otherwise unrestricted. Observe that in (1e) that $(t_u^+ + t_{vu})(1 - x_{vu})$ operates as a big-M term, which makes the inequality inactive when $x_{vu} = 0$.

In (1f) we enforce that a minimum number of vehicles exit the depot; equal to a lower bound on the number of vehicles required to service all demand. This lower bound is the sum of the demands divided by the vehicle capacity, then rounded up to the nearest integer. This is an optional constraint that is not common in the CG literature.

It is well know that this formulation can lead to weak LP bounds because of the following issues. (1) It does not explicitly eliminate sub-tours in the LP solution. (2) Capacity and time feasibility are not well enforced.

4 A Tighter Compact LP Relaxation

In this section, we introduce our compact formulation for VRPTW, which is, in our experiments, tighter than the two-index compact formulation in (1). This formulation is parameterized by sets constructed later in the document (see Section 5). Specifically, our formulation is parameterized by unique values of the following for each customer: **(1)** set of Local Area (LA) neighbors which limit the class of cycles permitted in LP solutions [Mandal et al., 2022] (which we discuss in Section 4.1), **(2)** a discretization of time [Boland et al., 2017], and **(3)** a discretization of capacity. Increasing the number of LA-neighbors of customers can tighten the LP relaxation by limiting the cycles (over customers localized in space) permitted in an LP solution, a principle also exploited by NG-routes [Baldacci et al., 2011]. Discretization is parameterized by grouping capacity remaining (or time remaining into buckets). For example given buckets associated with customer u denoted D_u we bin in the capacity remaining as follows: $D_u = [[d_u, d_u + 3], [d_u + 4, d_u + 5], [d_u + 6, d_u + 9]...[d_0 - 4, d_0]]$. Increasing the number of buckets makes the buckets smaller in range. We refer to increasing the number of buckets as decreasing granularity while decreasing the number of buckets as increasing granularity. Increasing the number of buckets in the time/capacity discretization sets tightens the LP relaxation by enforcing that routes must be time/capacity feasible more rigorously. However, increasing the number of LA-neighbors and decreasing the granularity of the time/capacity discretization increases the number of variables/constraints in the LP and hence increases the LP computation time. Special selection of these sets (LA neighbors and time/capacity buckets) can ensure both fast LP optimization time, and a tighter LP relaxation relative to the baseline in (1); and thus fast MILP optimization time. Given these sets (collectively called a parameterization) we need only solve the corresponding optimization problem as a MILP using an off-the-shelf solver to obtain an optimal solution to the VRPTW.

We organize this section as follows. In Sections 4.1 and 4.2, we tighten the LP relaxation in (1) using LA-arcs, and time/capacity discretization respectively. In Section 4.3, we produce our novel MILP formulation of VRPTW. In Section 4.4, we consider some properties of sufficient parameterizations, which we define as parameterizations that lead to the tightest bound possible given a maximum number of LA-neighbors per customer. In Section 4.5, we consider some properties of parsimonious parameterizations, which are parameterizations that have the widest time/capacity buckets and smallest LA-neighborhoods as possible, given fixed LP objective, leading to maximally efficient solutions to the LPs at each node of the branch & bound tree of the corresponding MILP.

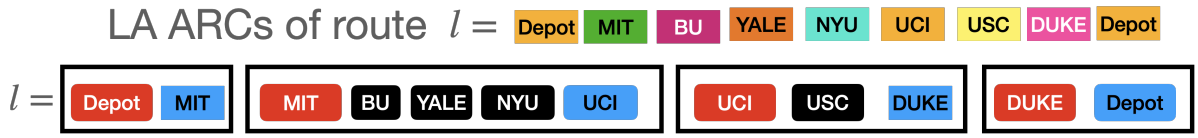

Figure 1: We display a route as a sequence of LA-arcs. Each LA-arc describes a sequence of customers near the first customer in the arc, followed by a distant customer. The example uses American universities and LA-neighborhoods defined by university location. LA-arcs are color-coded as follows. Red indicates the first customer (or depot) in an LA-arc, Black indicates intermediate customers in an LA-arc, and Blue indicates the final customer (or depot) in an LA-arc. The first customer of an LA-arc is shared with the last customer of the previous LA-arc. Observe that the class of cycles permitted in a sequence of LA-arcs is highly limited because individual LA-arcs that make up the route must be elementary.

4.1 Enforcing Elementarity via the Incorporation of Local Area Arcs

In this section, we provide additional variables and associated constraints using Local Area (LA) arcs, which were introduced in the arXiv paper [Mandal et al., 2022]. LA-arcs are designed to eliminate cycles of customers localized in space as described by the x variables used in (1). As a precursor, we provide some notation for describing LA-arcs, which we illustrate in Fig 1. For each $u \in N$ we use $N_u^{\odot}$ to denote the set of customers nearby u, not including u, called the LA-neighbors of u. Specifically $N_u^{\odot}$ is the set of K closest customers to u in space that are reachable from u considering time and capacity; for user-defined K.[1] For ease of notation we define $N_{+u}^{\odot} = \{N_u^{\odot} \cup u\}$. We use $N_u^{\odot\rightarrow}$ to denote the subset of N^+ not in $N_{+u}^{\odot}$ or the starting depot meaning $N_u^{\odot\rightarrow} = N^+ - (N_{+u}^{\odot} \cup \alpha)$. For each $u \in N, v \in N_u^{\odot\rightarrow}, \hat{N} \subseteq N_u^{\odot}$, we use $R^+_{u\hat{N}v}$ to denote the subset of all elementary sequences of customers (called orderings) starting at u; ending at v; including as intermediate customers $\hat{N}$; that are feasible with regard to time/capacity. Not all sets of intermediate customers $\hat{N} \subseteq N_u^{\odot}$ are feasible given u, v due to time/capacity constraints. For any given u, v and set of intermediate customers in $\hat{N} \subseteq N_u$, we define $R_{u\hat{N}v} \subseteq R^+_{u\hat{N}v}$ to contain only those orderings on the efficient frontier with respect to **(a)** travel distance, **(b)** latest feasible departure time from u and **(c)** earliest feasible departure time from v. Feasibility of **(b)**,**(c)** is a consequence of the time windows for the customers. Thus, orderings are preferred with **(a)** lower cost, **(b)** that can be started later, or **(c)** that can be finished earlier. In practice, $R_{u\hat{N}v}$ is a small subset of all possible orderings $R^+_{u\hat{N}v}$ as most do not lie on the efficient frontier. We use R_u to denote the union of $R_{u\hat{N}v}$ over all $v, \hat{N}$ where we clip off the final term in the ordering (meaning v). Observe that any optimal integer solution to VRPTW must select a route containing an ordering in R_u followed immediately by some $v \in N_u^{\odot\rightarrow}$ for each $u \in N$. In practice, R_u is smaller than the sum of the sizes of the $R_{u\hat{N}v}$ terms as many terms are replicated between $R_{u\hat{N}v}$ sets. We provide a dynamic programming approach for generating R_u in Section 6.

We describe an ordering $r \in R_u$ as follows. For any $w \in N_{+u}^{\odot}, v \in N_u^{\odot}$ we define $a_{wvr} = 1$ if w immediately precedes v in ordering r, and otherwise define $a_{wvr} = 0$. For any $w \in N_{+u}^{\odot}$ we define $a_{w*r} = 1$ if w is the final customer in ordering r, and otherwise define $a_{w*r} = 0$.

We use decision variables $y_r \in \mathbb{R}_{0+}\ \forall r \in R_u,\ u \in N$ to describe a selection of orderings to use in our solution. We set $y_r = 1$ for $r \in R_u$ to indicate that after servicing customer u the sequence of customers described by ordering r is used after, which a member of $N_u^{\odot\rightarrow}$ follows immediately, and otherwise set $y_r = 0$. We use $E_u^{*\odot} \subseteq E^*$ to denote the set of edges that can be contained in an ordering starting at u; thus $E_u^{*\odot} = \{wv \in E^*, \text{ s.t. } w \in N_{+u}^{\odot}, v \in N_u^{\odot} - w\}$. We use $E_{uw}^{*\odot}$ to denote the set of edges in E^* that can succeed the final customer w in an ordering starting at u; thus $E_{uw}^{*\odot} = \{wv \in E^*; w \in N_{+u}^{\odot}, v \in N_u^{\odot\rightarrow}\}$.

We now add the following equations to (1), which adds minimization over y to enforce that the solution

[1] Other definitions can be used such as excluding from $N_u^{\odot}$ customers for which the route ordered as follows $\alpha, u, v, u, \bar{\alpha}$ is infeasible by time/capacity.

to the x variables is consistent with the solution y, for which we provide an exposition below the equations.

$$\sum_{r \in R_u} y_r = 1 \qquad \forall u \in N \tag{2a}$$

$$x_{wv} \geq \sum_{r \in R_u} a_{w,v,r} y_r \qquad \forall u \in N, wv \in E_u^{\odot *} \tag{2b}$$

$$\sum_{wv \in E_{uw}^{*\odot}} x_{wv} \geq \sum_{r \in R_u} a_{w*r} y_r \qquad \forall u \in N, w \in N_{+u}^{\odot} \tag{2c}$$

In (2a) we enforce that one ordering in R_u is selected to describe the activities succeeding u for each $u \in N$. In (2b) we enforce that if an ordering $r \in R_u$ is selected in which w is immediately followed by v; then it must be the case that x_{wv} is selected meaning that w is succeeded by v in the solution over x. In (2c) we enforce that if an ordering $r \in R_u$ is selected in which w is the final customer in the ordering then, w must be followed by a customer (or depot) that is not an LA-neighbor of u (and not u) as described by x. Observe that if x is binary then y must be binary as well. The terms y_r are not present in the objective for optimization. The use of LA-arcs to remove cycles, is a key contribution as it allows the user to grow the size of the formulation as the computational capabilities of MILP solvers improve over time.

It is possible to show (see Section 4.5.2) that the number of LA-neighbors required can be less than that of $N_u^{\odot}$ for some u. To this end, we introduce $N_u \subseteq N_u^{\odot}$ to denote this reduced set. We use N_{+u} to denote $N_u \cup u$. We use $N_u^{\rightarrow}$ to denoted the subset of N^+ that does not lie in $N_{+u} \cup \alpha$. We use $E_u^* \subseteq E^*$ to denote the set of edges that can be contained in an ordering starting at u, thus $E_u^* = \{wv \in E^*, \text{ s.t. } w \in N_{+u}, v \in N_u - w, \}$. We use E_{uw}^* to denote the set of edges in E^* that can succeed the final customer w in an ordering starting at u, thus $E_{uw}^* = \{wv \in E^*; v \in N_u^{\rightarrow}\}$. We write (2) using LA-neighbors N_u below.

$$\sum_{r \in R_u} y_r = 1 \qquad \forall u \in N \tag{3a}$$

$$x_{wv} \geq \sum_{r \in R_u} a_{w,v,r} y_r \qquad \forall u \in N, wv \in E_u^* \tag{3b}$$

$$\sum_{wv \in E_{uw}^*} x_{wv} \geq \sum_{r \in R_u} a_{w*r} y_r \qquad \forall u \in N, w \in N_{+u} \tag{3c}$$

Note that R_u is determined by $N_u^{\odot}$ not N_u; while the set of constraints enforced in (3b)-(3c) is defined by N_u. Observe that this can create redundant y terms. These are simply removed before optimization.

4.2 Capacity/Time Discretization Discovery

This section considers using capacity/time discretization to tighten the LP relaxation in (1). As mentioned earlier, this discretization is inspired by the work of Boland et al., 2017, 2019. Earlier discretization methods can be found in Appelgren, 1969, 1971, Wang and Regan, 2002.

Consider that we have a partition of capacity associated with each customer u denoted D_u into continuous "buckets," e.g. $D_u = [[d_u, d_u + 3], [d_u + 4, d_u + 5], [d_u + 6, d_u + 9]...[d_0 - 4, d_0]]$. We index the buckets in D_u from smallest to greatest remaining demand with k; where $k = 1$ is the bucket that contains d_u and $k = |D_u|$ contains d_0. The lower/upper bound values of the $k'th$ bucket for customer u are referred to as $d_{(u,k)}^-, d_{(u,k)}^+$ respectively.

Consider a directed unweighted graph G^D (which we also use for node-set) with edge set E^D. For each $u \in N$, $k \in D_u$ we create one node $i = (u, k)$ for which we alternatively write as (u_i, d_i^-, d_i^+) where $u_i \leftarrow u, d_i^- \leftarrow d_k^-, d_i^+ \leftarrow d_k^+$. There is a single node (α, d_0, d_0) also denoted $(\alpha, 1)$ associated with the starting depot and a single node $(\bar{\alpha}, 0, d_0)$ also denoted $(\bar{\alpha}, 1)$ associated with the ending depot. We now define the edges E^D.

- For each u we create a directed edge from $(\alpha, 1)$ to $(u, |D_u|)$. Traversing this edge indicates that a route starts with u as its first customer.
- For each u we create a directed edge from $(u, 1)$ to $(\bar{\alpha}, 1)$. Traversing this edge indicates that a route ends after having u as its final customer.

- For each $i = (u, k)$ for $k > 1$ we connect i to $j = (u, k-1)$. Traversing this edge indicates that the vehicle servicing u will not use all possible capacity. These edges can be thought of as dumping extra capacity.
- For each pair of nodes $i = (u, k), j = (v, m)$ we connect i to j if the following are satisfied: $u \neq v$, $d_i^+ - d_u \geq d_j^-$ and if $k > 1$ we also require that $d_j^- > d_{(u,k-1)}^+ - d_u$. Traversing this arc indicates that a vehicle has between d_i^- and d_i^+ units of capacity remaining before servicing u and, upon servicing, u travels directly to v where it has between d_j^- and d_j^+ units of capacity remaining before servicing v. The second condition $d_j^- > d_{(u,k-1)}^+ - d_u$ ensures that redundant edges are not generated.

We now describe flow formulation on the discretized capacity. This flow governs the amount of the resource capacity available at a given node. The following constraints are written below using decision variable $z_{ij}^D \in \mathbb{R}_{0+}$ to describe the edges traversed .

$$\sum_{ij \in E^D} z_{ij}^D = \sum_{ji \in E^D} z_{ji}^D \quad \forall i \in G^D; i = (u, k); u \notin [\alpha, \bar{\alpha}] \tag{4a}$$

$$x_{uv} = \sum_{\substack{ij \in E^D \\ i=(u,k) \\ j=(v,m)}} z_{ij}^D \quad \forall uv \in E^* \tag{4b}$$

In (4a), we enforce that the solution z^D is feasible with regard to capacity by enforcing that the flow incoming to a node is equal to the flow outgoing. In (4b), we enforce that the solution x is consistent with the solution capacity graph z^D.

We now describe flow formulation on discretized time as we previously did for capacity. Time buckets and associated graph and edge sets are denoted T_u, G^T, E^T respectively. The constraints are defined analogously to capacity (with minimum/maximum values of t_u^- and t_u^+ respectively in place of d_u and d_0). We write the corresponding constraints below. We use non-negative decision variables z_{ij}^T to describe the movement of vehicles. We set $z_{ij}^T = 1$ to indicate that a vehicle leaves u_i with time remaining in between t_i^- and t_i^+ and leaves at u_j with time remaining between t_j^- and t_j^+. We write the constraints for time analogous to (4a) and (4b) below.

$$\sum_{ij \in E^T} z_{ij}^T = \sum_{ji \in E^T} z_{ji}^T \quad \forall i \in G^T; i = (u, k); u \notin [\alpha, \bar{\alpha}] \tag{5a}$$

$$x_{uv} = \sum_{\substack{ij \in E^T \\ i=(u,k) \\ j=(v,m)}} z_{ij}^T \quad \forall uv \in E^* \tag{5b}$$

We refer to the finest possible granularity of capacity and time buckets for each $u \in N$ as $D_u^{\odot}, T_u^{\odot}$.

4.3 Complete Linear Programming Relaxation

Given a set of LA-neighbors, time buckets, and capacity buckets for each $u \in N$ denoted N_u, T_u, D_u respectively, we write the MILP enforcing no-localized cycles in space (via (3)), capacity bucket feasibility (via (4)), and time bucket feasibility (via (5)) as $\bar{\Psi}(\{N_u, T_u, D_u\}, \forall u \in N)$ below, with its LP relaxation denoted $\Psi(\{N_u, T_u, D_u\}, \forall u \in N)$.

$$\bar{\Psi}(\{N_u, T_u, D_u\}, \forall u \in N) = \min_{\substack{x\in\{0,1\} \\ y\geq 0 \\ z\geq 0 \\ t_u^+ \geq \tau_u \geq t_u^- \\ d_0 \geq \delta_u \geq d_u}} \sum_{\substack{u\in N^+ \\ v\in N^+}} c_{uv} x_{uv} \qquad \text{Original Compact} \qquad (6a)$$

$$\sum_{uv\in E^*} x_{uv} = 1 \quad \forall u \in N \qquad \text{Original Compact} \qquad (6b)$$

$$\sum_{vu\in E^*} x_{vu} = 1 \quad \forall u \in N \qquad \text{Original Compact} \qquad (6c)$$

$$\delta_v - d_v \geq \delta_u - (d_0 + d_v)(1 - x_{vu}) \quad \forall u \in N, vu \in E^* \qquad \text{Original Compact} \qquad (6d)$$

$$\tau_v - t_{vu} \geq \tau_u - (t_u^+ + t_{vu})(1 - x_{vu}) \quad \forall u \in N, vu \in E^* \qquad \text{Original Compact} \qquad (6e)$$

$$\sum_{u\in N} x_{\alpha u} \geq \lceil \frac{\sum_{u\in N} d_u}{d_0} \rceil \qquad \text{Original Compact} \qquad (6f)$$

$$\sum_{r\in R_u} y_r = 1 \quad \forall u \in N \qquad \text{LA-arc Movement Consistency} \qquad (6g)$$

$$x_{wv} \geq \sum_{r\in R_u} a_{w,v,r} y_r \quad \forall u \in N, wv \in E_u^* \qquad \text{LA-arc Movement Consistency} \qquad (6h)$$

$$\sum_{wv\in E_{uw}^*} x_{wv} \geq \sum_{r\in R_u} a_{w*r} y_r \quad \forall u \in N, w \in N_{+u} \qquad \text{LA-arc Movement Consistency} \qquad (6i)$$

$$\sum_{ij\in E^D} z_{ij}^D = \sum_{ji\in E^D} z_{ji}^D \quad \forall i \in G^D; i = (u,k); u \notin [\alpha, \bar{\alpha}] \qquad \text{Flow Graph Capacity} \qquad (6j)$$

$$x_{uv} = \sum_{\substack{ij\in E^D \\ i=(u,k) \\ j=(v,m)}} z_{ij}^D \quad \forall uv \in E^* \qquad \text{Flow Graph Capacity} \qquad (6k)$$

$$\sum_{ij\in E^T} z_{ij}^T = \sum_{ji\in E^T} z_{ji}^T \quad \forall i \in G^T; i = (u,k); u \notin [\alpha, \bar{\alpha}] \qquad \text{Flow Graph Time} \qquad (6l)$$

$$x_{uv} = \sum_{\substack{ij\in E^T \\ i=(u,k) \\ j=(v,m)}} z_{ji}^T \quad \forall uv \in E^* \qquad \text{Flow Graph Time} \qquad (6m)$$

To be able to use an off-the-shelf MILP solver to solve for (6) efficiently, we must have parameterization $(\{N_u, T_u, D_u\}\forall u \in N)$ that is as tight as possible and small as possible in terms of using fewer variables and constraints. Since the tightest LP expressible is $\Psi(\{N_u^\odot, T_u^\odot, D_u^\odot\}\forall u \in N)$ we want $\Psi(\{N_u, T_u, D_u\}\forall u \in N)$ to be equal to $\Psi(\{N_u^\odot, T_u^\odot, D_u^\odot\}\forall u \in N)$ and refer to such parameterizations as sufficient.

We want the LP $\Psi(\{N_u, T_u, D_u\}\forall u \in N)$ and hence each node in the MILP $\bar{\Psi}(\{N_u, T_u, D_u\}\forall u \in N)$ to be easily solved. One key condition for such parameterizations is that of being parsimonious; any further contraction of the sets in the parameterization decreases the LP value of the optimal solution. Another way to say this is that the parameterization has the widest buckets and smallest LA-neighborhoods possible, given fixed LP tightness, leading to maximally efficient solutions to the LPs at each node of the branch & bound tree of the corresponding MILP. We refer to LPs that contain these two properties that as sufficient and parsimonious parameterization (SPP) and in Section 5 consider the construction of such LPs.

4.4 Properties of Sufficient Parameterizations

We now consider some sufficient (but not always necessary) conditions of sufficient parameterizations. To this end we consider a candidate optimal LP solution (x^*, y^*, z^*) given parameterization $(\{N_u^*, T_u^*, D_u^*\}\forall u \in N)$.

The property of bucket feasibility applies to capacity and time. This provides a condition for the LP solution that, if satisfied, guarantees that using the minimum granularity (minimum bucket sizes) of capacity would not tighten the bound given the current LA-neighbors. We require bucket feasibility to hold in our

conditions for sufficient parameterizations. Bucket feasibility states no relaxation of capacity/time occurs as a consequence of using the current bucket parameters. We formally define bucket feasibility for capacity and time below.

$$\textbf{Capacity Bucket Feasibility } [z^{D*}_{ij} > 0] \rightarrow [d^+_i - d_{u_i} = d^+_j] \quad \forall (i,j) \in E^D \; u_j \notin \{u_i, \bar{\alpha}\} \tag{7a}$$

$$\textbf{Time Bucket Feasibility } [z^{T*}_{ij} > 0] \rightarrow [\min(t^+_i - t_{u_i,u_j}, t^+_j) = t^+_j] \quad \forall (i,j) \in E^T; u_j \notin \{u_i, \bar{\alpha}\} \tag{7b}$$

Observe that if (7) is satisfied, then no relaxation of capacity/time occurs as a consequence of using the buckets, hence using the finest capacity/time discretization $\{D^{\odot}, T^{\odot}\}$ would not tighten the bound given the current LA-neighbors. We prove this formally in Appendix C.

The property of LA-arc feasibility is a condition for the LP solution that, if satisfied, guarantees that using the $N^{\odot}_u$ for the LA-neighbors of each customer would not tighten the bound given the current time/capacity buckets. Specifically, LA-arc feasibility states that given fixed x^*, a setting for y obeys (2), not merely (3). We require LA-arc feasibility to hold in our conditions for sufficient parameterizations.

4.5 Properties of Parsimonious Parameterizatons

We now consider some necessary but not always sufficient properties of parsimonious parameterizations.

4.5.1 Capacity/Time Parsimony

Consider the dual solution to the LP in (6), which we denote π where π^D_i is the dual value associated with (6j) for a given i. We write the reduced cost of z^D_{ij} as $\bar{z}^D_{ij}$ for any i, j s.t. $u_i = u_j$. Observe that $\bar{z}^D_{ij} = -\pi^D_i + \pi^D_j$.

Now observe that merging capacity buckets i, j corresponds to enforcing that $\pi^D_i = \pi^D_j$. Here merging $i = (u_i, k+1)$ with $j = (u_i, k)$ corresponds to removing buckets corresponding to $(u_i, k+1)$ and (u_i, k) and replacing them with a bucket associated with u_i and range of capacity $[d^-_{(u_i,k)}, d^+_{(u_i,k+1)}]$. When $\pi^D_i = \pi^D_j$ for $u_i = u_j$ and $i, j \in E^D$ is satisfied, then observe that merging capacity buckets associated with i, j together does not weaken the LP relaxation in (6). However, it does decrease the number of constraints/variables in (6), leading to faster optimization of the LP.

This same logic holds for time buckets. Let π^T_i denote the dual variable associated with (6l). We write the reduced cost for variable z^T_{ij}, which we denote as $\bar{z}^T_{ij}$ as follows $\bar{z}^T_{ij} = -\pi^T_i + \pi^T_j$ for any $z^T_{i,j}$ s.t. $u_i = u_j$ and $i, j \in E^T$. Thus, when $\pi^T_i = \pi^T_j$ for $u_i = u_j$ observe that we can merge the time buckets i, j without loosening the bound in (6). However, as is the case for capacity buckets, this decreases the number of constraints/variables in (6), leading to faster optimization of the LP.

4.5.2 LA Neighborhood Parsimony

Let us now consider the construction of parsimonious LA-neighborhoods to enforce in (6). Let us consider the following additional notation. Let N^k_u be the k closest LA-neighbors in N_u to u (in terms of distance). We use N^k_{+u} to denote $N^k_u \cup u$. Here k ranges from 0 to $|N_u|$.

For any ordering r where N_r are the customers in the order r and $w \in N_r$ we define $a^k_{wr} = 1$ if all customer in r prior to and including w lie in N^k_{+u} and otherwise set $a^k_{wr} = 0$.

For each $u \in N, r \in R_u, w \in N^k_{+u}, v \in N^k_u - w, k$ we define $a^k_{wvr} = 1$ if $a_{wvr} = 1$ and $a^k_{vr} = 1$. For each $u, r \in R_u, w \in N^k_{+u}$ we define $a^k_{w*r} = 1$ if $a^k_{wr} = 1$ and either ($a_{w*r} = 1$ or $\sum_{v \notin N^k_u - w} a_{wvr} = 1$) ; and otherwise set $a^k_{w*r} = 0$.

We now replace (6h) and (6i) with the following inequalities parameterized by tiny positive value ϵ and dual variables noted in brackets. We use slack constant ϵ which we multiply by the number of LA-neighbors associated with a given constraint in order to encourage the LP solver to use constraints corresponding to as few LA-neighbors as possible. Later this facilitates the construction of a more parsimonious LP as larger LA neighborhoods are not be used in the generated LP if the dual variables are not active.

$$\epsilon k + x_{wv} \geq \sum_{r \in R_u} a^k_{w,v,r} y_r \quad \forall u \in N, w \in N^k_{+u}, k \in [1,2,...|N_u|], v \in N^k_u, wv \in E^* \quad [-\pi_{uwvk}] \tag{8a}$$

$$\epsilon k + \sum_{\substack{v \notin N^+ - (N^k_{+u}) \\ wv \in E^*}} x_{wv} \geq \sum_{r \in R_u} a^k_{w*r} y_r \quad \forall u \in N, k \in [1,2,...|N_u|], w \in N^k_{+u} \quad [-\pi_{uw*k}] \tag{8b}$$

We now solve the LP form in (6) replacing (6h) and (6i) with (8a) and (8b) respectively. We refer to the associated LP as $\Psi^*(\{N_u, T_u, D_u\} \forall u \in N)$

Observe that the dual LP for $\Psi^*(\{N_u, T_u, D_u\} \forall u \in N)$ has a penalty for using dual variables of primal constraints associated with larger LA-neighborhoods than necessary. Consider any u, and let k_u be the largest neighborhood size containing a dual variable of the form (8a) or (8b) with positive value. We write k_u formally below.

$$k_u \leftarrow \max_{|N_u| \geq k \geq 1} k[0 < ((\sum_{w \in N_u} \pi_{uwk}) + (\sum_{\substack{w \in N^k_{+u} \\ v \in N^k_u - w \\ wv \in E^*}} \pi_{uwvk}))] \quad \forall u \in N \tag{9}$$

If $k_u < |N_u|$, then we can decrease the size of N_u to k_u without loosening the bound. This decreases the number of constraints in (6) of the forms (6h), and (6i). This also decreases the number of variables that need to be considered as some of the y_r variables become redundant. This, in turn, leads to the faster resolution of the associated LP. Observe that given u, if no dual variables of the form (8) are positively valued, then setting N_u to be empty does not loosen the bound.

5 LA-Discretization: An Algorithm for a Sufficient and Parsimonious Parameterization Construction

In this section, we consider the construction of a sufficient and parsimonious parameterization (SPP). We construct an SPP by iterating between the following two steps: **(1)** Solving $\Psi^*(\{N_u, T_u, D_u\} \forall u \in N)$ and **(2)** Augmenting $\Psi^*(\{N_u, T_u, D_u\}$ to achieve sufficiency. Whenever **(1)** tightens the bound (with respect to the previous iteration), we attempt to achieve parsimony by decreasing the size of some terms in $\{N_u, T_u, D_u\}$ for some customers. This decrease is always done not to loosen the bound but to achieve the specific necessary conditions for parsimony. We only aim towards parsimony after improving the bound (not at every step) to avoid cycling in the LP.

Thresholds determine the buckets, which we denote as W^D_u, W^T_u (sorted from smallest to largest) for capacity and time for customer u, respectively, where the thresholds determine the maximum value of the buckets. For example if the thresholds for W^D_u are $[d_u + 5, d_u + 12]$ then the associated buckets are $[d_u, d_u + 5], [d_u + 6, d_u + 12], [d_u + 13, d_0]$.

We organize this section as follows. In Section 5.1, we consider the contraction and expansion operations for time/capacity buckets, while Section 5.2 does the same for the number of LA-neighbors. In Section 5.3, we describe our complete optimization algorithm, which we call LA-Discretization.

5.1 Contracting/Expanding Capacity/Time Buckets

Recall from Section 4.5.1 that for any $i, j \in E^D$ for which $u_i = u_j$ if $\pi^D_i = \pi^D_j$ then we can merge nodes i, j without loosening the bound and remove d^+_j from $W^D_{u_j}$. The same logic applies to time. For any $i, j \in E^T$ for which $u_i = u_j$ if $\pi^T_i = \pi^T_j$, we can merge nodes i, j without loosening the bound and remove t^+_j from $W^T_{u_j}$.

To enforce bucket feasibility with respect to capacity we do the following. For each i, j s.t. $z^D_{ij} > 0$ where $i = (u_i, d^-_i, d^+_i)$ and $j = (u_j, d^-_j, d^+_j)$ and $j \neq \bar{\alpha}$ and $u_i \neq u_j$: We add to $W^D_{u_j}$ the term $d^+_i - d_{u_i}$ (if this term is not already present in $W^D_{u_j}$). Observe that this operation can never loosen the relaxation as undoing it simply corresponds to requiring that $\pi^D_i = \pi^D_j$ in the dual.

The same approach applies to enforcing bucket feasibility with respect to time. For each i, j s.t. $z^D_{ij} > 0$ where $i = (u_i, t^-_i, t^+_i)$ and $j = (u_j, t^-_j, t^+_j)$ and $j \neq \bar{\alpha}$: We add to $W^T_{u_j}$ the term $\min(t^+_i - t_{u_i,u_j}, t^+_j)$ (if this

term is not already present in $W^T_{u_j}$). Observe that this operation can never loosen the relaxation as undoing it simply corresponds to requiring that $\pi^T_i = \pi^T_j$ in the dual.

5.2 Contracting/Expanding Local Area Neighborhoods

Recall from (9) that we can set the number of LA-neighbors to the smallest number for which there is an associated non-zero dual value without loosening the bound. Thus our contraction operation on LA-neighborhoods is written as follows.

$$|N_u| \leftarrow \max_{|N_u| \geq k \geq 1} k[0 < ((\sum_{w \in N_u} \pi_{uwk}) + (\sum_{\substack{w \in N^k_{+u} \\ v \in N^k_u - w \\ wv \in E^*}} \pi_{uwvk}))] \quad \forall u \in N \tag{10}$$

To expand the neighborhoods, we can iterate over $u \in N$, then, for each u, identify the smallest number of LA-neighbors required to induce a violation of the current LP. To determine if a given number of LA-neighbors k induces a violation of the current LP we simply check to see if a fractional solution y_r exists to satisfy (6i), (6g) and (6h) where we fix the x terms to their current values and N_u to N^k_u.

A simpler approach is as follows. We can periodically simply reset the number of LA-neighbors to the maximum number $|N^{\odot}_u|$ for each $u \in N$. Next, we solve the LP and apply a contraction operation described in (10). Our experiments employ this approach when expansion operations on capacity/time buckets have not recently tightened the LP bound.

5.3 Complete Algorithm: LA-Discretization

We now consider the construction of an SPP by iteratively solving the LP relaxation and tightening the LP relaxation to aim toward sufficiency; each time we tighten the relaxation, we decrease the size of the parameterization with parsimony. We provide our algorithm in Alg 1 with exposition below.

- Line 1: Initialize the LA-neighbor sets, time buckets and capacity buckets. In our implementation We initialize $N_u \leftarrow N^{\odot}_u$ for all $u \in N$. We initialize W^D_u parameterized by a bucket size $d^s = 5$ where

Algorithm 1 LA-Discretization Algorithm

1: $\{N^{\odot}_u, N_u, W^D_u, W^T_u \quad \forall u \in N\} \leftarrow$ from User
2: iter_since_reset$\leftarrow 0$
3: last_lp_val$\leftarrow -\infty$
4: **repeat**
5: **if** iter_since_reset $\geq \zeta$ **then**:
6: $N_u \leftarrow N^{\odot}_u \quad \forall u \in N$
7: **end if**
8: $[z, y, x, \pi,$lp_objective$] \leftarrow$ Solve $\Psi^*(\{N_u, T_u, D_u\} \forall u \in N)$
9: **if** lp_objective>last_lp_val+MIN_INC **then**
10: $W^D_u \leftarrow$ Merge any nodes i, j for which $i, j \in E^D, u_i = u_j = u$ and $\pi^D_i = \pi^D_j$ for all $u \in N$
11: $W^T_u \leftarrow$ Merge any nodes i, j for which $i, j \in E^T, u_i = u_j = u$ and $\pi^T_i = \pi^T_j$ for all $u \in N$
12: $\{N_u \quad \forall u \in N\} \leftarrow$ Apply (10) to contract LA-neighbors.
13: last_lp_val$\leftarrow$ lp_objective
14: iter_since_reset $\leftarrow 0$
15: **end if**
16: $W^T_u \leftarrow W^T_u \cup_{ij \in E^T; u_j = u; z^T_{ij} > 0} \min(t^+_i - t_{u_i,u_j}, t^+_j)$ for all $u \in N$
17: $W^D_u \leftarrow W^D_u \cup_{ij \in E^D; u_j = u; z^D_{ij} > 0} (d^+_i - d_{u_i})$ for all $u \in N$
18: iter_since_reset ++
19: **until** $\{N_u, W^D_u, W^T_u \quad \forall u \in N\}$ are unchanged from the previous iteration OR iter_since_reset>ITER_MAX
20: $W^T_u, W^D_u, N_u \leftarrow$ via contraction operations from Lines 10-12
21: $x \leftarrow$Solve $\bar{\Psi}(\{N_u, T_u, D_u\} \forall u \in N)$ as MILP (only x is integer)

$W_u^D = \{[d_u, d_u + d^s), [d_u + d^s, d_u + 2d^s)...)$. The last bucket may be smaller than d^s . We initialize W_u^T parameterized by a bucket size $t^s = 50$ where $W_u^T = \{[t_u^-, t_u^- + t^s), [t_u^- + t^s, t_u^- + 2t^s)...)$. The last bucket may be smaller than t^s. We define the number of elements in $N_u^{\odot}$ for each $u \in N$ equal to user defined parameter n^s where $n^s = 6$ in our experiments.

- Line 2-3: We set the number of iterations since the last contraction operations, which we denote as iter_since_reset to zero. We also set the incumbent LP value denoted last_lp_val LP objective to $-\infty$. This value last_lp_val is the LP value immediately after the most recent contraction operation which is $-\infty$ since no contractions have yet happened.
- Line 4-19: Construct a sufficient parameterization.
  - Line 5-Line 7: If we have not tightened the bound by user defined parameter MIN_INC (MIN_INC=1 in experiments) for a given amount of iterations (denoted ζ), then we aim towards sufficiency by increasing the LA-neighborhoods of all customers to their maximum size. We set $\zeta = 9$ in our experiments.
  - Line 8: Solve the LP given the current parameterization.
  - Line 9-15: If we have improved the LP relaxation by MIN_INC since the last contraction operation, we apply contraction operations to aim toward parsimony for our parameterization.
    - Line 10-12: Apply contraction operations to time/capacity buckets and LA-neighborhoods.
    - Line 13: Update the LP objective immediately after the most recent contraction operation.
    - Line 14: Set the number of iterations since the most recent contraction operation to zero.
  - Line 16,17: Aim towards sufficiency by applying expansion operations to capacity and time buckets, respectively.
  - Line 18: Increment iter_since_reset.
  - Line 19: Terminate optimization when sufficiency is achieved, or progress towards sufficiency has ceased. We stop the algorithm after ITER_MAX(=10 in experiments) consecutive iterations of failing to tighten the LP relaxation by a minimum of MIN_INC (=1 in experiments) combined. In Section D we prove that if ITER_MAX=∞ then Alg 1 (Lines 4-19) produces a sufficient parameterization.
- Lines 20: Decrease the LP size by modifying the LP for parsimony.
- Line 21: Finally solve the appropriate MILP in (6) given our SPP. Note that only x is enforced to be binary, and all other variables are continuous. Enforcing y to be binary does not change the MILP solution objective but may accelerate optimization depending on the specifics of the MILP solver. We enforced y to be binary in experiments.

6 Local Area Arc Computation

In this section we consider the computation of the efficient frontier of orderings $R_{u\hat{N}v}$. We define helper terms P to denote the set of all unique values of $u \in N$,$\hat{N} \subseteq N_u^{\odot}$,$v \in N_u^{\odot\rightarrow}$, which we index by p. Here we describe a specific p as (u_p, N_p, v_p). We organize this section as follows. In Section 6.1, we provide notation needed to express the material in this section. In Section 6.2, we describe a recursive relationship for the cost of an LA-arc p given a departure times from u_p, v_p. In Section 6.3, we introduce the concept of an efficient frontier of orderings for each $p \in P$ trading off latest feasible departure time from u_p (later is preferred); earliest departure time from u_p when no waiting is required (earlier is preferred); and cost (lower cost is preferred). In Section 6.4, we provide a dynamic programming based algorithm to compute the efficient frontier for all LA-arcs jointly. In Section B, we provide a proof referenced in Section 6.2.

6.1 Notation for LA-Arcs Modeling Time

Let P^+ be a super-set of P defined as follows. Here $p = (u_p, N_p, v_p)$ for $u_p \in N, N_p \subseteq N, v_p \in N^+$ lies in P^+ if and only if all of the following conditions hold. There exists a $u \in N$ s.t. $u_p \in (u \cup N_u)$; $v_p \in N_u^{\odot\rightarrow}$; $N_p \subseteq N_u - (u_p + v_p)$. We use helper term c_{p,t_1,t_2} to denote the cost of lowest cost feasible ordering departing u_p

at time t_1, departing v_p at t_2, visiting N_p in between; where r_{p,t_1,t_2} denotes the associated ordering. We define $R_{u\hat{N}v}$ as the union of $r \in R^+_{u\hat{N}v}$ for which $r = r_{p,t_1,t_2}$ for some t_1, t_2. We use R^+_p, R_p as short hand for $R^+_{u_pN_pv_p}$ and $R_{u_pN_pv_p}$ respectively. We describe r as an ordered sequence listed from in chronological order of visits as follows $[u^r_1, u^r_2, u^r_3, ...v^r]$ with the final element denoted v^r. We define an ordering r^-, to be the same as r except with u^r_1 removed, meaning $r^- = [u^r_2, u^r_3, ...v^r]$.

For any given time t and ordering $r \in R^+_p$, we define $T_r(t)$ as the earliest time a vehicle could depart v^r, if that vehicle departs u^r_1 with t time remaining and follows the ordering of r. We write $T_r(t)$ mathematically below using $-\infty$ to indicate infeasibility, and [] to denote the binary indicator function.

$$T_r(t) = T_{r^-}(-t_{uw} + \min(t, t^+_u)) - \infty[t < t^-_u] \quad \forall r \in R^+_p, p \in P^+, u^r_1 = u, w = u^r_2 \tag{11}$$

Given $T_r(t)$ we express c_{p,t_1,t_2} as follows.

$$c_{pt_1t_2} = \min_{\substack{r \in R^+_p \\ T_r(t_1) \geq t_2}} c_r \quad \forall p \in P^+, t_1, t_2 \tag{12}$$

Here $c_{pt_1t_2} = \infty$ if no r exists satisfying $T_r(t_1) \geq t_2$. Observe that using (12) to evaluate $c_{pt_1t_2}$ is challenging as we would have to repeatedly evaluate $T_r(t)$ in a nested manner.

6.2 Recursive Definition of Departure Time

In this section we provide an alternative characterization of $T_r(t)$, that later provides us with a mechanism to efficiently evaluate $c_{pt_1t_2}$. For any $r \in R^+_p, p \in P^+$ we describe $T_r(t)$ using the helper terms $\phi_r, \hat{\phi}_r$ defined as follows. We define ϕ_r to be the earliest time that a vehicle could leave u^r_1 without waiting at any customer in $u_p \cup N_p \cup v_p$ if t^- terms were ignored (meaning all t^- terms are set to $-\infty$). Similarly we define $\hat{\phi}_r$ as the latest time a vehicle could leave u^r_1 if t^+ terms were ignored (meaning all t^+ terms are set to ∞). Below we define $\phi_r, \hat{\phi}_r$ recursively.

$$\phi_r = \min(t^+_u, t^+_v + t_{uv}) \quad \forall r \in R^+_p, p = (u, \{\}, v), p \in P^+ \tag{13a}$$

$$\hat{\phi}_r = \max(t^-_u, t^-_v + t_{uv}) \quad \forall r \in R^+_p, p = (u, \{\}, v), p \in P^+ \tag{13b}$$

$$\phi_r = \min(t^+_u, \phi_{r^-} + t_{uw}) \quad \forall r \in R^+_p, u^r_1 = u, u^r_2 = w, |N_p| > 0, p \in P^+ \tag{13c}$$

$$\hat{\phi}_r = \max(t^-_u, \hat{\phi}_{r^-} + t_{uw}) \quad \forall r \in R^+_p, u^r_1 = u, u^r_2 = w, |N_p| > 0, p \in P^+ \tag{13d}$$

Let c_r denote the total travel distance on ordering r. We rewrite $T_r(t)$ using $\phi_r, \hat{\phi}_r$, with intuition provided subsequently (with proof of equivalence in Appendix B).

$$T_r(t) = -c_r + \min(t, \phi_r) - \infty[\min(t, t^+_{u_p}) < \hat{\phi}_r] \quad \forall p \in P^+, r \in R^+_p \tag{14}$$

We now provide an intuitive explanation of (14). Observe that leaving u^r_1 after $\hat{\phi}_r$ and following the ordering r is infeasible. Observe that leaving u^r_1 at time t and following ordering r for $t > \phi_r$ incurs a waiting time of $t - \phi_r$ over the course of the ordering, and otherwise incurs a waiting time of zero. Thus we subtract the travel time c_r from ϕ_r to obtain the departure time at v^r. Given R^+_p and (14) we write c_{p,t_1,t_2} below.

$$c_{pt_1t_2} = \min_{\substack{r \in R^+_p \\ t_1 \geq \hat{\phi}_r \\ t_2 \leq -c_r + \min(t_1, \phi_r)}} c_r \quad \forall p \in P^+, t_1, t_2 \tag{15}$$

If no r satisfies the constraints in (15) then $c_{pt_1t_2} = \infty$. Observe that $|R^+_p|$ can grow factorially with $|N_p|$ thus using (15) is impractical when $|N_p|$ grows.

6.3 Efficient Frontier

In this section we describe a sufficient subset of R^+_p denoted R_p s.t. that applying (15) over that subset produces the same result as if all of R^+_p were considered. Given p, a necessary criterion for a given $r \in R^+_p$ to lie in $\in R_p$, is that r is not Pareto dominated by any other such ordering in R^+_p with regards to latest time to start the ordering (later is preferred), cost (lower is preferred), and earliest time to start the ordering

without waiting (earlier is preferred). Thus, we prefer smaller values of $\hat{\phi}_r, c_r$ and larger values of ϕ_r. We use $R_p \subseteq R_p^+$ to denote the efficient frontier of R_p^+ with regards to $\hat{\phi}_r, c_r, \phi_r$. Here for any $r \in R_p^+$ we define r to lie in R_p IFF no $\hat{r} \in R_p^+$ exists for which all of the following inequalities hold, (and at least one holds strictly meaning not with an equality). **(1)**$\hat{\phi}_r \geq \hat{\phi}_{\hat{r}}$. **(2)**$c_r \geq c_{\hat{r}}$. **(3)**$\phi_r \leq \phi_{\hat{r}}$. We use the following stricter criteria for **(3)**: there is no time t when we leave u_p using r in which we could depart v_p before we could depart v_p using $\hat{r}$. This criteria is written as follows $\phi_r - c_r \leq \phi_{\hat{r}} - c_{\hat{r}}$. Observe that given the efficient frontier R_p that we can compute $c_{pt_1t_2}$ as follows.

$$c_{pt_1t_2} = \min_{\substack{r \in R_p \\ t_1 \geq \hat{\phi}_r \\ t_2 \leq -c_r + \min(t_1, \phi_r)}} c_r \quad \forall p \in P^+, t_1, t_2 \tag{16}$$

If no r satisfies the constraints in (16) then $c_{pt_1t_2} = \infty$. Observe that we can not construct R_p by removing elements from R_p^+ as enumerating R_p^+ would be too time intensive.

6.4 An Algorithm to Generate the Efficient Frontier

In this section we provide an efficient algorithm to construct all R_p jointly without explicitly enumerating R_p^+. In order to achieve this we exploit the following observation for any $p \in P^+$ s.t. $|N_p| > 0$. Observe that if $r \in R_p$ then r^- must lie in $R_{\hat{p}}$ where $\hat{p} = (u_{\hat{p}} = u_2^r, N_{\hat{p}} = N_p - u_2^r, v_{\hat{p}} = v^r)$.

We construct R_p terms by iterating over $p \in P^+$ from smallest to largest with regard to $|N_p|$, and constructing R_p using the previously generated efficient frontiers. For a given p we first add u_p in front of all possible r^- (called predecessor orderings) ; then remove all orderings that do not lie on the efficient frontier. The set of predecessor orderings for a given r is written below.

$$\cup_{\substack{w \in N_p \\ \hat{p} = (w, N_p - w, v_p)}} R_{\hat{p}} \tag{17}$$

Observe that the base case where N_p is empty then there is only one member of R_p which is defined by sequence $r = [u_p, v_p]$ when feasible, and otherwise is empty. In Alg 2 we describe the construction of R_p, which we annotate below.

Algorithm 2 Computing the Efficient Frontier for all $p \in P$

```
1:  for p ∈ P+ from smallest to largest in terms of |N_p| with |N_p| > 0 do
2:      R_p ← {}
3:      for w ∈ N_p; p̂ = (w, N_p − w, v_p), r⁻ ∈ R_p̂ do
4:          r ← [u_p, r⁻]
5:          Compute c_r, and φ_r, φ̂_r via (13)
6:          if φ̂_r ≤ t_u^+ then
7:              R_p ← R_p ∪ r
8:          end if
9:      end for
10:     for r ∈ R_p, r̂ ∈ R_p − r do
11:         if c_r ≥ c_r̂ and φ̂_r ≥ φ̂_r̂ and φ_r − c_r ≤ φ_r̂ − c_r̂ and at least one inequality is strict then
12:             R_p ← R_p − r
13:         end if
14:     end for
15: end for
```

- Line 1-15: Iterate over $p \in P^+$ from smallest to largest with regards to $|N_p|$ and given p compute R_p.
  - Line 2: Initialize R_p to be empty.
  - Line 3-9: Compute all possible terms for R_p by adding a customer u_p in front of all potential predecessor r^- terms.
    - Line 4: Add u_p to the front of r^- creating r.
    - Line 5: Compute c_r, and ϕ_r, $\hat{\phi}_r$ via (13).

* Line 6-8: If r is feasible for some departure time t_1 then we add r to R_p.

– Line 10-14: Iterate over members of R_p and remove terms that do not lie on the efficient frontier.

* Line 11-13: If r is dominated by $\hat{r}$ then we remove r from R_p.

We compute R_p for all p using Alg 2 once prior to running Alg 1. To construct R_u we then take the union of all terms in $R_{u\hat{N}v}$ for all $v \in N_p^{\odot\rightarrow}, \hat{N} \subseteq N_u^{\odot}$ then finally clip v from the end.

7 Experiments

In this section, we provide experimental validation for LA-Discretization. We compare LA-Discretization to the baseline optimization of solving the two-index compact MILP in (1). We validated LA-Discretization on the standard Solomon instance data sets [Solomon, 1987] with additional results in Extension E. We provide the maximum computation time for the MILP call of 1000 seconds for each solver. All experiments are run using Gurobi with default parameters, and all codes are implemented in Python. We used a 2022 Apple Macbook Pro with 24 GB of memory and an Apple M2 chip. As is often done in the literature, all distances are rounded down to the nearest tenth.

We fixed the optimization parameters to reasonable values fixed once. These values are shared by all problem instances on all data sets. We set $n^s = 6, d^s = 5, t^s = 50$, ITER_MAX=10, $\zeta = 9$ and MIN_INC=1. We did not find the solver to be sensitive to these parameters.

We provide comparisons across Solomon data sets/problem sizes in Figures 2a-2b (MILP time) and Figures 2c-2d (MILP+ all LPs time) Figures 2e-2f (integrality gap as a percentage). We define the integrality gap to be the difference between the MILP upper bound and the MILP Lower bound which are both produced over the course of running the MILP solver. For Figures 2e-2f we normalize the data as follows. We take integrality gap and divided it by the MILP upper bound then multiply by 100 to get the value to plot for both baseline and our approach. For problems with larger integrality gaps we believe that using valid inequalities to tighten the bound.In Figures 3a-3b we plot the LP integrality gap. We define this as the difference between the MILP upper bound and the LP at the root node of the branch & bound tree. We normalize the data as follows. We take LP integrality gap and divided it by the MILP upper bound then multiply by 100 to get the value to plot for both baseline and our approach. Since a large number of points for the baseline hit the 1000 second time horizon we had issues with visualization. Hence for such instances we add random noise to the baseline value for ILP run time and Total run time in Figs 3c-3f.

We observe that LA-Discretization performs best relative to the baseline on problem instances which are hardest for baseline MILP to solve efficiently.