Real-World Pickup and Delivery Problem with Transfers

Václav Sobotka, Hana Rudová

Faculty of Informatics, Masaryk University, Brno, Czech republic
sobotka@mail.muni.cz, hanka@fi.muni.cz

Abstract

The pickup and delivery problem with transfers generalizes the classical pickup and delivery problem (PDP) by allowing the vehicles to exchange request loads at designated transfer points. Transfers often lead to substantial reductions in transportation costs, yet they come with a significant burden of additional computational complexity. Even meta-heuristic methods are thus limited to instances of at most lower hundreds of requests leaving the desirable benefits unreachable for larger instances. Our approach bypasses the complexities inherent to current methods by deciding about the transfers apriori and thus reducing the problem to a PDP instance. To make as informed decisions as possible, we analyze a broader set of characteristics that may be used to carry out the apriori decisions. We opt to derive and examine multiple such PDP instances to cover different transfer choices. Our analysis of the derived PDP instances then allows their efficient processing in parallel. The proposed framework addresses a large-scale freight transportation problem with real-world characteristics and transfers where typical instances count over 1,200 requests and 300 vehicles. We show the potential of the proposed framework on both real-world and synthetic instances with up to 1,500 requests. The experiments demonstrate that substantial savings may be achieved within favorable runtimes even for very large instances.

Introduction

Vehicle routing problems (VRP) are broadly studied problems in combinatorial optimization. In order to consider real-world transportation problems, a rich palette of VRP variants has emerged. State-of-the-art classification and taxonomic review of VRPs can be found in (Vidal, Laporte, and Matl 2020; Braekers, Ramaekers, and Van Nieuwenhuyse 2016; Toth et al. 2014). One of the interesting generalizations of VRP is the pickup and delivery problem with transfers (PDPT) (Mitrovic-Minic and Laporte 2006).

In the classical pickup and delivery problem (PDP) (Cherkesly and Gschwind 2022; Curtois et al. 2018; Berbeglia et al. 2007; Li and Lim 2001), customer requests consist of pickup and delivery locations where some amount of load must be transported between them. PDPT additionally allows for transferring request loads between vehicles in dedicated transfer points. Such transfers potentially reduce transportation costs due to load consolidation and more flexible service of requests over long distances. The savings may reach up to higher tens of percent, but, unfortunately, they come together with a significant burden of additional computational complexity. Consequently, state-of-the-art works are limited to less than ten requests for exact methods and lower hundreds of requests for heuristic approaches leaving large-scale instances out of their reach.

We address a real-world variant of PDPT with large-scale instances of around 1,200 requests and more than 300 vehicles. Despite the order of magnitude larger number of requests, we achieve significant savings by introducing the transfers in both real-world and synthetic instances.

Let us summarize the key contributions of our work.

1. We analyze existing works on PDPT where the complexity of the crucial transfer-handling mechanisms limits their scaling and we recognize the potential of apriori decided transfers to bypass these inherent limitations.
2. Inspired by the straightforward approach to transfers (Petersen and Ropke 2011), we propose an analytical approach by (1) identifying the transfer characteristics essential for the apriori decisions, (2) proposing the strategy for deriving a set of multiple complementary PDP instances and (3) efficiently processing the set of PDP instances in parallel in order to find the best-performing PDP instance.
3. We provide a novel analysis of the effects of transferred requests in large-scale instances and their potential to decrease the solution cost.
4. For large-scale real-world and synthetic instances, we experimentally show that our approach consistently outperforms the method from Petersen and Ropke (2011) by up to 18.9 % and achieves savings of up to 36.1 % in comparison to solutions without transfers while maintaining favorable scaling properties.
5. We provide the set of synthetic instances used in our experiments together with the PDPT instance generator utilizing real-world geographical data from OpenStreetMap and characteristics of real-world cases.
6. By exchanging the underlying PDP solver, our methodology has a high potential to address transfers in the context of different problem variants and constraints.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Problem Description

The approached problem and real-world data were provided to us by our industrial partner Wereldo.com (Wereldo 2023). It targets large-scale freight transportation problem with the possibility of transfers and multiple real-world characteristics. The core of the problem at hand is the pickup and delivery problem with transfers, with instances of around 1,200 requests and 300 vehicles in the fleet.

As outlined, each customer request in PDPT is represented by its pickup and delivery locations. First, some vehicle must pick up the request load at the pickup location. In contrast to a standard pickup and delivery problem, however, this vehicle may or may not deliver the load to the delivery location. Instead, it is possible to visit a transfer point and drop the load there. Sometime later, the load is transported from the transfer point to the delivery location, usually by a different vehicle.

Our problem includes *vehicle capacities* and *time windows*. Capacities are limited both in terms of weight and volume of the load. Time windows have hard boundaries. The vehicle fleet is *heterogeneous* in both capacities and costs. The objective is to minimize transportation costs, i.e., the sum of distances traveled by each vehicle weighted by the vehicle's price per kilometer.

Additional features are the limit on the *route duration*, limit on the number of *stops* in a route, and *service times*. Route duration refers to the on-road time of the given vehicle. A stop refers to a physical stop of a vehicle with the purpose of (un)loading. Importantly, if a route contains multiple subsequent (un)loading actions at the same location, the whole sequence of actions is counted as a single stop. Lastly, service time refers to the time necessary to handle (un)loading. The service time is counted per stop, i.e., only once per sequence of (un)loading actions at one location.

Regarding transfer points, several logistical facilities are available in strategic locations. Intermediate storage is possible, and thus, the transferring vehicles need not meet at the transfer point. The load of one request must not be split. Based on the discussions with our industrial partner, the transfers incur no costs. The facilities operate 24/7. Both the loading and unloading parts of transfers are subject to standard service times. Notably, there are multiple depots physically coinciding with the transfer points that serve as pickup locations for a large portion of requests.

As the full formal model with proper description takes around 5 pages, we do not reproduce it here due to the length restrictions. For key aspects of PDPT modeling and common problem features, we refer interested readers to (Cortés, Matamala, and Contardo 2010).

Related Works and Motivation

The main approaches used to address the transfers as well as their scaling limitations are presented. A summary of the reviewed works together with their properties is in Table 1.

With few exceptions, PDPT is mostly approached with heuristics dominated by meta-heuristic methods. Exact algorithms are proposed in (Cortés, Matamala, and Contardo 2010; Rais, Alvelos, and Carvalho 2014; Lyu and Yu 2022). Interestingly, the largest PDPT instances solved to optimality by exact methods contain less than 10 requests clearly indicating the intrinsic complexity of PDPT. We proceed by discussing how the complexity of transfers severely limits even the prevalent meta-heuristic methods. This realization ultimately motivates our approach as we face instances of more than 1,000 requests.

The pioneering work on PDPT (Mitrovic-Minic and Laporte 2006) approaches the problem with an iterative heuristic. Its core is a request insertion operation taking transfers into consideration. First, the request is greedily inserted without any transfer. Then, for each transfer point, the tour from request pickup to the transfer point is inserted greedily. This choice is fixed and the tour from the transfer point to the request delivery is again greedily inserted. Analogously, this is repeated starting with the transfer point to delivery tour. Finally, the cheapest of all attempted insertions is realized. Variants of this mechanism were also used in (Masson, Lehuédé, and Péton 2013) in the form of ALNS operators.

A GRASP solver with an ALNS improvement phase is introduced by (Qu and Bard 2012). In (Danloup, Allaoui, and Goncalves 2018), PDPT is approached with LNS and GA solvers. Sampaio et al. (2021) address a crowdshipping problem with transfers by means of ALNS solver. All these works share the mechanism of inserting requests with the possibility of transfer. First, all the possibilities to insert the tour from request pickup to a transfer point are identified. The same is done for the tour from a transfer point to request delivery. The options to serve the first and second parts of the transfer are then matched against each other. This is done for each transfer point, and the cheapest possible feasible variant is ultimately realized. It should be noted that strategies for limiting the number of involved transfer points or routes are present. However, their influence does not change the significantly higher asymptotical complexity of the full exploration of transfer options.

Voigt and Kuhn (2022) present an ALNS solver for a problem with occasional drivers. The key request insertion mechanism tests all combinations of vehicle pairs and transfer points. If a transfer point is not visited in the first route, it is inserted into the best position and fixed. The pickup location is then inserted into the best position before the transfer point visit. The second route is handled analogically. This greedy approach is suitable since time windows are not assumed, yet it results in a complexity comparable to (Mitrovic-Minic and Laporte 2006).

A different type of transfer-handling strategy is present in (Manier et al. 2019) and (Fu and Chow 2022). The core idea shared by these works is to construct a solution without any transfers first. Then, it is iteratively improved by introducing transfers. In both works, pairs of vehicles capable of meeting at a transfer point are considered. Then, it is tested whether it is possible to transfer any request from one vehicle to another via the given transfer point. After inspecting any valid combinations of a vehicle pair, and transfer point for transfers, the most improving transfer is realized greedily.

In summary, the described methods suffer from inherent scaling limitations. Standard transfer-less insertion for a single request in the pickup and delivery setup takes $O(|V| \cdot L^2)$

Work	*C*	*W*	*F*	*T*	*S*	*R*	*Solution method*	*Objective*
Mitrovic-Minic and Laporte (2006)	×	✓	✓	5	✓	100	Multi-phase iter. heur.	distance
Cortés et al. (2010)	✓	✓	×	1	✓	6	ILP, custom impl.	total time
Petersen and Ropke (2011)	✓	✓	×	1	cost	982	Apriori transfers + ALNS	$\sum_{\text{vehicles}}$(distance, initial & up-time veh. cost, cost of transfers, road toll)
Qu and Bard (2012)	✓	✓	×	1	✓	25	GRASP+ALNS	lexicographic(|vehicles|, distance)
Masson et al. (2013)	✓	✓	×	33	✓	193	ALNS	distance
Rais et al. (2014)	✓	✓	✓	any	✓	7	ILP, Gurobi	distance
Danloup et al. (2018)	✓	✓	×	5	✓	100	LNS, GA	distance
Manier et al. (2019)	✓	soft	✓	6	limit	150	Multi-phase iter. heur.	pareto(distance, tardiness)
Sampaio et al. (2021)	×	✓	×	5	✓	100	ALNS	distance
Fu and Chow (2022)	✓	×	×	any	none	300	Multi-phase iter. heur.	$\sum_{\text{weigh.}}$(distance, veh. transfer time, customer wait. time, total travel time)
Voigt and Kuhn (2022)	×	×	✓	4	✓	100	ALNS	distance + cost for occasional drivers
Lyu and Yu (2022)	✓	✓	✓	4	✓	5	ILP, Gurobi	distance
Our work	✓	✓	✓	4	✓	1,500	Apriori transfers + ALNS	$\sum_{\text{vehicles}}$ veh.cost-per-km * distance

Table 1: Overview of PDPT properties in the reviewed literature. *C* – capacitated, *W* – time windows, *F* – heterogeneous fleet, *T* – no. transfer points, *S* – unlimited storage at transfer points, *R* max. no. requests.

operations (V is the set of vehicles, L is the maximum route length in the number of locations). In contrast, even the cheapest transfer-aware mechanism from (Mitrovic-Minic and Laporte 2006) requires $4 \cdot |T|$ times more calculations per one insertion operation (T is the set of transfer points). More sophisticated approaches have complexities of up to $O(|T| \cdot |V|^2 \cdot L^4)$. Since such insertions are heavily iterated throughout the search, the additional complexity critically influences the overall performance. Consequently, the usual size of addressed PDPT instances is around 100 requests or less, with the largest addressed instances of around 300 requests (see Table 1).

The only work solving larger instances of up to 1,000 requests is (Petersen and Ropke 2011). They reduce the large PDPT instances into PDP instances by taking the transfer-or-not decisions apriori. The decisions are driven by a simple distance-based heuristic rule. Unfortunately, the work concentrates on the transfer aspects only marginally and its main focus is on parallel solving of the obtained PDP instance. We recognize the prospects of this direction especially in bypassing the complexities of transfers and we aim to explore and demonstrate its full potential.

Proposed Method

Our method follows the discussed direction of deciding about the transfer apriori and then solving the problem as PDP. For each request, it must be decided whether to transfer it and if so, a transfer point must be selected. A request to be transferred is then split into two new requests, one for the trip from the pickup to transfer point and second for the trip from the transfer point to delivery. Consequently, the apriori decisions induce a PDP instance together with a mapping between the original and derived requests. Any solution to this PDP instance together with the mapping forms a solution to the original PDPT instance.

In comparison to (Petersen and Ropke 2011) we introduce conceptual improvements in two areas. First, we choose to derive and examine multiple PDP instances rather than a single one. This allows us to explore different decisions about the transfers resulting in higher quality of results. Moreover, as we do not rely on a one-shot decision driven by a tuned threshold as Petersen and Ropke (2011), the method is more resilient against slight changes in the character of the instances at hand. The second area of our improvements is a systematic exploration of different characteristics of requests and transfers used for carrying out the apriori decisions and identification of the important ones. Overall, the main points of our method are (1) the construction of the *instance bundle* with multiple different derived PDP instances based on different sets of apriori decisions, and (2) efficient parallel processing of the instance bundle. The key concerns of (1) are to identify the characteristics that influence the suitability of a potential transfer and to build the instance bundle so that its instances complement each other. With (2), the goal is to avoid the need to fully solve every single instance in the bundle as this naive approach would be simply too time-consuming.

Transfer Schemes

Transfer scheme (rather than request) is the basic decision unit taken into consideration. With R being the set of requests in the PDPT instance and T being the set of transfer points, denote the elements of the set $R \times T$ as transfer schemes. A particular transfer scheme represents the possibility of serving the request $r \in R$ with a transfer via the transfer point $t \in T$. Note that a transfer scheme may be infeasible due to time constraints.

In order to make an apriori decision about transfers, each request $r \in R$ must be either chosen to be served directly (without a transfer), or a particular feasible transfer scheme must be selected for r. Then a PDP instance based on this decision may be obtained as follows. With the exception of

requests, all properties of the given instance are copied. Requests chosen to be served directly are copied without any change. For the remaining requests, their selected transfer scheme is applied.

The transfer scheme (r, t) is applied by replacing the request r from the original instance with two new requests r_1 and r_2. The request r_1 covers the tour from the pickup location of r to the transfer point t, whereas r_2 covers the tour from t to the delivery location of r. Service times of r_1 and r_2 at t are taken from properties of t. The time windows at the pickup and delivery locations are inhereted directly from r. The time windows at t have no lower bound for r_1 and no upper bound for r_2 since t operates 24/7. The upper bound of time window of r_1 at t is the same as the lower bound of time window of r_2 at t. The value is calculated as follows. First, the earliest possible arrival at t is calculated based on the lower bound of pickup time window, service times and travel times. Symmetrically, the latest possible arrival at t is calculated yielding a time interval of possible valid arrivals at t. The value in question is the center of this interval, i.e., the available slack is distributed evenly between r_1 and r_2.

Transfer Scheme Properties

In order to make reasonable apriori decisions about transfers, it is advisable to exploit as much available information as possible. The transfer scheme can be seen as a multi-dimensional object where each dimension represents a particular property potentially relevant to such decision-making. We worked with seven such properties some of which turned out to be redundant during parameter tuning.

Detour compares the tour from pickup via transfer point to delivery with the pickup-to-delivery tour in terms of distances. Absolute detour is the difference between these two distances while the relative detour is their ratio. Intuitively, transfer schemes with lower detours are naturally more favorable as their application tends to be cheaper.

Pickup-to-delivery distance is simply the direct distance between the pickup and delivery location of a given request. Requests with rather long pickup-to-delivery distances are generally more expensive to serve as it tends to be harder for a vehicle to serve such requests together with other requests. Splitting such a request into two reduces the risk of being served by a dedicated vehicle and situations in which an empty vehicle undertakes a very long return to its depot.

Uniformity measures how evenly the visit to the transfer point divides the tour from pickup via transfer point to delivery. While rather even divisions tend to produce two similarly long requests, asymmetric divisions create one short and one long (potentially longer than the original) request. The more even divisions should be preferred as they do not produce yet another long-distance request. Uniformity is calculated as $\frac{\|[P{\rightarrow}TP] - F\| + \|[TP{\rightarrow}D] - F\|}{[P{\rightarrow}TP] + [TP{\rightarrow}D]}$, where $[P{\rightarrow}TP]$ and $[TP{\rightarrow}D]$ denote the distances from request pickup to transfer point and from transfer point to request delivery, respectively. F is the ideal half of the complete distance upon transfer, i.e., $([P{\rightarrow}TP] + [TP{\rightarrow}D])/2$.

Slack represents the time-tightness of a given transfer scheme. Slack is calculated as the available time between the earliest pickup and the latest delivery reduced by the minimum time required to cover the given distances and service times at pickup, transfer point, and delivery locations. Applying transfer schemes with larger slacks is expected to introduce less inflexibility and is thus preferable to more time-tight alternatives.

Weight and volume tend to restrict the usefulness of transfers. When transfers are used to a larger extent, there is the potential to efficiently collect multiple request loads into a transfer point and subsequently deliver them. The usefulness of such consolidation relies on how efficiently the loads may be collected and later delivered in larger groups. In this regard, less capacity-restricted requests are more likely to aggregate into large groups.

Instance Bundle Construction

During preliminary experiments with apriori decided transfers, we identified that the achieved savings heavily depended on the general number of transfers applied. While too few transfers usually resulted in even worse solution qualities, adding more and more transfers generally greatly helped up to a certain point. Then, the trend sooner or later flipped and additional transfers became harmful. Eventually, too many transfers led to worse results than those achieved without transfers. This observation is further discussed in Section Experiments. We exploit this observation by using the following strategy to create the instance bundle.

The first instance in the bundle applies only the most suitable transfer schemes. Each following instance applies all the transfer schemes as its predecessor plus an additional group of less suitable transfer schemes. Ultimately, the last instance in the bundle forces all transfer schemes to be applied. The strategy outlined above is described in the procedure INSTANCEBUNDLE in Figure 1.

The goal of lines 2 and 3 is to get the set of transfer schemes to be further processed. All candidates are filtered by removing transfer schemes with less than an hour of slack (including the infeasible). Also, transfer schemes with relative detours larger than 2.0 and absolute detours exceeding half of the longest pickup-to-delivery distance are discarded. The point of the limitation of the relative detour is to disallow transfer schemes that necessarily produce requests with longer pickup-to-delivery distance than the request being split. The point of the absolute detour limitation is to further restrict the detours for long-distance requests.

Line 4 clusters the reasonable transfer schemes based on the dimensions (properties of transfer schemes) D and their respective weights W (given as a parameter). As a result, we obtain a set of K disjoint clusters grouping transfer schemes with similar properties. These clusters then serve as the groups, by which we incrementally add more and more transfer schemes to be applied when building the next instance in the bundle. Lines 5 and 6 order the clusters based on their suitability to be applied. This step defines a reasonable order of priority so that the more promising transfer schemes are fixed to be transferred first. Next, lines 7 and 8 filter the clusters so that there is at most one transfer scheme for each request in the instance. More details about the clustering (line 4), scoring (lines 5 and 6), and filtering

```
1: procedure INSTANCEBUNDLE(R, T, D, W, K)
   R requests, T transfer points, D dimensions,
   W weights, K number of clusters
2:   S ← R × T
3:   remove excessive-detour/infeasible schemes from S
4:   C ← CLUSTERSCHEMES(S, D, W, K)
5:   score(C) ← SCORE(centroids(C), D, W)
6:   sort clusters in C by score(C) ascending
7:   eliminate same-request schemes across clusters
8:   eliminate same-request schemes inside clusters
9:   I ← ∅                (bundle of instances)
10:  for i = 1 to K do
11:      I ← I ∪ {DERIVEINSTANCE(R, ⋃_{j=1}^{i} C_j)}
12:  return I
```

```
1: procedure SCORE(Schemes, Dimensions, Weights)
2:   foreach d ∈ Dimensions, s ∈ Schemes do
3:     a(s, d) ← rank of s among Schemes using d
4:   foreach s ∈ Schemes do
5:     score(s) ← Σ_{d∈Dimensions} Weights(d) * a(s, d)
6:   return score(Schemes)
```

Figure 1: Pseudo-codes of the instance bundle construction and transfer scheme scoring.

(lines 7 and 8) are given in the following parts of this section. Finally, lines 9 to 11 take the clusters in the established order and derive the bundle so that the i-th instance takes the i most suitable clusters and applies all their contained transfer schemes. Recall that the requests not included in any of the i applied clusters are simply copied without any transfer.

Transfer scheme clusters The clustering described in the procedure CLUSTERSCHEMES is performed by means of standard methods, e.g., hierarchical agglomerative clustering or k-means (Aggarwal and Reddy 2014, p. 101, 89) algorithms. Euclidean distance is used as the similarity measure. The transfer scheme is understood as a multi-dimensional object where the particular dimensions are represented by the discussed properties. As a preprocessing step, all the dimensions $d \in D$ from the input S are first standardized and scaled by dimension weights W. Then, the required K clusters are computed. The values of W and K passed to the clustering procedure are hyperparameters of our method.

The clusters obtained in this way cannot be, however, used as the increments for the bundle construction directly. If a subset of clusters is selected and all the contained transfer schemes would need to be applied, there may be multiple transfer schemes for a single request. Such conflicts are greedily prevented on lines 7 and 8 and require the possibility to compare the transfer suitability of clusters and transfer schemes in general. Line 7 takes each request and discards its relevant transfer schemes in all but the most suitable cluster. After resolving ambiguities across clusters, potential conflicts inside clusters are addressed similarly in line 8. In the case of more transfer schemes of the same request within one cluster, they get compared in the same way as the clusters, and only the most suitable transfer scheme is kept. As a result, each request is unambiguously set to be transferred via a single transfer point, or not at all.

Transfer scheme suitability scoring The transfer scheme clusters need to be ordered based on their transfer suitability. This is done by representing the clusters by their centroids (average transfer schemes) which are then compared based on the discussed transfer scheme properties. The pseudocode of the procedure SCORE is in Figure 1.

The score used for the ordering is a weighted sum of ranks across each dimension $d \in D$. During the ranking, potential ties are resolved by assigning the same lowest possible rank to all of the tied elements. The dimension weights W are parameters of the algorithm and are the same as the weights used for dimension scaling during the clustering phase.

Instance Bundle Processing

With tens of PDP instances in the bundle, it is clearly unacceptable to naively perform a full run of PDP solver for each of the instances. This is especially important as our main motivation is to allow for addressing very large problem instances within acceptable runtimes. Fortunately, it is possible to drastically reduce the naive time requirements.

Instance pruning In principle, the whole procedure can be seen as a search on two levels. First, the search in terms of the transfers is about choosing the PDP instance from the bundle with as little achievable solution cost as possible. Second, we search for optimal routing for a particular PDP instance. In order to compare the instances, it is essential to run the PDP solver on them. However, the key observation is that a relatively short PDP search is enough to discriminate which instances have the perspective of yielding high-quality solutions.

This observation is used to prune instances in the bundle in two waves. First, very short runs (several hundreds of iterations) of the PDP search are performed on all instances in the bundle. Then, only a certain number of best-performing instances are kept. In the second wave, the PDP search is restarted for the surviving instances from their best-achieved solution and a longer search is performed. Then, only a handful of the best-performing instances are kept. These candidates are then again restarted to their best-achieved solutions and the search continues for a longer time. In the end, the solutions from this last phase together with their transfer decisions form the output of the solver.

Faster convergence We observed that the PDP solver converges notably faster and with more stable results in instances where some transfer schemes have been applied. This behavior is likely a result of the more constrained character of such instances stemming from the applied transfers. Based on this observation, the last phase of the search may be relatively short without damaging the quality of the overall results.

Parallelization Processing the instance bundle consists of many relatively long and independent tasks. This makes it a perfect fit for a trivial parallelization. Thus, the performance of this approach may be conveniently scaled in an almost linear manner by simply providing additional CPUs.

Experiments

This section evaluates the proposed transfer handling framework. The framework was implemented in Python 3.9. The internal PDP solver is an implementation of adaptive large neighborhood search (ALNS) from (Ropke and Pisinger 2006) written in Go (go1.15.15 linux/amd64) which is used by the company in practice (Sassmann et al. 2023). The computational experiments were conducted on virtualized infrastructure under CentOS8, Intel Xeon Skylake 2.3 GHz, 8 CPUs, and 16 GB RAM.

Datasets

The evaluation used both real-world and synthetic instances. The synthetic instances were generated by our instance generator that utilizes geographical backgrounds from OpenStreetMap© (OSM) (OSM 2022). Characteristics extracted from the original data were used for the instance generation. The generator and synthetic instances are available at *https://sites.google.com/view/real-world-pdpt/*.

Instances The real-world data consists of 5 instances, each covering one full day of transportation requests. The instances contain around 1,200 requests, a fleet of 323 vehicles, and 4 transfer points. The synthetic instances were generated based on 3 different countries (Czechia, Hungary, Slovakia) with 500, 1,000, and 1,500 requests. Different setups with 2, 3, and 4 transfer points are considered.

Several characteristics are shared by all requests or vehicles in real-world instances. All service times are 15 minutes long, and the routes are limited to at most 15 stops and 11 hours of duration. An important property of the dataset is that around 85 % of request is delivery-only (the pickup locations are in one of the transfer points serving as depots). Since the transfer points operate 24/7, the pickup time windows of the delivery-only requests are typically limited only by the delivery deadline. The vehicle fleet is heterogeneous both in capacities as well as in operational costs. Regarding capacities, a significant portion of the fleet (68.1 %) is composed of large trucks capable of accommodating 66 or 72 pallets and up to 24 tons of load. The remaining 6 vehicle types are very different, with capacities from 4 to 25 pallets. Around 70 % of the vehicles have cost of K per kilometer. The majority of the remaining vehicles has cost around 3.0 K since these vehicles are contracted from a third party to complement the regular fleet.

OpenStreetMap instance generator Our aim is to generate instances with reasonable spatial properties as in real-world data. In order to achieve this, the spatial distributions are extracted from the OSM geographical datasets for whole countries. Moreover, the positions for the transfer points are chosen so that they reasonably cover the serviced region rather than picking fixed or random locations.

The OSM datasets serve to extract candidate locations for pickups and deliveries of the requests. In order to mimic the real-world data, industrial zones were used as the image for pickup locations, and the delivery locations were represented by supermarkets. These extracted locations together with transfer points form the pool from which pickup and delivery locations of the generated requests are sampled.

In real infrastructure, the placement of transfer points is far from arbitrary. In fact, deciding where to build a new logistical facility is a strategic decision. In order to reflect this, we place the transfer points by solving a variant of the facility location problem (Drezner and Hamacher 2002) by means of integer linear programming. Thus, the transfer point placement is reasonable with respect to the serviced spatial distribution of customer locations.

With the pool of candidate locations and transfer point placement strategy at hand, the generator creates new instances by sampling pairs of pickup and delivery locations. Then, the properties of the requests are randomly chosen based on fully configurable distributions. The instances contain multiple depots coinciding with the transfer point locations. Vehicles are evenly distributed across the depots. The vehicle types and their properties available in the generator were identified together with our industrial partner.

Parameter Tuning

The parameters form two groups. The first group controls how the instance bundle is processed. Its parameters govern the size of the bundle, the number of iterations in each phase, pruning between phases, and possible replication factors of the calculations. The second group is the weights W used for scoring and rescaling during the clustering phase.

Bundle processing parameters The choice of these parameters is mainly empirical since it requires balancing the tradeoffs between result stability and the required computational costs. For the initial phase, we choose 40 instances in the bundle solved in 2 replicas for 500 iterations. For the middle phase, we filter 16 best-performing instances and run them for additional 1,500 iterations in a single replica. The final phase takes 4 best instances and runs them for additional 5,000 iterations in 2 replicas. Generally, the parameterization should allow for full utilization of all 8 CPUs.

Based on our preliminary experiments, a bundle of 40 instances turned out to be sufficiently broad. Second, running each instance in the bundle for 500 iterations in 2 replicas was identified to be enough to discard poorly performing instances. Attempts to prolong the initial phase even for the largest instances did not yield any significant qualitative improvements and, thus, are not worth the additional resources.

The purpose of the middle phase is to prolong the search on the reasonably performing instances. We observe that a significant portion of the improvements occurs during the first 2,000 iterations of the search. Result qualities in this stage of search are thus relatively close to the final results and may be used for more precise and strict pruning.

The goal of the final phase is to search for a longer time on a handful of the most promising instances. As discussed, the instances containing transferred requests tend to converge faster as they are more constrained. Consequently, additional 5,000 iterations of the search proved to be sufficient. Again, further prolongation results in marginal benefits at the expense of much longer runtimes.

Weights of transfer scheme properties Systematic tuning of the weights revealed two important insights. First, contrary to our initial experience, only 3 of the properties

turned out to affect the results positively. Second, the ratio between the weights of the relevant properties does not play a meaningful role. Consequently, the final choice of the weights is 1 for the relative detour, pickup-delivery distance, and uniformity, while the remaining weights are set to 0.

Preliminary experiments suggested that relative detour and pickup-delivery distance consistently play important roles, and some properties may be redundant. Thus, the tuning proceeded in two stages. First, a grid search with possible values 0 or 1 was performed on the properties that were suspected to be redundant. Only the properties whose weights positively correlated with the achieved savings were kept. Second, another round of grid search with possible values 0, 0.5, and 1 was performed on the remaining properties. Its purpose was to assess whether the key factor is only the presence of the weights or whether the ratio of their values has a significant impact on the overall results.

Both the gird searches were conducted on a set of 9 separately generated 500 request instances with 3 different geographies and setups with 2, 3, and 4 transfer points. The measured response variable was the sum of savings across the 9 instances. These savings were measured in percent with respect to the baseline results for the respective instance in order to mitigate cost differences between instances. Further results from the tuning are available at *https://sites.google.com/view/real-world-pdpt/*.

Benefits of Transfers

The goal of the experimental evaluation is to analyze the potential savings that can be achieved by allowing to transfer requests. To the best of our knowledge, the only work handling the transfers apriori (Petersen and Ropke 2011) does not report this information. Thus, we adapt their method for more than one transfer point and evaluate the transfer benefits for both the adaptation and our PDPT framework.

Petersen and Ropke adaptation Petersen and Ropke (2011) use a few simple rules to decide whether to split a request into two or not. First, any realized transfer must be feasible, i.e., it must be possible to serve both the created requests without violating any time constraints. Second, requests with a large temporal gap between the latest pickup and earliest delivery are transferred. Lastly, requests that may be transferred with relative detours smaller than some threshold are transferred.

To allow for solving instances with more than one transfer point, we adapt these rules as follows. First, the transfer point with the lower detour is always preferred when deciding how to transfer a request. Second, we drop the rule targeting the large temporal gaps as such gaps do not appear in our dataset due to typical time window patterns. The relative detour threshold was subject to tuning.

For the purpose of the tuning, we performed a grid search with 40 threshold values uniformly spread on the interval $\langle 1.0, 2.0 \rangle$. Each threshold value was evaluated 5 times, and the best-performing threshold value was then chosen. Each combination of country and transfer point placement was tuned on a separate instance. For synthetic instances, new tuning instances with 500 requests were generated. The real-world geography was tuned on the instance *Real 5*. This instance was not then evaluated with the adapted approach.

Instance	C_1	C_2	C_3	S_1^2	S_1^3	S_2^3
Real 1	1,318	1,146	929	13.1	29.5	18.9
Real 2	1,176	1,038	944	11.7	19.8	9.1
Real 3	1,137	1,021	846	10.2	25.5	17.1
Real 4	1,293	1,114	924	13.8	28.5	17.1
Real 5	1,054	–	825	–	21.7	–
CZ 500 2	1,556	1,148	1,134	26.2	27.1	1.2
CZ 500 3	1,427	1,174	1,082	17.7	24.2	7.9
CZ 500 4	1,670	1,186	1,113	29.0	33.3	6.1
HG 500 2	1,108	1,023	944	7.6	14.8	7.8
HG 500 3	1,578	1,406	1,284	10.9	18.6	8.7
HG 500 4	1,435	1,228	1,091	14.4	24.0	11.1
SK 500 2	1,318	997	980	24.4	25.7	1.8
SK 500 3	1,177	941	874	20.0	25.7	7.2
SK 500 4	1,211	1,084	954	10.5	21.2	12.0
CZ 1,000 2	2,684	2,166	2,032	19.3	24.3	6.2
CZ 1,000 3	2,283	1,773	1,659	22.4	27.4	6.5
CZ 1,000 4	2,574	2,218	1,977	13.8	23.2	10.9
HG 1,000 2	2,418	1,926	1,839	20.4	24.0	4.5
HG 1,000 3	2,765	2,484	2,131	10.1	22.9	14.2
HG 1,000 4	2,885	2,758	2,451	4.4	15.1	11.1
SK 1,000 2	2,033	1,614	1,563	20.6	23.1	3.1
SK 1,000 3	2,359	1,812	1,675	23.2	29.0	7.6
SK 1,000 4	2,796	2,328	1,910	16.7	31.7	17.9
CZ 1,500 2	3,668	3,026	2,778	17.5	24.3	8.2
CZ 1,500 3	3,538	2,678	2,482	24.3	29.9	7.4
CZ 1,500 4	4,823	3,484	3,082	27.8	36.1	11.5
HG 1,500 2	3,095	2,757	2,559	10.9	17.3	7.2
HG 1,500 3	3,902	3,652	3,037	6.4	22.2	16.8
HG 1,500 4	4,050	3,577	3,025	11.7	25.3	15.4
SK 1,500 2	2,975	2,428	2,346	18.4	21.2	3.4
SK 1,500 3	3,263	2,607	2,315	20.1	29.0	11.2
SK 1,500 4	3,540	3,076	2,630	13.1	25.7	14.5

Table 2: Cost comparison of experimental setups (1), (2), and (3). C_i are avg. costs of setup i over 10 repeated runs (in thousands). S_j^i are transfer savings of setup i over j (in %). Instance code format: *country*, *#requests*, *#transfer points*.

The resulting solver derives a single instance by applying all feasible transfer schemes with relative detours lower than the threshold. The instance is solved in 8 replicas in parallel under the default settings of our PDP solver (for 25,000 iterations). The best solution out of the 8 replicas is reported.

Experiment results All instances were evaluated 10 times in each of the following settings. We consider (1) baseline without transfers (PDP), (2) adaptation of Petersen and Ropke (2011) (PDPT), and (3) our method (PDPT). Primarily, we compare the transportation costs (objective) among the setups. We also inspect the covered distances, the number of used vehicles, and runtimes.

The main results of our experiments are summarized in Table 2. Overall, the introduction of transfers resulted in a substantial reduction in costs for all of the tested instances.

	CPU time			User time	
Instances	T_1	T_2	T_3	U_2	U_3
Real	3,161	30,165	13,975	3,820	2,006
Synth. 500	999	8,843	3,958	1,120	580
Synth. 1,000	2,252	21,948	9,850	2,780	1,395
Synth. 1,500	3,737	33,190	15,785	4,200	2,295

Table 3: Average runtimes (in seconds) for different instance sizes. Total CPU times are reported for setups (1), (2), and (3). User times are reported for parallel setups (2) and (3).

The savings ranged between 4.4 % and 29.0 % under the setup (2) and between 14.8 % and 36.1 % under the setup (3). Our method systematically outperforms (2) with differences ranging from 1.2 % up to 18.9 % with 9.8 % on average. Moreover, the differences are more prominent as the number of transfer points grows. Our approach is clearly better in identifying the more suitable transfer point for a request as it considers more characteristics (not only detour).

Both setups (2) and (3) significantly reduce the traveled distances with respect to baseline (1). The savings of (2) range between 5.7 % and 17.8 % with 10.4 % on average. Our approach (3) reduced the distances between 7.9 % to 18.6 % with an average of 12.4 %. We note that differences between savings in transportation costs and distances are due to the transfers reducing the dependency on the much more expensive vehicles contracted from a third party. The introduction of transfers also greatly influenced the number of vehicles used in comparison to (1). Under setup (2), the number of vehicles in solutions grew by 1.6 % on average. The worst increase was 9.3 % while some instances ended with decreases of up to 5.0 % of vehicles. In contrast, the number of vehicles under setup (3) always decreased in interval between 1.0 % to 12.2 % with an average of 6.1 % resulting in better utilization of resources.

Next, we compare the runtimes of all three setups both in terms of total CPU time, i.e., the sum of all computations across all utilized CPUs, and user time, i.e., the time of computation perceived by the user. Setup (1) utilizes 1 CPU, and setups (2) and (3) use 8 CPUs. We note that we could run (1) in 8 replicas to match the available resources across all setups. This would, however, improve the results marginally at the expense of unnecessary computations. The results are summarized in Table 3. Regarding total CPU time, the requirements of the parallel setups (2) and (3) are naturally higher than for the single-CPU setup (1). To compare the parallel setups, our approach requires roughly half the CPU time than (2). In terms of user time, our approach requires around 60 % user time in comparison to (1). Overall, our approach returns significantly better results than (2) almost twice as fast as (1) thanks to efficient parallelization.

Lastly, Figure 2 provides insights into the dependency of the number of transfers and solution costs. Each plot is for a single PDPT instance. A point in the plot represents one run on a derived PDP instance. All runs of our method on all derived PDP instances (from one original PDPT instance) from all 10 repeated calculations are included. The reason behind the convex trend is the tradeoff between load

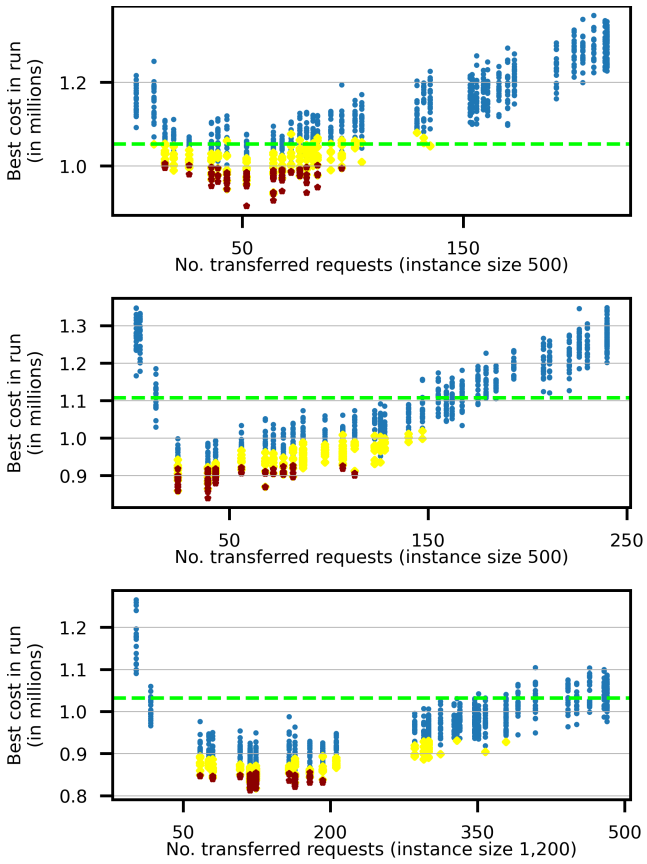

Figure 2: Cost dependency on no. transferred requests. Blue, yellow and red points are runs from the initial, middle, and final phases. The green line is the best cost without transfers.

consolidation and the introduced inflexibilities, both due to transfers. When applied en masse, transfers bring the advantage of extensive load consolidation resulting in substantial cost savings. With too many and gradually less suitable transfer schemes applied, however, the inflexibilities of forcing transfers eventually overweight the consolidation advantages. In contrast, too few transfers yield worse results than no transfers at all since they introduce inflexibilities without providing enough consolidation benefits.

Conclusion

In this paper, we propose a novel approach for addressing transfers in very large problems. Our method based on apriori decided transfers is demonstrated to achieve substantial savings while keeping favorable scaling properties for very large instances of up to 1,500 requests. To the best of our knowledge, we are the first to analyze the savings both in the context of large-scale instances as well as by deciding the transfers apriori. Experimental results achieved savings due to the introduced transfers between 14.8 % and 36.1 % consistently outperforming the method adapted from Petersen and Ropke (2011). Lastly, we provide our PDPT instance generator using OpenStreetMap geographical data and the synthetic instances used in the experiments for further use.

Acknowledgements

Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Aggarwal, C. C.; and Reddy, C. K. 2014. *Data clustering: Algorithms and Applications*. CRC Press, Taylor & Francis Group.

Berbeglia, G.; Cordeau, J.-F.; Gribkovskaia, I.; and Laporte, G. 2007. Static pickup and delivery problems: A classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research.*, 15: 1–31.

Braekers, K.; Ramaekers, K.; and Van Nieuwenhuyse, I. 2016. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99: 300–313.

Cherkesly, M.; and Gschwind, T. 2022. The pickup and delivery problem with time windows, multiple stacks, and handling operations. *European Journal of Operational Research*, 301(2): 647–666.

Cortés, C. E.; Matamala, M.; and Contardo, C. 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3): 711–724.

Curtois, T.; Landa-Silva, D.; Qu, Y.; and Laesanklang, W. 2018. Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. *EURO Journal on Transportation and Logistics*, 7(2): 151–192.

Danloup, N.; Allaoui, H.; and Goncalves, G. 2018. A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. *Computers and Operations Research*, 100: 155–171.

Drezner, Z.; and Hamacher, H. W., eds. 2002. *Facility location. Applications and theory*. Berlin: Springer.

Fu, Z.; and Chow, J. Y. 2022. The pickup and delivery problem with synchronized en-route transfers for microtransit planning. *Transportation Research Part E: Logistics and Transportation Review*, 157: 102562.

Li, H.; and Lim, A. 2001. A metaheuristic for the pickup and delivery problem with time windows. In *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*, 160–167.

Lyu, Z.; and Yu, A. J. 2022. The pickup and delivery problem with transshipments: Critical review of two existing models and a new formulation. *European Journal of Operational Research*, 305(1): 260–270.

Manier, H.; Godart, A.; Bloch, C.; and Manier, M.-A. 2019. A greedy based algorithm for a bi-objective Pickup and Delivery Problem with Transfers. In *Conference on Systems, Man and Cybernetics (SMC)*, 3229–3234. IEEE.

Masson, R.; Lehuédé, F.; and Péton, O. 2013. An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers. *Transportation Science*, 47(3): 344–355.

Mitrovic-Minic, S.; and Laporte, G. 2006. The pickup and delivery problem with time windows and transshipment. *Information Systems and Operational Research*, 44(3): 217–227.

OSM. 2022. OpenStreetMap. https://www.openstreetmap.org/copyright. Accessed: 2022-11-30.

Petersen, H. L.; and Ropke, S. 2011. The Pickup and Delivery Problem with Cross-Docking Opportunity. In Böse, J. W.; Hu, H.; Jahn, C.; Shi, X.; Stahlbock, R.; and Voß, S., eds., *Computational Logistics*, volume Lecture Notes in Computer Science, vol 6971, 101–113. Springer Berlin Heidelberg.

Qu, Y.; and Bard, J. F. 2012. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers and Operations Research*, 39: 2439–2456.

Rais, A.; Alvelos, F.; and Carvalho, M. S. 2014. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235: 530–539.

Ropke, S.; and Pisinger, D. 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40: 455–472.

Sampaio, A.; Savelsbergh, M.; Veelenturf, L. P.; and Van Woensel, T. 2021. Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. *Networks*, 76(2): 232–255.

Sassmann, V.; Rudová, H.; Gabonnay, M.; and Sobotka, V. 2023. Real-World Vehicle Routing Using Adaptive Large Neighborhood Search. In Pérez Cáceres, L.; and Stützle, T., eds., *Evolutionary Computation in Combinatorial Optimization*, 34–49. Springer Nature Switzerland.

Toth, P.; Vigo, D.; for Industrial, S.; and Mathematics, A. 2014. *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics.

Vidal, T.; Laporte, G.; and Matl, P. 2020. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2): 401–416.

Voigt, S.; and Kuhn, H. 2022. Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks*, 79(3): 403–426.

Wereldo. 2023. Wereldo.com. Accessed May 12, 2023. https://www.wereldo.com/.