered graph

Fig. 1. Example of a layered graph for a TSPTW instance with $t = c$.

Thereby, node copy $i_l \in V_{\mathrm{L}}$ at layer l represents original graph node $i \in V$ reached at time l. To get an abstraction for connecting the layered node copies we introduce function $\theta(i_l, j) \coloneqq \max(r_j, l + t_{(i,j)})$ that provides the layer at which node j is reached when starting at node i at layer l. The obtained arc set is $A_{\mathrm{L}} = \{(i_l, j_m) \mid i_l, j_m \in V_{\mathrm{L}}, (i, j) \in A, \theta(i_l, j) = m\}$. To obtain a smaller graph we remove all unnecessary node copies—and their incident arcs—that cannot be reached from depot copy α_{r_α}[3]. For an example see Fig. 1 where node 3_1 is not reachable from α_0.

We model the TSPTW in terms of binary arc variables x_a, for $a \in A$, indicating which arcs of the original graph are part of the tour, and non-negative arc variables z_a for the LG. Observe that this problem could also be modeled without the original graph variables. However, these variables are convenient for imposing certain strengthening inequalities and beneficial for the reduced cost fixing explained in Section 3.

$$\text{(TSPTW-L)} \quad \min \quad \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{s.t.} \quad \sum_{i_l \in V_{\mathrm{L}}} \sum_{a \in \delta^-(i_l)} z_a = 1 \qquad \forall i \in V \setminus \{\alpha\}, \tag{2}$$

$$\sum_{a \in \delta^+(i_l)} z_a = \sum_{a \in \delta^-(i_l)} z_a \qquad \forall i_l \in V_{\mathrm{L}}, \tag{3}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}} z_{(i_l, j_m)} = x_{(i,j)} \qquad \forall (i, j) \in A, \tag{4}$$

$$\boldsymbol{x} \in \{0, 1\}^{|A|}, \, \boldsymbol{z} \in \mathbb{R}_{\geq 0}^{|A_{\mathrm{L}}|}. \tag{5}$$

[3] When referring to the full LG G_{L} in the following, we assume this step to be completed.

The model presented above can be strengthened[4] by well-known cut-set inequalities of the following form:

$$\sum_{a \in \delta^-(W)} x_a \geq 1 \qquad \forall W \subseteq V \setminus \{\alpha\},\, W \neq \emptyset. \tag{6}$$

Stronger cut-set inequalities can be specified w.r.t. the z variables (see [8]):

$$\sum_{a \in \delta^-(W)} z_a \geq 1 \qquad \forall W \subseteq V_{\mathrm{L}} \setminus \{\alpha_{r_\alpha}\},\, W \neq \emptyset,\, \exists v \in V : \{v_l \in V_{\mathrm{L}}\} \subseteq W. \tag{7}$$

Both sets of cut-set inequalities are of exponential size and require dynamic separation in practice. Although the original graph cut-set inequalities are known to be weaker than their LG counterpart, they are still worth considering due to faster convergence as a result of the smaller size of the original graph.

2.3 Reduced Layered Graphs

The full LG defined above guarantees that the associated MILP model, denoted by TSPTW-L(G_{L}), contains all feasible solutions that are possible w.r.t. the original graph. However, depending on the number of layers and the density of the original graph we often end up with a problematic model size. Smaller graphs can be extracted from the full LG by either giving up optimality or feasibility. For pragmatic reasons we require each reduced LG $G'_{\mathrm{L}} = (V'_{\mathrm{L}}, A'_{\mathrm{L}})$ to contain at least one copy for each node of the original graph, i.e., $V = \{i \mid i_l \in V'_{\mathrm{L}}\}$.

Due to the omitted node copies, arcs are redirected. We say that an arc (i_l, j_m) is *shortened* (*lengthened*) if there exists an arc (i_l, j_k) in the full LG s.t. $m < k$ ($m > k$), otherwise it has the *correct* length.

Dual Layered Graphs A *dual* LG $G_{\mathrm{dL}} = (V_{\mathrm{dL}}, A_{\mathrm{dL}})$ is obtained by considering only a subset of the layered node copies $V_{\mathrm{dL}} \subseteq V_{\mathrm{L}}$ inducing the reduced arc set

$$A_{\mathrm{dL}} = \{(i_l, j_m) \mid i_l, j_m \in V_{\mathrm{dL}},\, (i,j) \in A,\, m \leq \theta(i_l, j), \\ \nexists m' \; (m < m' \leq \theta(i_l, j) \wedge j_{m'} \in V_{\mathrm{dL}})\}.$$

In short, this means that if a layered node is present in V_{dL} but the target of its outgoing arc according to G_{L} is not, then we use the copy of the target node at the maximum layer no larger than the originally used copy and omit the arc if such a copy does not exist. An example is provided in Fig. 2.

In order to guarantee that the associated MILP model is a relaxation we only consider node subsets $V_{\mathrm{dL}} \subseteq V_{\mathrm{L}}$ s.t. $i_l \in V_{\mathrm{dL}} \wedge (i_l, j_m) \in A_{\mathrm{L}} \implies \exists m' \; (m' \leq m \wedge (i_l, j_{m'}) \in A_{\mathrm{dL}})$. This ensures that we loose no connections that were present in the original graph. By using only shortened and correct arcs we never arrive at a node on a higher layer than in the full LG. As a result TSPTW-L(G_{dL})

[4] In case of cycles due to zero travel times, these inequalities become mandatory.

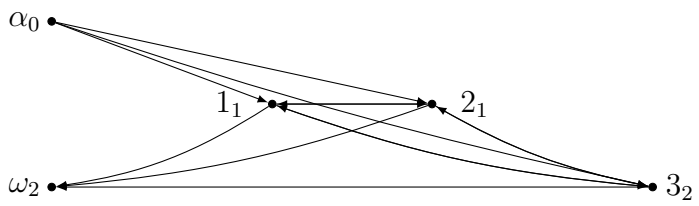


Fig. 2. Example of a dual layered graph w.r.t. the TSPTW instance provided in Fig. 1.

is a relaxation w.r.t. the original problem. Consequently, the LP relaxation of TSPTW-L(G_{dL}) yields a dual bound. Moreover, if an optimal integral solution to TSPTW-L(G_{dL}) is feasible (on the x variables) w.r.t. the original problem, then it is guaranteed to be optimal. Observe that the dual LG is—opposed to the full LG—usually not acyclic. Therefore, separating cut-set inequalities is necessary to obtain a connected solution.

Primal Layered Graphs A primal LG $G_{\mathrm{pL}} = (V_{\mathrm{pL}}, A_{\mathrm{pL}})$ is obtained by considering only a subset of the layered node copies $V_{\mathrm{pL}} \subseteq V_{\mathrm{L}}$ and an associated induced arc set

$$A_{\mathrm{pL}} = \{(i_l, j_m) \mid i_l, j_m \in V_{\mathrm{pL}},\, (i,j) \in A,\, m \geq \theta(i_l, j), \\ \nexists m' \,(\theta(i_l, j) \leq m' < m \wedge j_{m'} \in V_{\mathrm{pL}})\}.$$

This time we redirect arcs to the node copy at the minimum layer at least as large as the original one. Therefore, the primal LG turns its associated MILP model into a heuristic as it may exclude feasible solutions—possibly to the extent that no solutions remain. A feasible solution to TSPTW-L(G_{pL}) provides a primal bound but cannot be shown to be optimal on its own—not even if all layered arcs associated with the selected z variables have the correct length.

2.4 Other Problems

The definitions provided above can easily be adjusted to other problems. To cover the rooted DCMST (see [9])—for which we also perform experiments in Section 4—it suffices to redefine function θ to $\theta(i_l, j) \coloneqq l + d_{(i,j)}$, i.e., waiting is not permitted/necessary. The problem's distance restriction can be imagined as time window for each node with a lower bound of zero and an upper bound equal to the global distance limit. A suitable MILP model can then be obtained by taking the model for the TSPTW and replacing constraints (3) by

$$z_{(i_l, j_m)} \leq \sum_{(k_h, i_l) \in \delta^-(i_l) : k \neq j} z_{(k_h, i_l)} \quad \forall i_l \in V_{\mathrm{L}},\, \forall (i_l, j_m) \in \delta^+(i_l). \tag{8}$$

3 Algorithmic Framework

In this section we describe our iterative refinement algorithm. In particular, we consider different refinement strategies that are used to iteratively extend an initially small dual LG.

To simplify the description in what follows, we make some assumptions on the considered input problem. We focus on problems for which each node must be connected to a designated source node. This guarantees that we can perform the necessary path computations in the LG for some of the refinement strategies presented in Section 3.2. Consequently, suitable connectivity inequalities must be available (e.g., cut-set inequalities (7) for the TSPTW). This assumption might seem restrictive at first but actually covers a large variety of problems. Depot based routing problems as well as most network design problems are compatible with this restriction. In addition, we assume that the model is specified as minimization problem and includes at least design variables $\boldsymbol{x}$ for the original graph arcs and design variables $\boldsymbol{z}$ for the LG variables—further auxiliary variables are of course possible. Note that these conditions are more strict than necessary but being more general would go beyond the scope of this work.

3.1 Iterative Refinement Algorithm

The main idea of our iterative refinement algorithm (IRA) is to start with a small dual graph and solve the associated MILP model or its corresponding LP relaxation, respectively. The result is then used to either prove optimality or—if this cannot be done—to obtain a larger dual graph (closer to the full LG) for repeating the procedure. This step of adding not-yet-present node copies of the full LG to the dual LG is called *refinement*. If the refinement adds at least one new node copy in each iteration, then it is guaranteed that the algorithm terminates with an optimal solution in finitely many iterations since the dual graph eventually converges to the full LG.

Algorithm 1 provides the detailed procedure. The mentioned gap refers to the absolute difference between the current dual (db) and primal (pb) bounds and is considered to be closed if $db \geq pb$. In the beginning we need an initial dual LG. This step depends on the problem at hand. For the TSPTW—and many other problems—a minimal starting graph that satisfies the restrictions imposed above can be obtained by considering for each original graph node the copy at the smallest feasible layer. Based on this initial dual LG we solve the LP relaxation of the associated MILP model. The obtained solution value is a dual bound and can be used to prove optimality if a primal bound is available. In order to get a more meaningful solution for the subsequent refinement process, we assume the LP to be extended by connectivity inequalities. If optimality could not be proven yet, we use a refinement algorithm to identify possible infeasibilities in the relaxed solution. If infeasibilities could be detected, we add further nodes to the graph to reveal them. Otherwise, we test whether the obtained LP solution is integral. An integral solution that is feasible must be optimal according to the construction of the dual LG. A fractional solution, on the other hand, might prevent the

Algorithm 1: Iterative refinement algorithm (IRA)

```
1  while termination condition not met do
2  |  solve LP; stop if gap closed
3  |  refine LG
4  |  if solution is integer and feasible then terminate    // optimal solution
5  |  apply primal heuristic; stop if gap closed
6  |  apply reduced cost fixing
7  |  if graph could be refined then continue with next iteration
8  |  solve IP
9  |  refine LG
10 |  if solution is feasible then terminate                // optimal solution
11 |  apply primal heuristic; stop if gap closed
12 end
```

refinement algorithm from detecting remaining infeasibilities. Therefore, we solve the MILP model in the following. However, before doing this, we can apply a heuristic (guided by the current fractional solution) to obtain a primal bound to possibly close the gap and prove optimality. Furthermore, we can use the obtained primal bound to attempt *reduced cost fixing* w.r.t. the x variables of the MILP model. To this end let db be the current solution value of the LP relaxation, pb the current primal bound, $\boldsymbol{x}^*$ the solution vector of the original graph variables, and $\boldsymbol{x}^{\mathrm{r}}$ the vector of reduced costs of the x variables. For each arc $a \in A$ we consider two cases. If $x_a^* = 0 \wedge db + x_a^{\mathrm{r}} \geq pb$, we can remove arc a from the input graph and consequently all its copies in any LG. On the other hand, if $x_a^* = 1 \wedge db - x_a^{\mathrm{r}} \geq pb$, we know that arc a must be part of an optimal solution and its associated variable can therefore be fixed to one in subsequent iterations. When the algorithm is already close to convergence, reduced cost fixing might fix a sufficient number of variables to zero s.t. the model becomes infeasible, proving optimality of the solution that provided the current primal bound.

Unless one of the previous considerations allowed proving optimality, we are now in the situation that the (fractional) LP solution does not allow the refinement algorithm to identify the remaining infeasibilities. In this unfortunate case we have to take the additional computational burden and solve the MILP to optimality. If the integral solution is feasible, we proved optimality. Otherwise, we apply the primal heuristic once more before solving the LP relaxation according to the refined dual LG.

Primal graph heuristic. Steps 5 and 11 of Algorithm 1 can in principle be realized by any suitable heuristic. Problem-dependent algorithms typically provide better solution quality but are sometimes tedious to implement and often have to be replaced completely if a slightly different problem variant is considered. A more convenient problem-independent way to obtain heuristic solutions is to use the MILP formulation on the primal LG. We construct the primal LG by taking

the node set of the dual LG and add for each node a copy at the maximum feasible layer. This enables us to benefit from the iterations made so far while reducing the risk of obtaining a graph that encodes no feasible solution. Primal graph heuristics of this type were considered in [5,14,15,16].

3.2 Refinement Procedures

The perhaps most crucial part of IRA is the refinement step. Solutions w.r.t. the current dual LG typically contain multiple infeasibilities and it is usually not clear how they can be handled most efficiently. In this context one has to deal (among others) with the following important questions: (a) for which nodes should further copies be added, (b) how many copies should be added, and (c) on which layers should the copies be inserted. Answering those questions typically involves keeping a suitable balance between the growth of the dual LG and the number of iterations IRA has to complete before proving optimality. The latter is quite important as it determines how often the MILP solver has to be invoked which is usually the most time-consuming part of the algorithm. On the other hand, the time each invocation takes increases with the size of the associated dual LG.

Full Infeasible Arc Refinement (FAR) The probably most straightforward refinement strategy simply refines all nodes that are part of the current solution. To this end, we consider all layered arcs whose associated solution value is non-zero. All shortened arcs are fully corrected by adding the appropriate target node to the dual LG. To avoid refining already feasible solutions, we check if the solution w.r.t. the x variables is feasible before starting the refinement process. This refinement procedure was employed in [4,11,14,15].

Infeasible Path Refinement (PR) Instead of considering all arcs, we only consider those that are most relevant from the structural perspective. We start by constructing an auxiliary LG that is obtained by taking all arcs of the dual LG with associated non-zero z variable value in the current solution. Based on this graph we compute a shortest path w.r.t. time, using travel times weighted by $1 - z_a^*$, to each node and determine the effective time at which the node would be reached. If the resulting arrival time is incompatible with the node's time window, we compute a refinement. This is done by traversing the path backwards and refining each arc as done for FAR stopping once we reach a node for which the effective time is equivalent to its layer.

Repeated Infeasible Path Refinement (RPR) We start by performing PR. In a subsequent step, we check for each formerly infeasible path if it is still contained in the adjusted dual LG—traversing different node copies but still reaching the target node after its time window when considering the effective length of the path—and repeat the refinement step until the path is no longer present.

Single-Copy Infeasible Path Refinement (SPR) Especially in later iterations the dual LG contains multiple layered copies w.r.t. each node of the original graph. We perform the same approach as done in PR, however, for each node of the original graph we only compute a refinement for the path reaching the node at the latest effective time, i.e., the most infeasible path.

Minimum Sum of Negative Waiting Times (DASH) This strategy was developed by Dash et al. [6]. Similar to the other techniques they consider the subgraph G'_{dL} induced by non-zero z variable values. If a node copy v_l is reached by shortened arcs, a new node copy is added that minimizes the sum of *negative waiting times*. The negative waiting time of an arc is essentially the amount by which it was shortened, i.e., if $(i_l, j_m) \in A'_{\mathrm{dL}}$ and $(i_l, j_k) \in A_{\mathrm{L}}$, then the negative waiting time is $k - m$. A new layered copy of node v is added at the layer λ that achieves the minimum when computing the sum of negative waiting times weighted by the associated arcs' solution values. Let $\boldsymbol{z}^*$ be the current solution vector and let $I_{v_l} = \{((j_m, v_l), l') \mid (j_m, v_l) \in A'_{\mathrm{dL}}, (j_m, v_{l'}) \in A_{\mathrm{L}}\}$. Then we seek the layer λ that minimizes $\mu(v_l, \lambda) = \sum_{(a,l') \in I_{v_l} : l' < \lambda} (l' - l) z^*_a + \sum_{(a,l') \in I_{v_l} : l' \geq \lambda} (l' - \lambda) z^*_a$. Dash et al. do not indicate which value is used if there are multiple options for λ that achieve the minimum. Function μ is piece-wise linear for a fixed v_l and changes slope only at points at which at least one arc arrives at the correct time. Therefore, it makes sense to restrict the procedure to such values, i.e., $l' - \lambda$ is zero for at least one element from I_{v_l}, as this guarantees to reduce the number of shortened arcs. However, this still might leave several options. In preliminary experiments we tested using either the smallest or the largest value of λ that achieves the minimum. The performance was roughly the same with a slight advantage for the latter.

Again we check feasibility w.r.t. the original graph variables to avoid superfluous refinements.

4 Computational Study

Our algorithms are implemented in C++ using CPLEX 12.8.0 as general-purpose MILP solver. All experiments have been performed in single thread mode with default parameter settings. For performance reasons the implicitly integral LG variables z are implemented as binary variables together with a cost-based branching priority to focus on the original graph variables. Experiments have been executed on an Intel Xeon E5540 machine with 2.53 GHz. The computation time limit has been set to 7200 seconds and the memory limit to 8 GB RAM. To test our framework we consider two benchmark sets for the TSPTW from http://lopez-ibanez.eu/tsptw-instances. The first set is from [1] and contains 50 instances while the second one was proposed in [7] and contains 135 instances. In addition, we also tested on instances for the rooted DCMST by [9]. Experiments were limited to the subset of 60 "TE" instances with distances ranging to 10, 100, and 1000 as the other instances turned out to be too easy. The TSPTW

instances were preprocessed as described in [1] and the DCMST instances as described in [14].

We want to emphasize that our aim is to compare the different refinement strategies on a common basis and not to beat the state of the art. Achieving the latter would require further problem-specific tuning and incorporation of additional strengthening inequalities which is not the focus of this work.

4.1 Experiments

In the upcoming tables we present averages for gaps, computation times, the number of iterations in which the LP relaxation was solved (itr), the number of iterations in which the MILP was solved (itr-ip) as well as the number of nodes and arcs in the final LGs. Instances that terminated due to the memory limit are omitted when computing averages and those that ran into the time limit are considered with a value of 7200 seconds. Gaps are computed by $(pb^* - db)/pb^*$ where db is the dual bound of the respective run and pb^* is the best primal bound known for the respective instance. The remaining columns report the number of runs that ran into the time limit (tl) or the memory limit (ml), respectively, and the number of instances solved to proven optimality (opt).

For the TSPTW we consider each refinement strategy in three variants: without a primal component, with an initially provided primal solution ("HS"), and with an initially provided primal solution and reduced cost fixing activated ("HS_RCF"). We do this in order to show two things: (1) the benefits of a strong primal component and (2) the potential of reduced cost fixing if a high-quality solution is available. High-quality heuristic solutions for the Ascheuer instances were obtained from http://lopez-ibanez.eu/tsptw-instances and optimal solutions for the instances by Dumas et al. were obtained from http://homepages.dcc.ufmg.br/~rfsilva/tsptw.

Table 1 reports our results on the instances by Ascheuer et al. [1]. The first observation is that directly solving the MILP on the full LG (MIP) is not effective. The size of the associated model leads either to problems with the memory limit or to long runs that can frequently not be completed within the time limit. Consequently, the remaining gap is quite large with more than 10 % on average. All variants of our refinement algorithm perform much better. The most striking difference is that we deal with considerably smaller graphs that help to avoid any memory issues. This enables us to solve significantly more instances to optimality. Among the various refinement strategies we observe that the naive approach (FAR) solves the fewest instances to optimality while leading to the largest graph sizes. The approach by Dash et al. [6] works noticeably better and solves one more instance to optimality but requires comparatively large graphs. Our new path-based strategies solve the largest number of instances to optimality. The drawback of these rather careful and minimalist approaches is that they require a higher number of iterations to converge, even including some iterations where the MILP has to be solved, which is not necessary for FAR and DASH. Nevertheless, we observe the smallest average computation times for these strategies. The more slowly growing graphs outweigh the higher number of iterations through

Table 1. Results on the TSPTW instances by Ascheuer et al. [1]

Algorithm	Gap [%]	Time [s]	#itr	#itr-ip	$\|V\|$	$\|A\|$	#tl	#ml	#opt
MIP	20.24	3800	-	-	78437	1339490	21	5	24
IRA_FAR	0.09	2230	**17.9**	**0.0**	547	9820	14	0	36
IRA_DASH	0.08	2114	18.8	**0.0**	531	9546	13	0	37
IRA_PR	0.08	2090	31.4	3.5	**398**	**7663**	13	0	37
IRA_RPR	0.08	**2007**	22.1	2.5	417	8010	12	0	**38**
IRA_SPR	0.08	2080	32.6	3.6	399	7761	12	0	**38**
IRA_FAR_HS	0.08	2215	**15.1**	**0.0**	503	9248	14	0	36
IRA_DASH_HS	0.08	2064	15.5	**0.0**	480	8872	13	0	37
IRA_PR_HS	0.08	2038	27.9	3.2	**376**	**7379**	12	0	**38**
IRA_RPR_HS	0.08	**1937**	19.1	1.9	394	7710	12	0	**38**
IRA_SPR_HS	0.08	2068	29.2	3.1	377	7398	12	0	**38**
IRA_FAR_HS_RCF	0.08	2047	**15.3**	**0.0**	492	8340	13	0	37
IRA_DASH_HS_RCF	0.08	1935	15.4	**0.0**	475	7985	13	0	37
IRA_PR_HS_RCF	0.08	2096	29.2	3.6	**384**	**6876**	13	0	37
IRA_RPR_HS_RCF	0.08	**1933**	19.2	2.0	399	7190	12	0	**38**
IRA_SPR_HS_RCF	0.08	2001	30.6	3.3	**384**	6948	12	0	**38**

the smaller associated models. Among the three path-based approaches, we see that RPR performs best.

Providing high-quality initial solutions improves all variants of IRA alike. We observe a decrease in the number of iterations as well as the final graph sizes. This shows that a tight dual bound is sometimes obtained before feasibility can be established through further refinement steps. Enabling reduced cost fixing helps to improve the results further. For strategy PR we observe a minor slowdown compared to the variant in which only the initial solution is provided. The reason is that solution quality and refinement quality are not directly correlated. A weaker solution might lead to a very successful refinement in a subsequent iteration that is not reached by a better solution. The slowdown, however, is not dramatic and we achieve the smallest final graph size with this approach.

In Table 2 we provide the results on the instances by Dumas et al. [7]. Compared to the Ascheuer instances this set features much narrower time windows (100 at most). Therefore, the MILP on the full LG performs considerably better. Although it no longer faces problems with the memory limit, it is still not able to solve all instances to optimality within the time limit. The remaining gap is rather small but could also not be closed completely. In terms of computation times we again observe a clear advantage for IRA. The performance of the different refinement strategies is comparable to what we observed for the Ascheuer instances. Strategies FAR and DASH require larger graphs but converge within fewer iterations. The path-based approaches, on the other hand, lead to much smaller final graphs but also have to complete some iterations in which the MILP is solved. Providing an initial primal solution again improves the results significantly. This time reduced cost fixing provides a consistent improvement and

Table 2. Results on the TSPTW instances by Dumas et al. [7]

Algorithm	Gap [%]	Time [s]	#itr	#itr-ip	$\|V\|$	$\|A\|$	#tl	#ml	#opt
MIP	0.42	2000	-	-	3276	37069	29	0	106
IRA_FAR	0.00	250	10.5	0.2	285	3597	1	0	134
IRA_DASH	0.00	240	10.7	**0.1**	282	3549	1	0	134
IRA_PR	0.00	80	13.1	4.0	142	1719	0	0	**135**
IRA_RPR	0.00	**55**	**9.2**	3.0	145	1773	0	0	**135**
IRA_SPR	0.00	93	13.2	4.1	**141**	**1711**	0	0	**135**
IRA_FAR_HS	0.00	110	**8.2**	**0.1**	256	3207	0	0	**135**
IRA_DASH_HS	0.00	123	8.3	**0.1**	251	3135	0	0	**135**
IRA_PR_HS	0.00	65	12.3	3.5	139	1681	0	0	**135**
IRA_RPR_HS	0.00	**45**	8.4	2.5	139	1691	0	0	**135**
IRA_SPR_HS	0.00	73	12.4	3.5	**138**	**1674**	0	0	**135**
IRA_FAR_HS_RCF	0.00	44	**8.0**	**0.1**	255	2247	0	0	**135**
IRA_DASH_HS_RCF	0.00	45	8.3	**0.1**	250	2187	0	0	**135**
IRA_PR_HS_RCF	0.00	56	12.4	3.5	**138**	1395	0	0	**135**
IRA_RPR_HS_RCF	0.00	**42**	8.3	2.5	**138**	1401	0	0	**135**
IRA_SPR_HS_RCF	0.00	55	12.4	3.5	**138**	**1387**	0	0	**135**

does not suffer from side effects. It is even effective enough to almost improve the slower refinement strategies to the level of the better ones through variable fixes that significantly reduce the LG size.

Finding feasible solutions to the TSPTW is NP-hard (see [1]) but can be done in (pseudo-)polynomial time for the rooted DCMST. Therefore, we use this problem to show the performance of a simple problem-specific heuristic ("PHeu") in comparison to the general purpose heuristic based on the primal LG ("PG"). The considered problem-specific heuristic iteratively computes a resource-constrained shortest path to a still unreached node farthest from the source. The costs of the thereby added arcs are set to 0 for the next iteration and the procedure is stopped once all nodes are connected to the source. We use the solution on the x variables as guidance by operating on adjusted costs weighted by $1 - x^*_a$. We do not use reduced cost fixing here to avoid side effects that could influence the results. The outcome of the experiments on the DCMST instances is summarized in Table 3.

The MILP model on the full LG once more solves the fewest instances to optimality while being the slowest algorithm on average. The reason why the MILP is not that far off this time is that the considered instance set includes also small distance limits. For small and medium distance limits the MILP is competitive while it is clearly outperformed for the larger ones or cannot be solved due to the memory limit. Again, all variants of the iterative approach outperform the pure MILP approach for the larger distance restrictions. Strategies FAR and SPR do not work as well as the other strategies. The former appears to refine too unstructured while the latter does not make enough progress resulting in a comparatively high number of iterations. Among the remaining three variants

Table 3. Results on the hard DCMST instances (TE) by Gouveia et al. [9]

Algorithm	Gap [%]	Time [s]	#itr	#itr-ip	$\lvert V\rvert$	$\lvert A\rvert$	#tl	#ml	#opt
MIP	0.72	3063	-	-	23833	529314	13	13	34
IRA_FAR	0.08	2481	21.9	**0.0**	474	10827	9	0	51
IRA_DASH	0.05	2118	22.4	0.1	473	10816	6	0	54
IRA_PR	0.05	2555	23.6	0.2	418	9703	7	0	53
IRA_RPR	0.04	**1913**	**18.0**	0.1	481	10767	4	0	**56**
IRA_SPR	0.06	2601	24.3	0.2	**415**	**9640**	10	0	50
IRA_FAR_PHeu	0.07	2211	21.5	**0.0**	471	10769	6	0	54
IRA_DASH_PHeu	0.05	2058	21.9	**0.0**	472	10793	5	0	55
IRA_PR_PHeu	0.04	2256	23.0	0.1	417	9678	6	0	54
IRA_RPR_PHeu	0.04	**1680**	**17.8**	0.1	482	10781	4	0	**56**
IRA_SPR_PHeu	0.04	1987	23.6	0.2	**414**	**9632**	4	0	**56**
IRA_FAR_PG	0.08	2453	21.2	**0.0**	469	10709	9	0	51
IRA_DASH_PG	0.05	2185	21.5	**0.0**	467	10675	5	0	55
IRA_PR_PG	0.04	2269	23.3	0.2	416	9659	6	0	54
IRA_RPR_PG	0.04	**1816**	**17.6**	0.1	479	10702	4	0	**56**
IRA_SPR_PG	0.07	2553	23.9	0.2	**413**	**9606**	9	0	51

we observe that RPR works best. Although it leads to larger graphs than the other path-based strategies, it turned out to be quite fast. Apparently, the repeated refinement helps to significantly reduce the number of iterations which compensates for the larger graph size.

Adding heuristics significantly decreases computation times and allows solving further instances to proven optimality. The primal LG heuristic turns out to be a valuable alternative to the problem-specific one. Nevertheless, we want to point out that it strongly depends on the problem whether the generic approach works well. Preliminary experiments for the TSPTW showed that it can be difficult to obtain feasible solutions if the underlying problem is challenging in this respect. Node copies corresponding to an initial heuristic solution might be inserted into the LG to resolve these issues.

Finally, we applied the one-tailed Wilcoxon signed-rank test for RPR and each other refinement strategy (without heuristics or reduced cost fixing). The alternative hypothesis that RPR is faster was assumed with a significance level of 0.05, except for the instances by Ascheuer where PR, SPR, and DASH performed too similar, mainly due to the comparatively high number of unsolved instances.

5 Conclusion and Future Work

In this work we considered a general framework for iteratively refining a reduced layered graph (LG). Based on solutions to an associated mixed integer linear programming formulation and heuristically obtained primal bounds the approach converges towards proven optimality if given enough time. In particular, we

focused on one of the crucial points of such algorithms which did not receive much attention in previous works: the refinement step that extends the LG in each iteration. We investigated strategies from the literature and suggested new path-based ones. Through our experiments on two benchmark problems we could show that the previous approaches work reasonably well but still leave room for improvement. The path-based approaches are able to solve a higher number of instances to optimality while leading to smaller LGs in the final iteration. We also showed that a strong heuristic component is important for the algorithm to converge faster and to provide high-quality (intermediate) solutions.

A problem-independent heuristic was shown to be competitive with a simple problem-specific one. Future work could put more effort into this component to improve the obtained results. In this work we focused on refinement strategies for the reduced LG that is used to compute dual bounds. However, one may also consider refinements based on the LG that is used to obtain heuristic solutions.

For brevity, we restricted the discussion to problems with a designated source node to which all other nodes must be connected. However, the method can in principle be applied to any network design problem for which feasibility of the relaxation may be checked by path computations. Interesting problems are those whose resource dependencies can be naturally modeled through LGs. This especially includes problems with resource-dependent (non-linear) costs, e.g., time-dependent travel times. If many layers are present of which only few are assumed to be traversed, the iterative algorithm is expected to work particularly well. In general, the approach does not work for applications represented by a cyclic LG because the described dual LGs not necessarily represent a relaxation for them. Typical examples are pickup and delivery problems with increasing and decreasing load along the route as well as the energy state in electric vehicle routing.

To be comparable to the state of the art further tuning would be necessary for both benchmark problems. In terms of the traveling salesman prblem with time windows our algorithms struggle in particular with some of the harder Ascheuer instances. A promising solution appears to be the inclusion of information related to node precedences as done, e.g., in [2,6]. Concerning the rooted distance-constrained minimum spanning tree problem our results are already quite close to the state-of-the-art column generation approach in [10] with only four instances that could not be solved to optimality.

In preliminary experiments we tested a cleanup algorithm that removes nodes from the LG that were not used for a specified number of iterations. This approach showed potential to decrease the final graph sizes further. Unfortunately, we ran into problems with cycling that increased the number of iterations, negating the provided benefits. Future research could address these issues by more complex cleanup or cycle-prevention strategies. Having shown that even smaller final graph sizes can be achieved, we think that a more theoretical investigation could prove useful. Computing minimal or even minimum LGs that lead to tight dual bounds or optimal solutions could serve as starting point to design more elaborate refinement strategies.

References

1. Ascheuer, N., Fischetti, M., Grötschel, M.: Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math. Programming, Ser. B **90**(3), 475–506 (2001)
2. Baldacci, R., Mingozzi, A., Roberti, R.: New state-space relaxations for solving the traveling salesman problem with time windows. INFORMS J. Comput. **24**(3), 356–371 (2012)
3. Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M.: The continuous-time service network design problem. Oper. Res. **65**(5), 1303–1321 (2017)
4. Boland, N., Hewitt, M., Vu, D.M., Savelsbergh, M.: Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks. In: Salvagnin, D., Lombardi, M. (eds.) CPAIOR 2017: Integration of AI and OR Techniques in Constraint Programming. LNCS, vol. 10335, pp. 254–262. Springer, Cham (2017)
5. Clautiaux, F., Hanafi, S., Macedo, R., Voge, M.É., Alves, C.: Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. Eur. J. Oper. Res. **258**(2), 467–477 (2017)
6. Dash, S., Günlük, O., Lodi, A., Tramontani, A.: A time bucket formulation for the traveling salesman problem with time windows. INFORMS J. Comput. **24**(1), 132–147 (2012)
7. Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M.M.: An optimal algorithm for the traveling salesman problem with time windows. Oper. Res. **43**(2), 367–371 (1995)
8. Gouveia, L., Leitner, M., Ruthmair, M.: Layered graph approaches for combinatorial optimization problems. Computers & Operations Research **102**, 22–38 (2019)
9. Gouveia, L., Paias, A., Sharma, D.: Modeling and solving the rooted distance-constrained minimum spanning tree problem. Comput. Oper. Res. **35**(2), 600–613 (2008), Part Special Issue: Location Modeling Dedicated to the memory of Charles S. ReVelle
10. Leitner, M., Ruthmair, M., Raidl, G.R.: Stabilizing branch-and-price for constrained tree problems. Networks **61**(2), 150–170 (2013)
11. Macedo, R., Alves, C., de Carvalho, J.M.V., Clautiaux, F., Hanafi, S.: Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. Eur. J. Oper. Res. **214**(3), 536–545 (2011)
12. Picard, J.C., Queyranne, M.: The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Oper. Res. **26**(1), 86–110 (1978)
13. Riedler, M., Jatschka, T., Maschler, J., Raidl, G.R.: An iterative time-bucket refinement algorithm for a high-resolution resource-constrained project scheduling problem. Int. Trans. Oper. Res. (2017). https://doi.org/10.1111/itor.12445, available online
14. Ruthmair, M.: On solving constrained tree problems and an adaptive layers framework. Ph.D. thesis, TU Wien, Vienna, Austria (2012)
15. Ruthmair, M., Raidl, G.R.: A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems. In: Günlük, O., Wöginger, G.J. (eds.) Integer Programming and Combinatoral Optimization. IPCO 2011. LNCS, vol. 6655, pp. 376–388. Springer, Berlin, Heidelberg (2011)
16. Wang, X., Regan, A.C.: Local truckload pickup and delivery with hard time window constraints. Transportation Res. B **36**(2), 97–112 (2002)