An Exact Algorithm for a Practical Pickup and Delivery Problem

Lucas Sippel, Michael A. Forbes, Joseph Menesch

School of Mathematics and Physics, The University of Queensland, Brisbane, Australia
uqlsippe@uq.edu.au, m.forbes@uq.edu.au, joseph.menesch@uqconnect.edu.au

Extended abstract submitted for presentation at the 11th Triennial Symposium on Transportation Analysis conference (TRISTAN XI)
June 19-25, 2022, Mauritius Island

April 4, 2022

Keywords: Vehicle routing, Fragments, Learning

1 Introduction

Fragment based methods have been used in Alyasiry *et al.* (2019) and Rist & Forbes (2021) to solve the Pickup and Delivery Problem with Time Windows and the Dial-a-Ride Problem. We propose a fragment based branch-and-cut algorithm for the Practical Pickup and Delivery problem (PPDP) from Xu *et al.* (2003).

Xu *et al.* (2003) state that the PPDP "involves a set of practical complications that are commonly seen in practice but have received little attention in the vehicle routing literature". The most important of these are the existence of multiple time windows per request and the American hours of service (HOS) constraints. These require a driver to have a 10 hour rest minimum after at most 11 hours driving or 14 hours working. Also since Xu *et al.* (2003)'s study was published, another HOS constraint requiring drivers to take a half hour break at most every eight hours has been introduced, which we consider.

Xu *et al.* (2003) use a forward labelling dynamic program (DP) to construct negative reduced cost vehicle routes in a price-and-branch heuristic. However, high dimensional labels are needed to handle the HOS constraints. So the DP is slow for instances up to 200 requests, and intractable for larger instances. Thus, the exact column generation DP is replaced by two heuristics for the pricing problems in larger instances. The set partitioning integer program is solved on a restricted subset of the routes once column generation terminates.

The proposed fragment based branch-and-cut algorithm has no need for column generation, but instead uses a DP to schedule rests and breaks for routes retrospectively. The lifted feasibility cuts of Rist & Forbes (2021) are used to eliminate routes when rests and breaks cannot be feasibly scheduled. When they can, optimality cuts are used to incorporate the extra waiting cost into the objective. Using a schedule DP and optimality cuts in a fragment based algorithm comprise two novelties of this work. Combined, they enable the first exact algorithm for the PPDP. The algorithm is also the first exact method for a vehicle routing problem with HOS constraints, that considers multiple time windows per request, and a cost in the objective function for each hour vehicles are active. Computational results demonstrate the algorithm's ability to solve instances with up to 700 requests. We also describe a supervised learning model used to restrict the set of fragments, when fragment enumeration becomes intractable.

2 Methodology

A PPDP instance with n requests has locations $V = \{1, ..., 2n\}$, where $i \leq n$ is the pickup of request i and $i + n$ is the delivery. We denote the set of pickups by P. For all $i, j \in V$, d_{ij} is the distance and t_{ij} is the travel time between i and j. Travel times are multiples of half an hour such that the time windows of each $i \in V$ can be converted to a discrete set of time points TP_i.

2.1 Fragments

Generating all feasible routes for even moderately sized instances is intractable. In the same way as Alyasiry *et al.* (2019), we circumvent this by generating fragments instead. Fragments are sequences of pickups and deliveries where the vehicle starts empty, ends empty, is never intermediately empty and time window, capacity, pairing, precedence and HOS constraints are satisfied. We also create extended fragments by connecting each fragment to every feasible next pickup. Since fragments are shorter than routes, there are typically far fewer of them, and any route can be constructed by concatenating fragments. For example, a route such as $(p_1, p_2, d_2, d_1, p_3, d_3, p_4, d_4)$ is constructed with extended fragments $(p_1, p_2, d_2, d_1, p_3)$ and (p_3, d_3, p_4), and fragment (p_4, d_4).

2.2 Timed Network

We build a timed network from the fragments. The timed network has timed nodes for each pickup $p \in P$, (p, t), where $t \in TP_p$. We denote the set of timed nodes $\bar{P}$. Consecutive pairs of timed nodes for the same pickup are connected by waiting arcs. We denote the set of waiting arcs A. Each $a \in A$ starts at timed node $a^- = (p_a, e_a)$ and ends at timed node $a^+ = (p_a, f_a)$. We then create timed fragments from each fragment. Each timed extended fragment ω connects a timed node (i, t) to another timed node $(j, t + \Delta_\omega)$, where Δ_ω is the minimum duration of the timed fragment sequence calculated by the schedule DP. Timed fragments with no extension do not connect to a succeeding timed node since they represent the end of a route. We denote the set of timed fragments which include pickup $p \in P$ by Ω_p, and let $\Omega = \cup_{p \in P} \Omega_p$. $\Omega_{\mathtt{f}}$ is the set of timed fragments originating from fragment $\mathtt{f}$. Each $\omega \in \Omega$ has corresponding fragment $\mathtt{f}_\omega$ and starts at timed node ω^-. Timed extended fragments ω end at timed node ω^+.

We then define the relaxed fragment mixed integer program, RF. RF has binary variables x_ω for each $\omega \in \Omega$, y_a for each $a \in A$, and z_j for each $j \in \bar{P}$ which are one if and only if a route starts at j. Variables θ_p for each $p \in P$ are used to implement the optimality cuts described later. RF is thus

$$\min \sum_{j \in \bar{P}} F \cdot z_j + \sum_{a \in A} \beta \cdot (f_a - e_a) \cdot y_a + \sum_{\omega \in \Omega} c_\omega \cdot x_\omega + \sum_{p \in P} \beta \cdot \theta_p \tag{1}$$

subject to

$$z_j + \sum_{a \in A | a^+ = j} y_a + \sum_{\omega \in \Omega | \omega^+ = j} x_\omega - \sum_{a \in A | a^- = j} y_a - \sum_{\omega \in \Omega | \omega^- = j} x_\omega = 0 \qquad \forall j \in \bar{P}, \tag{2}$$

$$\sum_{\omega \in \Omega_p} x_\omega = 1 \qquad \forall p \in P, \tag{3}$$

$$x_\omega, y_a, z_j \in \{0, 1\} \qquad \forall \omega \in \Omega, a \in A, j \in \bar{P}, \tag{4}$$

$$\theta_p \geq 0 \qquad \forall p \in P, \tag{5}$$

where F is the fixed route cost and β is the waiting cost per hour. For timed fragment ω with corresponding fragment $\mathtt{f}_\omega = (i_1, ..., i_l)$, c_ω is the sum of distance and waiting costs incurred by the vehicle, $c_\omega = \alpha \sum_{j=1}^{l-1} d_{i_j i_{j+1}} + \beta(\Delta_\omega - \sum_{j=1}^{l-1} t_{i_j i_{j+1}})$ where α is the per mile driving cost. So RF minimises the total fixed, distance and waiting costs, subject to flow conservation constraints (2), coverage constraints (3), binary constraints (4), and non-negativity constraints (5).

Routes are represented by chains of timed fragments at integer solutions to RF, however, chains may correspond to sequences for which rests and breaks cannot be scheduled. These are eliminated with lifted lazy constraints as in Rist & Forbes (2021).

2.3 Optimality Cuts

Suppose a route in an integer solution to RF is represented by the chain of timed fragments, $(\omega_1, ..., \omega_l)$. The route may require extra rests and breaks not included in the timed fragment schedules, since these are calculated assuming timed fragments start with zero accumulated working and driving hours. To handle this, variables θ_p for each pickup $p \in P$ model the extra waiting time of a route starting at p in the current integer solution. The extra waiting time, E, is calculated via the schedule DP. If p is the route's first pickup, we use the optimality cut

$$\theta_p \geq E \cdot \Bigg(\sum_{j \in \{p\} \times TP_p} z_j + \sum_{i=1}^{l} \sum_{\omega \in \Omega_{f_{\omega_i}}} x_\omega - l \Bigg), \tag{6}$$

to include the extra waiting cost in the objective.

3 The Schedule DP

We update Goel & Kok (2012)'s schedule DP to handle the additional HOS constraint mentioned in the introduction. The schedule DP calculates the minimum completion time for some request sequence $(i_1, ..., i_l)$ and start time t^*, given rests and breaks must be scheduled according to HOS constraints. It returns ∞ if the sequence cannot satisfy the HOS constraints. The DP is used to assess the feasibility of fragment sequences during their enumeration, and to calculate the minimum completion time of each timed fragment given their start time. Also, it is used to assess the feasibility of routes and separate optimality cuts at integer solutions to RF.

4 Learning Model

In large instances with high request spatial density, enumerating time fragments becomes impossible due to the memory limit. We propose a simple supervised learning model which restricts the set of fragments and hence $|\Omega|$. A sequence of pickups and deliveries $\mathtt{f}' = (i_1, ..., i_l)$ is a partial fragment if there exists a fragment $\mathtt{f} = (i_1^*, ..., i_m^*)$ such that $i_j = i_j^*$ for $j = 1, ..., l$. We let $\mathcal{X}_{\mathtt{f}'}$ be the set of pickups p for which $(i_1, .., i_l, p)$ is also a partial fragment. Enumerating fragments involves repeatedly extending partial fragments $\mathtt{f}'$ to all $p \in \mathcal{X}_{\mathtt{f}'}$.

For each partial fragment $\mathtt{f}'$, instead of extending to all pickups in $\mathcal{X}_{\mathtt{f}'}$, we can extend to some subset $R_\kappa(\mathtt{f}') \subseteq \mathcal{X}_{\mathtt{f}'}$ where κ is a fixed integer and $|R_\kappa(\mathtt{f}')| = \kappa$. To construct $R_\kappa(\mathtt{f}')$ we require feature vector $\underline{v}(\mathtt{f}', p) \in \mathbb{R}^h$ for each $p \in \mathcal{X}_{\mathtt{f}'}$ which depends on the partial fragment $\mathtt{f}'$ and the candidate pickup extension p. Firstly, we obtain an ordering, $p^{(1)}, ..., p^{(|\mathcal{X}_{\mathtt{f}'}|)}$, of the pickups in $\mathcal{X}'_{\mathtt{f}}$ by arranging them in ascending order with respect to a linear combination of the features, $u(\mathtt{f}', p|\underline{a}) = \underline{a}^\top \underline{v}(\mathtt{f}', p)$, where $\underline{a} \in \mathbb{R}^h$. Then we let $R_\kappa(\mathtt{f}') = R_\kappa(\mathtt{f}'|\underline{a}) = \{p^{(1)}, ..., p^{(\kappa)}\} \subseteq \mathcal{X}_{\mathtt{f}'}$.

Choosing weight vector $\underline{a}$ involves collating N partial fragment next pickup pairs, $\{(\mathtt{f}'_j, p_j)\}_{j=1}^N$, from optimal solutions to small, randomly generated instances. We use a loss function that counts the number of times $p_j \notin R_\kappa(\mathtt{f}'_j|\underline{a})$,

$$l(\underline{a}|\{(\mathtt{f}'_j, p_j)\}_{j=1}^N) = N - \sum_{j=1}^{N} \mathbb{I}(p_j \in R_\kappa(\mathtt{f}'_j|\underline{a})). \tag{7}$$

Hence, to find $\underline{a}$ we solve

$$\min_{\underline{a} \in \mathbb{R}^h} l(\underline{a}|\{(\mathtt{f}'_j, p_j)\}_{j=1}^N). \tag{8}$$

5 Computational Results

Table 1 gives summary results for the proposed algorithm on instances generated according to Xu *et al.* (2003)'s procedure, except we restrict problems to one vehicle and carrier type with no request incompatibilities. Column Size gives the side length of the square region which locations are distributed in. Values given in each row are averages over 15 instances.

The Ω column gives the number of timed fragments enumerated. The T_1 column gives timed fragment enumeration duration in seconds. Column T_2 gives solve times, which are limited to 7200 seconds. Bold font indicates that all 15 instances are solved to optimality. The numbers in superscript give the number of instances where timed fragments can be enumerated, and the number of instances solved to optimality respectively. Column Gap is calculated as $\frac{UB-LB}{LB}$ where UB is the best objective value found and LB is the largest lower bound found within the time limit when solving exactly. The $\hat{\Omega}$ Prop. column gives the proportion of timed fragments enumerated when the extension restriction model is used with $h = 3$ and $\kappa = 3$. Finally, column $\hat{\text{Gap}}$ is calculated in the same way as column Gap, except UB is replaced with $\hat{UB}$ which is the best objective value found when using the extension restriction model.

Instances with up to 300 requests and high spatial density are solved to optimality. It is worth noting that Xu *et al.* (2003) only give near optimal solutions for similarly sized instances with lower spatial density. Our algorithm can also solve larger instances with lower spatial density to optimality. Finally, the extension restriction model achieves near optimal solutions with fewer timed fragments, especially for problems with high spatial density. Hence, we infer that near optimal solutions can be reliably obtained for instances with $n \geq 400$ and a Size of 800.

Table 1 – *Computational Results for Randomly Generated Instances*

n	Size	Ω	$T1$	$T2$	Gap	$\hat{\Omega}$ Prop.	$\hat{\text{Gap}}$
50	800	5161.5	0.9	**1.3**	0.0000	0.77	0.0015
	1200	2434.1	0.7	**0.5**	0.0000	0.94	0.0004
	1600	1385.0	0.6	**0.3**	0.0000	0.98	0.0003
100	800	37807.7	5.5	**21.8**	0.0000	0.50	0.0055
	1200	9918.5	2.6	**2.1**	0.0000	0.82	0.0031
	1600	4934.3	2.1	**1.1**	0.0000	0.93	0.0016
200	800	329574.5	48.2	$2475.1^{(15,7)}$	0.0018	0.25	0.0121
	1200	52497.0	12.1	**24.1**	0.0000	0.65	0.0062
	1600	21907.1	9.0	**4.5**	0.0000	0.83	0.0041
300	800	1921789.9	295.9	$5684.4^{(15,2)}$	0.0061	0.13	0.0215
	1200	184671.5	37.0	**107.0**	0.0000	0.46	0.0097
	1600	57035.6	21.4	**15.3**	0.0000	0.74	0.0062
400	800	5017686.6	814.7	$7261.5^{(5,0)}$	0.0206	0.09	0.0299
	1200	432336.1	90.7	$1398.4^{(15,12)}$	0.0002	0.37	0.0145
	1600	114166.9	41.9	**38.0**	0.0000	0.67	0.0098
500	1200	936553.5	183.5	$3679.6^{(15,7)}$	0.0010	0.29	0.0155
	1600	212178.4	71.4	**124.2**	0.0000	0.58	0.0125
600	1200	1492127.7	307.6	$5166.9^{(15,1)}$	0.0019	0.26	0.0185
	1600	346486.3	117.6	**565.9**	0.0000	0.53	0.0133
700	1200	3062339.8	647.5	$6499.0^{(12,1)}$	0.0046	0.19	0.0230
	1600	521197.7	178.1	$1202.7^{(15,13)}$	0.0001	0.47	0.0154

References

Alyasiry, Ali Mehsin, Forbes, Michael, & Bulmer, Michael. 2019. An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and Last-in-First-out Loading. *Transportation science*, **53**(6), 1695–1705.

Goel, Asvin, & Kok, Leendert. 2012. Truck Driver Scheduling in the United States. *Transportation science*, **46**(3), 317–326.

Rist, Yannik, & Forbes, Michael A. 2021. A New Formulation for the Dial-a-Ride Problem. *Transportation science*, **55**(5), 1113–1135.

Xu, Hang, Chen, Zhi-Long, Rajagopal, Srinivas, & Arunapuram, Sundar. 2003. Solving a Practical Pickup and Delivery Problem. *Transportation science*, **37**(3), 347–364.