A Data-Driven Three-Layer Algorithm for Split Delivery Vehicle Routing Problem with 3D Container Loading Constraint

Xijun Li
Shanghai Jiao Tong University
Noah's Ark Lab of Huawei
lixijun@sjtu.edu.cn

Mingxuan Yuan
Noah's Ark Lab
Huawei Technologies
yuan.mingxuan@huawei.com

Di Chen
Noah's Ark Lab
Huawei Technologies
di.chen@huawei.com

Jianguo Yao
Shanghai Jiao Tong University
jianguo.yao@sjtu.edu.cn

Jia Zeng
Noah's Ark Lab
Huawei Technologies
Zeng.Jia@huawei.com

ABSTRACT

Split Delivery Vehicle Routing Problem with 3D Loading Constraints (3L-SDVRP) can be seen as the most important problem in large-scale manufacturing logistics. The goal is to devise a strategy consisting of three NP-hard planning components: vehicle routing, cargo splitting and container loading, which shall be jointly optimized for cost savings. The problem is an enhanced variant of the classical logistics problem 3L-CVRP, and its complexity leaps beyond current studies of solvability. Our solution employs a novel data-driven three-layer search algorithm (DTSA), which we designed to improve both the efficiency and effectiveness of traditional meta-heuristic approaches, through learning from data and from simulation.

A detailed experimental evaluation on real data shows our algorithm is versatile in solving this practical complex constrained multi-objective optimization problem, and our framework may be of general interest. DTSA performs much better than the state-of-the-art algorithms both in efficiency and optimization performance. Our algorithm has been deployed in the UAT (User Acceptance Test) environment; conservative estimates suggest that the full usage of our algorithm would save millions of dollars in logistics costs per year, besides savings due to automation and more efficient routing.

*This work has been done when Xijun Li worked as intern at Noah's Ark Lab of Huawei.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219872

CCS CONCEPTS

• **Applied computing** → **Transportation**; • **Computing methodologies** → *Planning and scheduling*; *Machine learning approaches*;

KEYWORDS

Logistics; Transportation planning; Container loading

ACM Reference Format:
Xijun Li, Mingxuan Yuan, Di Chen, Jianguo Yao, and Jia Zeng. 2018. A Data-Driven Three-Layer Algorithm for Split Delivery Vehicle Routing Problem with 3D Container Loading Constraint. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219872

1 INTRODUCTION

Container transportation planning is a perpetually important problem to manufacturing enterprises. Huawei, one of the largest global telecommunication vendors, transports tens of billions of cargoes to around 180 countries/regions every year. Given an order for the delivery of a batch of cargoes, often produced in different factories, a fleet of heterogeneous vehicles (carrying containers of different types) are sent out to collect the cargo and ship them to a port for overseas shipping. A good transportation plan would take into account multiple aspects of the transportation process to reduce logistics costs. Huawei's logistics management estimates that a 3% reduction in the number of containers employed would correspond to millions of USD in savings.

This is the *Split Delivery Vehicle Routing with 3D Loading Constraints Problem* (3L-SDVRP). 3L-SDVRP aims to calculate the least-cost cargo transportation given a delivery order, where we simultaneously perform (1) the route planning of vehicles, (2) the allocating of cargoes at different collection points to different vehicles (3) the feasible packing of the transported items into the available loading space.

3L-SDVRP therefore consists of three sub-problems (Figure 1): The *Vehicle Routing Problem* (VRP) [7] whose objective is to decide the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers; (2) *Splitting Delivery Routing* (SDR) [4] where a delivery to a

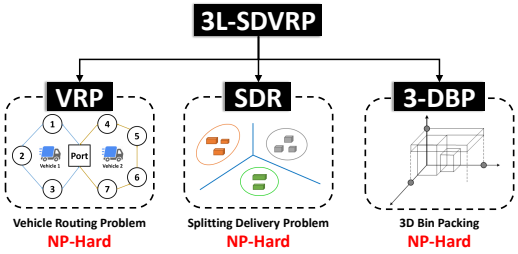


Figure 1: The proposed problem is comprised of three NP-hard sub-problems.

customer can be split between any number of vehicles; and (3) *3D Bin Packing* (3-DBP) [8] where a set N of 3D boxes is to be packed in a minimum number of containers (bins). Each of the sub-problems has been proven to be NP-hard[4, 7, 8], but we are further burdened with the combination of these problems and other practical process constraints. Much work has been devoted to VRP, SDR and 3D Bin Packing respectively or the combination of two of them; to the best of our knowledge, there is no work discussing 3L-SDVRP, the combination of all three.

Solving 3L-SDVRP is especially challenging due to the enormous combined solution space of three difficult combinatorial problems. In our scenario, the average delivery order contains 300+ cargoes distributed across the Pearl River Delta region, which is a problem scale that prohibits brute-force solutions. To develop a system that solves this problem in a short amount of time, we propose our **D**ata-Driven **T**hree-Layer **S**earch **A**lgorithm (DTSA). By making use of historical data that represent expert knowledge, efficient solution generation through probabilistic models, and fast container loading evaluation using a learning-based prediction model, the algorithm can produce good solutions with reasonable computational costs.

Our algorithm jointly optimizes the three sub-problems using three corresponding layers. The routing aspect of the problem is tackled by an *Estimation of Distribution Algorithm* (EDA), which is the outmost layer. The VRP solution space is efficiently represented by a generator distribution to improve the search efficiency; historical data rich in human expert knowledge are fused into this process as prior knowledge, to improve the quality of initial candidate solutions. Similar to route planning, an inner EDA deals with the cargo splitting to vehicles. The innermost algorithm solves the 3D loading problem. Besides directly using eight packing heuristics [13, 14] to solve the loading problem, we further design a novel supervised learning based strategy which can directly and quickly estimate the best possible loading effectiveness (without explicitly simulating and optimizing the loading), through a regression model trained from offline simulation. This design helps us rapidly explore the solution space with little impact on solution quality.

To summarize, we make the following contributions to 3L-SDVRP, and multi-objective optimization in general:

- A practical data-driven three-layer search algorithm (DTSA) for 3L-SDVRP: We design and deploy an efficient three-layer data-driven algorithm for 3L-SDVRP. To the best of our knowledge, this is the first attempt in literature to solve 3L-SDVRP; we provide a summary of the related work in Section 2.
- Learning from data: We collect and make use of historical routing data to estimate the hyperparameters of VRP probability matrix. This helps improve the quality of initial solutions for the EDA search.
- Omitting the rule designing step for *Evolutionary Algorithms* (EA): Classical EA requires the designing of a number of complex rules, guiding the initialization and transformation of candidate solutions. Instead, our algorithm makes use of EDA to learn from data or from simulation probability distributions capable of generating candidate solutions that are likely to be feasible, avoiding the need for manual specification of generation rules. This offers flexibility in applying EA to complicated business situations.
- Dramatically speeding up solution evaluation for optimization through machine learning. We learn an approximate scoring function for container loading, which replaces the costly subroutine of explicit solution evaluation through loading optimization. This greatly improves the efficiency of our algorithm while maintaining the quality of the final solution.

Our algorithm has been deployed in our company's UAT (User Acceptance Test) environment and is being promoted for wide adoption. Compared with the state-of-the-art searching algorithms [9, 11, 14], our algorithm runs four times faster with solutions that are at least 14.5% better. Dispatch engineers of Huawei's logistics system confirmed that our algorithm can save 8.9% containers over the old system, which is based on heuristics and human expert intervention, suggesting that the full deployment of our algorithm would save millions of dollars in logistics costs per year, besides savings due to automation and more efficient routing.

2 RELATED WORK

Recall that 3L-SDVRP is the combination of VRP, SDR and 3-DBP. Much work has been done on each or combinations of the two of them; we briefly summarize those past efforts.

VRP is one of the most well-studied combinatorial optimization problems. The goal of VRP is to calculate the optimal route planning for a fleet of vehicles to serve customers at different locations. It was first proposed by Dantzig et. al. [2] and solved by different approaches ranging from integer programming, heuristic algorithms, to metaheuristics.

Based on VRP, SDVRP deals with the scenario in which each customer may be served by multiple vehicles, which results in split deliveries. It was first introduced by Dror et. al. [4], who developed a heuristic algorithm for the problem. In [1], Archetti et. al. applied the tabu search heuristic to the SDVRP. Insertion moves are used for local improvement, with the possibility of inserting a customer into a route without removing it from another route. However, SDVRP assumes one-dimensional container loading, so the proposed solutions do not take into account the geometric constraints imposed by the three-dimensional nature of the problem.

3L-CVRP calculates feasible routes with the minimum total travel cost while satisfying customers' demands expressed in terms of cuboid and weighted items. Practical constraints related to stability, fragility, and Last-In-First-Out sequential loading can be considered for the problem. Most existing techniques for 3L-CVRP are based on TS combined with efficient packing heuristics [15]. Gendreau et al. [5] were the first to employ TS to tackle routing aspects of the problem, with the use of two packing heuristics to handle loading constraints. Tarantilis et al. [14] describe a hybrid metaheuristic methodology called GTS that combines the approaches of TS and guided local search (GLS). In contrast to SDVRP, 3L-CVRP takes into account the 3D loading of cargoes; on the other hand, more transportation savings can be obtained by allowing split deliveries as in SDVRP.

Yi et. al. [16] proposed a hybrid algorithm for a simplified version of 3L-SDVRP, where a tabu search procedure was employed for routing alongside some packing heuristics. Compared to 3L-SDVRP, fewer loading constraints are considered in [16]. A simple heuristic method is utilized to tackle cargo splitting in Yi's work, and there is some room for improvement for cargo splitting optimization.

These previous attempts only solved parts or simplified subscenarios of 3L-SDVRP, and without incorporating historical data to improve efficiency and optimization performance. In this work, we propose a novel data-driven three-layer algorithm for 3L-SDVRP in a practical industrial scenario.

3 PROBLEM DESCRIPTION

A delivery order consists of a set of cargoes distributed at multiple collection points. Given a delivery order, the goal is to devise a cargo transportation strategy to efficiently transport cargoes to the port with minimum transportation cost. Each vehicle starts from a single port, collects cargo from collection points, and returns to the port. The cargo transportation strategy should consist of three coupled parts: route planning, cargo splitting and container loading.

The input of the problem includes:

- A road network $G = (N, E)$, which is a complete directed graph. N is the vertex set consisting of the port 0 and n geographically dispersed collection points i $(i = 1, ..., n)$ with cargos, and $E = \{(i, j) : i, j \in N, i \neq j\}$ is the edge set. Each edge (i, j) has a non-negative cost c_{ij} which represents the transportation cost from collection point i to collection point j. Note that $c_{ij} = c_{ji}$.
- A delivery order $O(\{IT_i | i = 1, ..., n\})$, where each IT_i is a set of m_i 3D items (throughout this paper, the terms cargo and item are used interchangeably) of collection point i. The cargoes IT_i are to be delivered from collection point i to the port, using one vehicle or several vehicles. Splitting needs to be considered whenever some IT_i is to be delivered by several vehicles. For a vehicle v, we let $A_i^v \subseteq IT_i$ denote the set of collection point i's items allocated to v. The length, width, and height dimensions of an cargo $I_{ik} \in IT_i$ $(k = 1, ..., m_i)$ are denoted by l_{ik}, w_{ik}, and h_{ik}, respectively. The weight and volume of I_{ik} are denoted by q_{ik} and s_{ik} respectively. In addition, each item I_{ik} has a fragility status fr_{ik} whose value is 1 if I_{ik} is fragile and 0 otherwise. HR_{ik} and VR_{ik} are boolean variables which indicate whether I_{ik} can be rotated horizontally and vertically. MA_{ik} represents the packing material of I_{ik}, including paper, wood, metal etc.
- Configuration of each category k of containers. L_k, W_k and H_k denote the length, width and height of the container's interior, respectively. Q_k and S_k are the maximum carrying weight and the maximum carrying volume of the container. IC_k represent cubes around the corners where no cargoes can be put there. FE_k denotes the fee of the container. Due to space constraints, we do not present a full listing of all other descriptor variables, such as the minimum amount of space to reserve to allow for loading by a forklift.

In addition to the routing constraints of classical VRP [7], our problem is also subject to the following:

- Volume constraint. For each container k, the sum of loaded items' volume cannot exceed its S_k.
- Weight constraint. For each container k, the sum of loaded items' weight cannot exceed its Q_k.
- Support area constraint. Every item's bottom surface must be sufficiently supported by another item's top surface or by the container's bottom surface.
- Packing material constraint. A paper box cannot be put under a metal/wooden box or between two metal/wooden boxes.
- Sequentially loading constraint. Every item must be loaded and unloaded according to the Last-In-First-Out (LIFO) principle.
- Fragility constraint. No non-fragile item can be stacked on any fragile item.
- Non-overlapping loading constraint. There must be an orthogonal non-overlapping loading of the items into the vehicle's container.
- Orientation constraint. Items must be packed with a predetermined vertical orientation (i.e. "this side up")
- Other process constraints. There are many more process constraints in our industrial scenario. The detailed description of other process constraints is omitted for the sake of clarity.

The objectives of 3L-SDVRP are,

- Container costs: the number of containers should be as small as possible;
- Transportation efficiency cost: the distance traversed by the vehicles should be as small as possible.

4 ALGORITHM DESIGN

4.1 System Architecture

Figure 2 shows the algorithm implementation as it was deployed in the UAT system. For a given delivery order, we first estimate the minimum number nv of required vehicles with the assumption of container being fully loaded, calculated

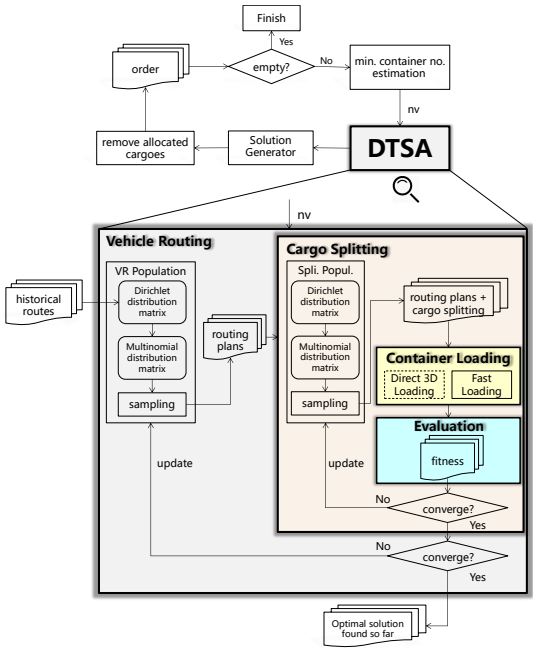


Figure 2: System architecture

according to Eq.(1).

$$nv = \max\left(\frac{\sum_{IT_i \in O} \sum_{I_{ik} \in IT_i} s_{ik}}{S}, \frac{\sum_{IT_i \in O} \sum_{I_{ik} \in IT_i} q_{ik}}{Q}\right), \quad (1)$$

where Q and S are the maximum carrying weight and the maximum carrying volume of the highest-cost container. Based on the minimum vehicle number nv, DTSA calculates the optimal cargo transportation strategy using only nv vehicles. It is possible that the nv vehicles can only transport a subset of cargoes. The algorithm may need several more epochs until all cargoes are covered.

The outermost layer of DTSA uses EDA to search for vehicle routing solutions. We model a *random solution generator* using a probability matrix of vehicle transitions. By sampling from the generator, we can get a set of routing plans. Given a routing plan, the inner layer further uses EDA to search for cargo-splitting solutions. The splittings are also sampled from a generator, represented as a probability matrix of cargo allocation. Sampling from this generator we obtain a set of candidate solutions, each with a routing plan and a cargo splitting plan. Then, for each candidate solution, the innermost layer uses a container loading module to estimate a 3D loading plan. Finally, based on the loading results, each solution will be evaluated to obtain its *fitness*.

The cargo splittings of solutions with good fitness are selected to update the generator distribution for the EDA for cargo splitting. When the inner layer EDA converges, the routing plans of solutions with good fitness are selected to update the generator distribution of the EDA for vehicle routing. The algorithm repeats the above process until the outermost EDA has converged. After that, the optimal cargo transportation strategy found so far for the nv vehicles is output as an intermediate result. The intermediate result is further processed to find the best container category to fit the current transportation strategy. Because the number of vehicles (and therefore containers) is constrained to be nv, not all cargoes may be allocated, and the remaining cargoes are used to generate a 'virtual' delivery order, fed as input for the next epoch.

The above will repeat until all intermediate outputs cover all cargoes in the initial delivery order, at which collection point the results are merged as the final output strategy. The benefit of using EDA is that learning the parameters of the probability matrix is more efficient than directly searching for individual solutions, also offering a convenient way to introduce expert knowledge.

4.2 Vehicle Routing

The outermost search algorithm is responsible for optimizing the routing plans using EDA. At first, a probability matrix representing a Dirichlet distribution is extracted from the historical routes to model vehicle transitions, as presented in Figure 3. This prior Dirichlet distribution will be gradually updated into to a posterior distribution, using information provided by the vehicle routing search algorithm. In this paper, we refer to the parameters of the probability matrix as *hyperparameters*. We will describe this step in more detail in the following subsection. As shown in Figure 3, the searching algorithm is composed of the following steps and terminates when a stopping criterion is satisfied.

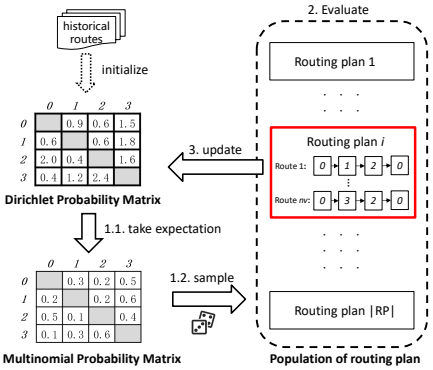


Figure 3: The process of vehicle routing search

(1) Sample routing plan (routes): We generate a multinomial distribution matrix by taking the expectation of the underlying Dirichlet distribution. Each element $p_{i,j}$ in the multinomial distribution matrix represents the probability that a vehicle transition from collection point i to collection point j. With this matrix, a population RP of routing plans can be sampled.

(2) Evaluate the routing plan: Each routing plan i of the population RP is evaluated alongside its optimal cargo splitting plan A_i^* obtained from the cargo splitting module. When the cargo splitting module converges, i receives a fitness score $f(i)$.

(3) Update the generator distribution: As shown in Figure 3, the routing plans (boxed in red) with better fitness contribute to the updating of the Dirichlet distribution probability matrix (to increase the probability that similar 'good' routing plans are sampled in the future). The probability p_i that an individual i is selected

is calculated via Eq.(2) based on its fitness.

$$p_i = \frac{e^{-f(i)}}{\sum_{j \in RP} e^{-f(j)}} \quad (2)$$

An update to the Dirichlet matrix adds the number of valid transitions appearing in those selected routing plans to the corresponding element in the matrix. We illustrate this process in Figure 4, where the valid transition $2 \rightarrow 0$ appears twice in the selected routing plan, so the element in row 0 and column 2 of the matrix will be added two, updated as 4.0.

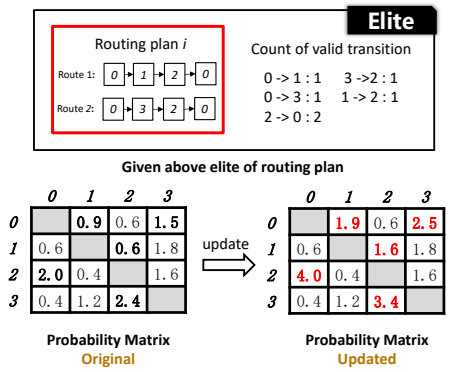


Figure 4: Parameter update of Dirichlet distribution

4.2.1 Initializing Generator Distribution from Historical Data. To initialize the probability matrix for the generator distribution of routing plans, a Dirichlet distribution is extracted from historical routing plans decided by human experts (Huawei's dispatch engineers). This allows us to incorporate human knowledge, and provide initial populations of high-quality solutions for the outmost EDA. The parameters of the probability matrix are then updated as above in a Bayesian manner. We take the following steps to generate the initial Dirichlet generator distribution.

(1) Extracting relevant historical routes. We extract the historical routes which contain at least one path in the current delivery order (so at least two collection points in the current order are connected with a path)
(2) Constructing the sample data. We sort the selected historical routes by their timestamps and add the valid transitions (two adjacent collection points are both in the current order) in every 100 routes to a bundle. Each bundle of transitions is a sample for estimating the parameters of the Dirichlet distribution.
(3) Hyperparameter estimation. We use the maximum likelihood estimation method [10] to estimate the Dirichlet distribution.

4.3 Cargo Splitting

After the vehicle routing module generates a population of routing plans, we continue to use EDA to search for the optimal cargo splitting strategy for these routing plans. We cluster cargoes based on multiple features (e.g. length, width, height and weight). Then for each collection point, we build a distribution matrix which represents the probability that each cluster of cargoes should be allocated to each route. When the distribution matrices of all collection points converge, the cargo-splitting solutions are then generated from these matrices. Given a routing plan and a collection point, the cargo splitting module contains the following steps.

(1) Generator distribution: Similar to the process of generating a routing plan, cargo splitting plans are also sampled from a multinomial distribution defined through a Dirichlet distribution matrix. We first construct a uniform Dirichlet distribution for each collection point, which means that at the beginning, each cluster of cargoes has equal chance to be allocated to each route.
(2) Sampling cargo splitting plans: For a collection point, a multinomial distribution matrix can be computed from the expectations of its Dirichlet distribution. Then a set of partial cargo splitting plans of this collection point can be sampled from the multinomial distribution.
(3) Cooperative evaluation of cargo splitting: By combining the partial cargo splitting plans of all collection points, we can get an integrated splitting plan. The integrated splitting will be evaluated by invoking the container loading module. The above process is called *Cooperative Co-Evolution Algorithm* (CCEA) [12], where each individual of one subpopulation is evaluated together with representative individuals of other subpopulation.
(4) Updating the probability matrices: When we get the fitness scores of splitting plans based on the loading result, we select the 'good' individuals to update probability matrices. Similar to the vehicle routing module, an individual is selected with a probability based on its fitness, using the same formula Eq.(2). We use the same update strategy as the vehicle routing module to update the probability matrices here.

4.4 Container Loading

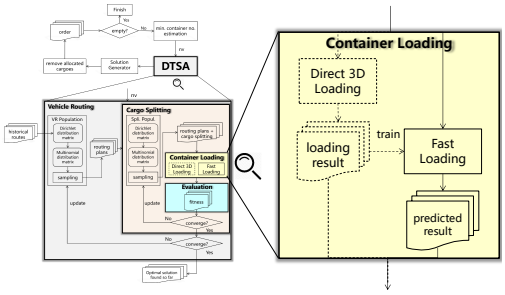


Figure 5: Container loading module

The two EDA algorithms for vehicle routing and cargo splitting generates a set of plans, with routing plans and cargo splits. In the innermost layer, we should evaluate how good these plans are when loading them into containers.

Given a route, a good loading plan should have high volume utilization rate r_{VOL_v} and weight utilization rate r_{WGH_v}. The combination of these two rates are used to evaluate the fitness of container loading. As shown in Figure 5, we design two solutions for this module. One does the same as [5, 14], where we use a combination of eight practical 3D loading

heuristics to generate exact loading plans and obtain the two utilization rates. We call this solution *Direct 3D Loading*. If only part of cargoes allocated to a vehicle can be loaded, we change the solution by only allocating the cargoes that can be loaded to his vehicle. The other is making use of a learned model to perform fast evaluation, which we call *Fast Loading*. Experiments verify that either of these two solutions generates much better results than the state-of-the-art algorithms. Fast Loading runs four times faster than Direct Loading with some loss in loading effectiveness. We deployed the fast evaluation solution due to high efficiency requirements.

4.4.1 Fast Loading. The Direct 3D loading is a sequential process, which needs to load cargoes one by one into the container, thus cannot be effectively parallelized. In a deployed system, the total time spent on loading over the whole process is in the range of 300-600 seconds per container, which is a major source of the time consumption.

We observe that during optimization, we do not need to explicitly construct loading plans, if we can predict the loading effectiveness; we only need the detailed loading plans together with routes and cargo splitting when we output the final result. This motivated us to design a regression model to directly estimate r_{VOL_v} and r_{WGH_v} without having to run a full optimization subroutine for cargo loading. Our idea is to train a regression model from large-scale simulated data, which will help us make use of offline computation power to greatly speed up online optimization.

In the offline training stage, we repeatedly run DTSA with Direct 3D Loading to generate simulated data, where for each vehicle v, a piece of simulated data consists of its cargo splitting plan A^v and the two utilization rates r_{VOL_v}, r_{WGH_v}. The data can be written as

$$[(1, A_1^v), ..., (i, A_i^v), ..., (k, A_k^v), r_{VOL_v}, r_{WGH_v}], \tag{3}$$

where i is a collection point visited by vehicle v, A_i^v is the set of cargoes belonging to i but allocated to v, and k is the number of collection points visited by v. We train regression models from these data to predict r_{VOL_v} and r_{WGH_v}, whereupon we use linear programming to determine the set of cargoes loaded into the vehicle.

Utilization Rate Prediction. We train two regression models for volume and weight utilization rate respectively. The features for the prediction models are designed as follows.

(1) Cargo features. For a cargo $I_{ik} \in A_i^v$, its features $\vec{f}_{I_{ik}}$ include: length, width, height, weight, minimum touching area, which are real variables, as well as material and pressure coefficients which are one-hot encoded categorical variables.
(2) Collection point features. $\vec{f}_i^v$ is the feature of the visited collection point i. $\vec{f}_i^v$, consisting of the mean and covariance of $\vec{f}_{I_{ik}}$ $(I_{ik} \in A_i^v)$, the counts of various categories in A_i^v and a quantity t_i that we will describe shortly.
(3) Splitting feature. The splitting feature $\vec{O}_{A^v}$ is the weighted sum of $\vec{f}_i^v$ $(i = 1, ..., k)$. It is computed by $\vec{O}_{A^v} = \sum_{i=1}^{k} r_i \vec{f}_i^v$, where r_i is the weight. Based on historical route records, r_i and t_i are calculated using Correspondence Analysis (CA) [6].

The prediction model is experimentally selected, from common models including Support Vector Regression (SVR), Gradient Boosted Decision Tree (GBDT), Xgboost, and Neural Network (NN), and we found that GBDT offers the best performance. See Section 5.

Cargo Loading Plan Prediction. Since it is possible that only part of the assigned cargoes can be loaded into a vehicle, we design a heuristic based on a Linear Programming (LP) relaxation to confirm the loaded cargoes.

Let p_k denote the probability that the kth cargo $I_k \in A^v$ can be loaded into vehicle v. Based on the predicted $\tilde{r}_{VOL_v}$ and $\tilde{r}_{WGH_v}$, we solve the following LP:

$$Minimize \quad S_{load} + Q_{load} - \sum_{I_k \in A^v} (s_k + q_k) p_k, \tag{4}$$

subject to

$$\sum_{I_k \in A^v} s_k p_k \leq S_{load}, \tag{5}$$

$$\sum_{I_k \in A^v} q_k p_k \leq Q_{load}, \tag{6}$$

$$Q_{load} = Q \cdot \tilde{r}_{WGH_v}, \tag{7}$$

$$S_{load} = S \cdot \tilde{r}_{VOL_v}, \tag{8}$$

where $p_k \in [0, 1]$ are decision variables. After solving the LP, we generally obtain fractional solutions to the p_k's, on which we perform LP rounding in the following way: we select cargoes in descending order of p_k until S_{load} and Q_{load} are exceeded. The selected set of cargoes are then assumed to be loaded.

4.5 Solution Evaluation

A solution to 3L-SDVRP includes a routing plan and its corresponding optimal cargo splitting. The transportation costs resulting from such a plan can be measured in terms of the following metrics:

(1) Travel complexity of the fleet of vehicles.
(2) Number of used containers.

The travel complexity is the function of the route length rl_v that vehicle v traverse. By observing historical records of loading pattern, we found that a container of vehicle v is fully used if 1) The volume utilization rate r_{VOL_v} and the weight utilization rate r_{WGH_v} are as large as possible 2) there is a balance between r_{VOL_v} and r_{WGH_v}.

Based on the above considerations, we design two fitness functions F_1 and F_2 to evaluate any given solution, presented in Eq.(9) and Eq.(10) respectively.

$$F_1 = \sum_{v=1}^{nv} tl_v, \tag{9}$$

$$\begin{aligned} F_2 = &- \sum_{v=1}^{\text{nv}} (r_{VOL_v} + r_{WGH_v}) \\ &+ \sum_{v=1}^{\text{nv}} |r_{VOL_v} - \overline{r_{VOL}} - r_{WGH_v} + \overline{r_{WGH}}|, \end{aligned} \tag{10}$$

where $\overline{r_{VOL}}$ and $\overline{r_{WGH}}$ are respectively the average volume utilization rate and the average weight utilization rate of the fleet, as calculated by Eq.(11) and Eq.(12):

$$\overline{r_{VOL}} = \frac{\sum_{i=1}^{n}\sum_{k=1}^{m_i} s_{ik}}{\mathrm{nv}\cdot S}, \tag{11}$$

$$\overline{r_{WGH}} = \frac{\sum_{i=1}^{n}\sum_{k=1}^{m_i} q_{ik}}{\mathrm{nv}\cdot Q}. \tag{12}$$

Both F_1 and F_2 are supposed to be minimized in the algorithm. Note that this is a multi-objective optimization problem with possibly conflicting objectives, so it makes more sense to pursue *Pareto*-optimality which instead offers *optimal trade-offs* between objectives.

4.6 Solution Generator

The three-layer search algorithm generates the best transportation strategies (on pareto curve) using nv vehicles, under the assumption that the highest-cost container is used. There are three types of containers, and we invoke a subroutine to replace the container with lower-cost ones by a direct 3D loading algorithm. If a lower-cost container is feasible, we'll use it to replace the highest-cost one. We get a partial solution in this step.

Strategies calculated by the search algorithm try to transport as many cargoes as possible using only nv vehicles. It is possible that some cargoes cannot be transported, in which case we generate a 'virtual' delivery order for these cargoes, and repeatedly invoke the searching algorithm and solution generation until no cargo remains undelivered. By merging all partial solutions, we get the final transportation strategy for the input delivery order.

5 EVALUATION

5.1 Experiment Environment

Our test dataset is extracted from the historical delivery orders dated 2017, where 30 delivery orders are sampled from each month. This gives 360 delivery orders, which differ in the number of cargoes, number of cargo categories and number of collection points. We present a statistical summary of the data set in Table 1.

Table 1: Test data statistics

	# categories	# items	# collection points
Mean	94.68	307.13	5.836
Median	35	219	5
Sum	64596	158321	2525

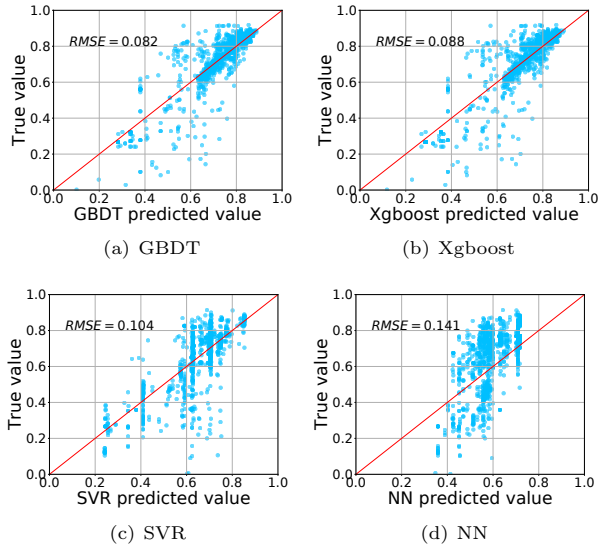
We compare our algorithm against the historical performance (heuristics + human expertise), as well as our implementation of three state-of-the-art algorithms that were proposed to solve problems closely resembling 3L-SDVRP: *Simple Genetic Algorithm* (SGA) [9], *Discrete Particle Swarm Optimization* (DPSO) [11], and *Tabu Search* (TS) [14].

DPSO performs a two-level discrete particle swarm optimization, where an outer DPSO explores the routing aspects and an inner DPSO searches for the optimal cargo splitting for a given routing plan. SGA and TS were designed for 3L-CVRP [9, 14] but we adapt them for our problem. All the above methods use the same 3D heuristic loading algorithms as ours do.

Since our problem has two optimization objectives, we also compare the performance of algorithms under four major multi-objective fitness assignment methods, including *Non-dominated Sorting Genetic Algorithm* (NSGA) [3], *Strength Pareto Evolutionary Algorithm* (SPEA) [19], *Indicator-Based Selection* (denoted as $I_{\epsilon+}$) [18], and MOEA/D [17]. We use the best tuned parameters for each algorithm we compare against, whose details we omit due to space constraints.

The algorithms are parallelized on a Spark cluster using the CentOS 7 operating system equipped with 150 cores, $512G$ memory.

5.2 Loading Estimation Model Selection



(a) GBDT (b) Xgboost

(c) SVR (d) NN

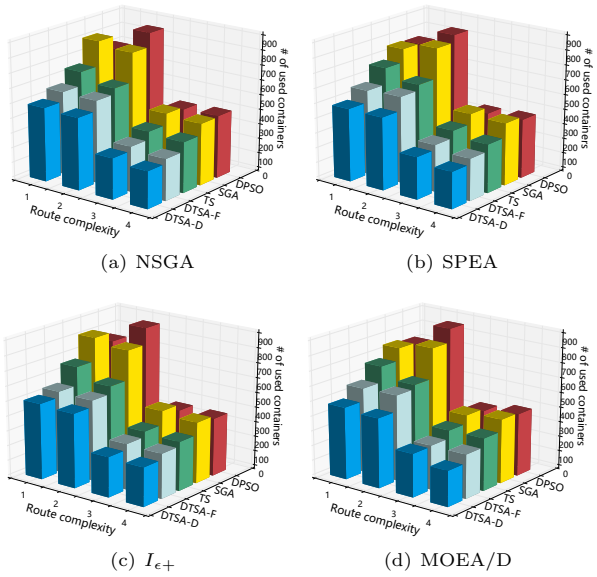
Figure 6: Prediction accuracy of regression models

We collected 5595 loading records to train the fast loading module using various regression models such as GBDT, XGBoost, SVR and NN, and the results of 10-fold cross validation are shown in Figure 6. The X axis represents the true value of the container utilization rate, and Y axis is the value predicted by the regression model. We chose GBDT based on comparing RMSE shown in the figures.

5.3 Optimization Performance

In this section, we compare the optimization performances of different algorithms. Since the problem has two objectives, we compare one objective when the other objective is held roughly constant. To do this efficiently, we normalize the transportation distances and divided them equally into four separate bins, and for each bin we compare the number of containers required by the solution of each algorithm. In order to study the influence of fast loading, we list the results of our algorithm both with and without fast loading. The algorithm with fast loading is denoted by DTSA-F (**D**ata-driven **T**hree-Layer **S**earch **A**lgorithm with **F**ast Loading),

and the latter is referred to as DTSA-D where the final D means **D**irect 3D Loading.



(a) NSGA (b) SPEA

(c) $I_{\epsilon+}$ (d) MOEA/D

Figure 7: No. of containers calculated by different algorithms with various fitness assignment methods

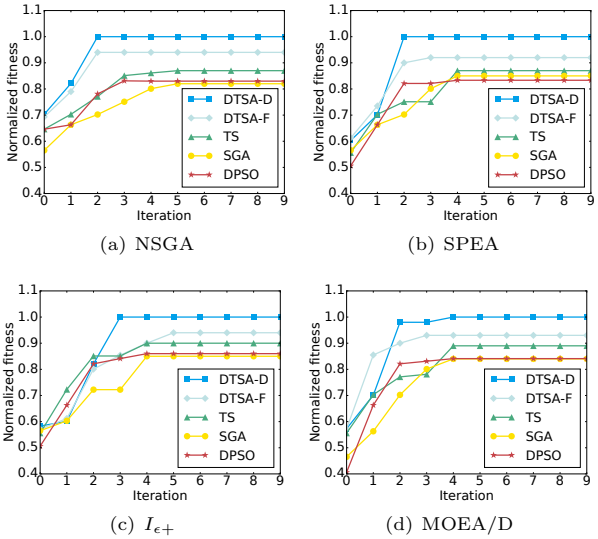
We recorded the average of five runs of each algorithm to limit the effect of randomness. Figure 7 presents the experimental results under different multi-objective methods. The axis of 'route complexity' denotes 4 bins of the normalized travel distances. One can observe the following from Figure 7.

(1) Regardless of the multi-objective method, the numbers of containers calculated from our algorithms are less than that calculated by TS, SGA and DPSO, also under any route complexity. In particular, DTSA-D performs the best in this regard; the total numbers of containers given by DTSA-D are 27.1%, 38.7% and 39.2% less than that of TS, SGA and DPSO respectively.
(2) For any route complexity, the number of containers calculated by DTSA-F is slightly larger than that calculated by DTSA-D, it is still 14.5%, 31.6% and 22.9% less than that calculated by TS, SGA and DPSO respectively.
(3) For all algorithms, the total number of containers is the smallest when using the multi-objective method NSGA, compared to SPEA, $I_{\epsilon+}$ and MOEA/D.

We see that DTSA-D and DTSA-F gives much better results in terms of container costs. DTSA-F gives slightly worse results than DTSA-D, due to prediction errors in the fast loading model; in return for this relatively small loss of solution quality, DTSA-F offers a significant gain in speed, which we demonstrate in the next part.

5.4 Efficiency Evaluation

5.4.1 Convergence. First we compare the convergence speed of DTSA-D and DTSA-F with that of SGA, TS, DPSO under different multi-objective methods, as summarized in



(a) NSGA (b) SPEA

(c) $I_{\epsilon+}$ (d) MOEA/D

Figure 8: Convergence property of different algorithms with various fitness assignment methods

Figure 8. In the figure, the X axis denotes the number of iterations until convergence, and the Y axis denotes the normalized solution fitness in terms of the number of containers, taking values in $[0, 1]$. We observe that:

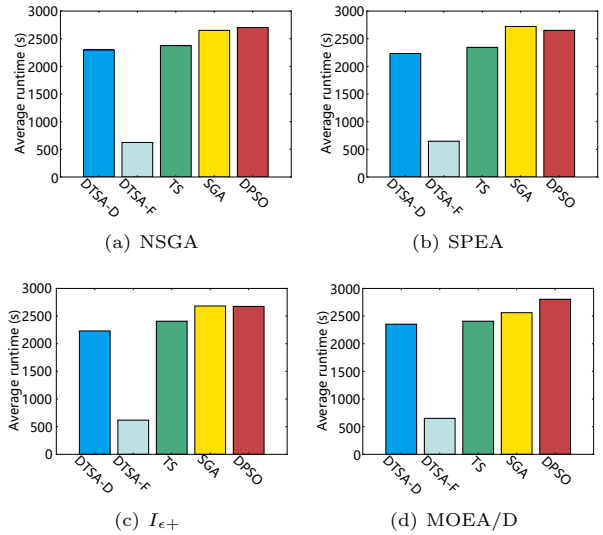
(1) Historical routing data benefits the fitness (quality) of the initial solutions given by DTSA-D and DTSA-F, which are always better than that of TS, SGA and DPSO except for the MOEA/D multi-objective control method.
(2) For any multi-objective method, the average fitness of the solutions calculated by DTSA-D is the highest among all algorithms when converging.
(3) DTSA-D converges the fastest among these algorithms, followed by TS.
(4) There is no obvious difference in convergence rates among different multi-objective methods.

5.4.2 Execution Time. We then compare the total execution times of different algorithms. We calculate the average execution time on each order and report the results in Figure 9, where we can observe that:

(1) Regardless of the multi-objective method used, the average execution times of DTSA-D and DTSA-F are much shorter than that of TS, SGA and DPSO. Most prominently, the average execution time of DTSA-F is only one quarter of that of DTSA-D, around 600 seconds for one delivery order, whereas TS, SGA and DPSO need around 2500 seconds on average.
(2) There is no significant difference in execution time using different multi-objective methods.

From the experimental results, we can conclude that:

(1) DTSA-D and DTSA-F outperform other benchmark algorithms significantly in both effectiveness and efficiency.



(a) NSGA (b) SPEA

(c) $I_{\epsilon+}$ (d) MOEA/D

Figure 9: Execution time comparison

(2) DTSA-D can save 12.7% more containers than DTSA-F with four times the time consumption. Given there was a 15-minute time constraint for processing and optimizing each delivery order, DTSA-F was recommended over DTSA-D for deployment.

In UAT, we also compared the performance of our algorithm with the old system (heuristic algorithms + human expert adjustment). Dispatch engineers of Huawei's logistics system confirmed that the proposed algorithm can save more than 8.9% containers over old system, attained by the best strategies by human experts with domain knowledge. Besides, we also explore the performance of DTSA without expert knowledge, whose result reveals that DTSA without prior knowledge is still better than the benchmark algorithms in terms of accuracy and efficiency but slightly worse than integrated DTSA.

6 CONCLUSION

In this work, we propose a novel data-driven three-layer search algorithm to solve the 3L-SDVRP problem under practical process constraints. Experiments show our algorithm performs much better than the benchmark algorithms and the old system, which represents the best solution obtained by human experts. The algorithm has been deployed in UAT environment to do comprehensive evaluation, and is awaiting full end-to-end integration with Huawei's large and complex logistics management system. Once the deployment process is complete, we expect to realize the potential promised by our testing and evaluation. In the future, we may refine our approach using reinforcement learning based approaches for better results.

ACKNOWLEDGMENTS

The authors would like to fully appreciate Dr. Haoyang Chen for providing his brilliant idea and valuable insight as well as Riemann Data Studio, FAT Process Department and Logistics Department of Huawei Supply Chain. The authors would also like to thank three anonymous referees for their valuable comments and helpful suggestions. This work was supported in part by National Key Research & Development Program of China (No. 2018YFB1003603), the Program for NSFC (No. 61772339), and the Shanghai Rising-Star Program (No.16QA1402200). The corresponding authors are Dr. Yuan, Mingxuan, Dr. Yao, Jianguo and Dr. Zeng, Jia.

REFERENCES

[1] Claudia Archetti, Martin WP Savelsbergh, and M Grazia Speranza. 2008. To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review* 44, 1 (2008), 114–123.
[2] George B Dantzig and John H Ramser. 1959. The truck dispatching problem. *Management science* 6, 1 (1959), 80–91.
[3] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature*. Springer, 849–858.
[4] Moshe Dror and Pierre Trudeau. 1990. Split delivery routing. *Naval Research Logistics (NRL)* 37, 3 (1990), 383–402.
[5] Michel Gendreau, Manuel Iori, Gilbert Laporte, and Silvano Martello. 2006. A tabu search algorithm for a routing and container loading problem. *Transportation Science* 40, 3 (2006), 342–350.
[6] Michael J Greenacre. 1984. Theory and applications of correspondence analysis. (1984).
[7] Jan Karel Lenstra and AHG Kan. 1981. Complexity of vehicle routing and scheduling problems. *Networks* 11, 2 (1981), 221–227.
[8] Silvano Martello, David Pisinger, and Daniele Vigo. 2000. The three-dimensional bin packing problem. *Operations Research* 48, 2 (2000), 256–267.
[9] Lixin Miao, Qingfang Ruan, Kevin Woghiren, and Qi Ruo. 2012. A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. *RAIRO-Operations Research* 46, 1 (2012), 63–82.
[10] Thomas Minka. 2000. Estimating a Dirichlet distribution.
[11] Quan-Ke Pan, Ling Wang, M Fatih Tasgetiren, and Bao-Hua Zhao. 2008. A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. *The International Journal of Advanced Manufacturing Technology* 38, 3-4 (2008), 337–347.
[12] Mitchell A Potter and Kenneth A De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257.
[13] Yi Tao and Fan Wang. 2015. An effective tabu search approach with improved loading algorithms for the 3L-CVRP. *Computers & Operations Research* 55 (2015), 127–140.
[14] Christos D Tarantilis, Emmanouil E Zachariadis, and Chris T Kiranoudis. 2009. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on Intelligent Transportation Systems* 10, 2 (2009), 255–271.
[15] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 2013. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research* 231, 1 (2013), 1–21.
[16] Junmin Yi and Andreas Bortfeldt. 2018. The Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints and Split DeliveryA Case Study. In *Operations Research Proceedings 2016*. Springer, 351–356.
[17] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation* 11, 6 (2007), 712–731.
[18] Eckart Zitzler and Simon Künzli. 2004. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*. Springer, 832–842.
[19] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report* 103 (2001).