Computers and Artificial Intelligence; Vol.3, No.4; 2026
E-ISSN: 3006-3728 P-ISSN: 3006-371X
DOI: https://doi.org/10.70267/cai.v3n4.105126
Published by: Zeus Press

# An Agent for Collaborative Optimization of Route Planning and Three-Dimensional Loading in Land Logistics

**Jingxiang Wang[1†]*, Xinyue Cai[2†], Yufeng Xin[3†], Bujia Ye[4] and Yingqi Li[5]**

[1] *College of Artificial Intelligence, Shenyang Aerospace University, Shenyang, 110136, China*

[2] *International Economics and Trade, Ningbo University of Finance and Economics, Ningbo, 315032, China*

[3] *School of Software, Shandong University, Jinan, 250101 China*

[4] *College of Transportation, Chongqing Jiaotong University, Chongqing, 400074, China*

[5] *School of Integrated Circuits, Huazhong University of Science and Technology, Wuhan, 430074, China*

[†]*These authors contributed equally to this work.*

**Corresponding author: Jingxiang Wang.*

## Abstract

Cross-border trade is growing fast. The same is true for urban distribution, e-commerce fulfillment, port transfer, and cold-chain transportation. Because of that, land-logistics decisions need to be workable in practice and flexible enough to adjust, especially in heavy-traffic places like Singapore. This study sets out to build an intelligent agent that plans vehicle routes and handles three-dimensional cargo loading at the same time. The agent treats routing as a capacitated vehicle routing problem with time windows. For small and medium cases, it turns the subproblems into quadratic unconstrained binary optimization so simulated-annealing or a quantum-inspired solver can be used. Bigger cases rely on heuristics and metaheuristics instead. The approach uses Solomon I1 insertion and 2-opt. Relocate is applied too. On top of that, it runs adaptive large neighborhood search and genetic algorithms. For cargo loading, the setup follows a three-dimensional bin-packing problem. An extreme-point heuristic checks cargo dimensions and weight. It considers orientation as well. The method accounts for stacking rules, non-overlap, load balance, and unloading-order constraints. It uses a feedback loop too. That loop sends the reasons for loading infeasibility back to the routing module, which then regroups customers or assigns a different vehicle. In a 50-customer routing case, Solomon I1 with local search found a feasible plan with nine vehicles and a total travel time of 129.0. Adaptive large neighborhood search kept the fleet at nine vehicles but cut total travel time to 92.0. This is about a 28.7% improvement. The results suggest that tying routing and loading together in a closed-loop setup makes plans easier to carry out. It improves vehicle utilization and delivery punctuality, and it makes the decisions easier to explain. This proposed agent offers a practical way to manage land-logistics optimization in one integrated system that can be adjusted on the fly.

## Keywords

intelligent logistics, agent, vehicle routing problem, QUBO, three-dimensional bin packing

## 1. Introduction

Logistics systems are a fundamental part of the modern economy. Manufacturing supply chains, urban retail networks, cross-border e-commerce fulfillment, and port and airport cargo transfers all rely on stable, efficient, and predictable transportation networks. For China and Singapore, logistics not only connects production with consumer demand but also supports regional trade, international shipping, and urban service systems. As economic scale expands and customer demand becomes increasingly personalized, the complexity of logistics decision-making rises significantly: customer locations become more dispersed, order batches become smaller, delivery time requirements become stricter, and road traffic conditions become more dynamic.

Traditional logistics scheduling usually depends on manual experience. Dispatchers arrange routes according to customer locations, vehicle capacity, and order priority, and then make temporary adjustments during execution. This approach can work when the number of customers is small or when traffic conditions are stable. However, in high-density urban traffic environments such as Singapore, the limitations of manual dispatching are clear. First, traffic conditions change quickly, and static routes can easily be affected by congestion, accidents, road construction, or port-area queues. Second, customer time windows are often tight. Some stores, warehouses, or terminals receive goods only during specific periods, and an improper visiting sequence can cause lateness. Third, vehicle resources are limited, and different vehicle types vary in compartment dimensions, payload, temperature-control capability, and access permissions. Fourth, cargo itself may involve complex loading constraints. For example, large items cannot be tilted, fragile goods cannot be pressed by heavy goods, cold-chain goods require refrigerated vehicles, and irregularly sized cargo can fragment the usable space inside the compartment.

Therefore, route optimization that focuses only on the shortest distance or the minimum number of vehicles cannot fully satisfy actual logistics needs. A route that appears optimal may still be impossible to execute if the goods on that route cannot be loaded into the vehicle or if the loading sequence conflicts with the unloading sequence. Conversely, if the system pursues vehicle fullness only from the loading perspective, it may cause detours, customer lateness, or excessively long routes. Route planning and loading optimization are strongly coupled and must be coordinated within a unified framework.

The agent proposed in this paper is designed for this problem. It is not a single algorithm, but a coordinating system that can perceive data, select algorithms, invoke modules, verify constraints, and adjust decisions through feedback. It treats vehicle route planning and three-dimensional loading optimization as two core submodules and establishes a feedback loop between them. The agent first reads customer, vehicle, cargo, and traffic data, generates initial customer groups and route plans, and then calls the 3D loading module to check whether the cargo assigned to each vehicle can be loaded. If loading is infeasible, the agent converts the infeasibility reason into a penalty or adjustment rule and invokes the route planning module again until a complete solution satisfying both routing and loading constraints is obtained.

The significance of this study is reflected in three aspects. First, at the theoretical level, this paper discusses vehicle route planning, QUBO modeling, simulated annealing, and three-dimensional bin packing within the same agentic framework, providing a structured approach for collaborative solution of combinatorial optimization problems. Second, at the methodological level, this paper incorporates loading feasibility into the route planning objective through a feedback loop, rather than treating loading as an after-the-fact check. Third, at the application level, this paper uses Singapore land logistics as the scenario and shows that the agent can be applied to urban delivery, port transfer, airport cargo transportation, e-commerce logistics, cold-chain transportation, retail replenishment, and large-item delivery.

## 2. Research Background and Problem Source

With the development of transportation technology, information technology, and artificial intelligence, intelligent logistics has become a key direction in modern supply-chain systems. Intelligent logistics emphasizes the use of data, algorithms, and automated systems to optimize the full logistics process, including order reception, warehouse management, vehicle scheduling, route planning, loading allocation, in-transit monitoring, and exception handling.

Compared with traditional logistics, its core features are real-time perception, intelligent decision-making, and explainable execution.

These features are especially important in Singapore land logistics. Singapore has limited land area, a dense road network, and clear spatial constraints. It also supports international shipping, airport freight, regional trade, and urban distribution. Ports, airports, industrial parks, commercial centers, and residential areas generate a large amount of short-distance, high-frequency, time-sensitive cargo movement. In such scenarios, vehicle scheduling must consider not only distance and time, but also congestion, customer time windows, vehicle capacity, compartment dimensions, cargo attributes, and carbon-emission goals.

The vehicle routing problem (VRP) originated from the truck dispatching problem. In its basic form, a depot and a set of customer nodes are given. Several vehicles depart from the depot, visit all customers, and return to the depot while minimizing total travel cost. As practical constraints are added, VRP has developed into many variants, including the capacitated VRP (CVRP), the VRP with time windows (VRPTW), the capacitated VRP with time windows (CVRPTW), the pickup and delivery VRP (VRPPD), the green VRP (GVRP), and the dynamic VRP (DVRP).

The agentic coordination concept and the QUBO-oriented routing formulation used in this study are informed by the course material and the MathorCup problem background [1,2]. The classical VRP formulation traces to Dantzig and Ramser [3]. Route construction and local-search design draw on Solomon's insertion heuristic and later VRPTW reviews [4,5], while simulated annealing provides a stochastic acceptance mechanism [6]. General vehicle-routing heuristics and standard VRP treatments provide broader methodological context [7,8], and column-generation methods constitute an exact/decomposition benchmark for large instances [9]. Genetic algorithms [10], adaptive large neighborhood search [11], and extreme-point three-dimensional bin packing [12] underpin the remaining optimization modules.

In this paper, the route planning problem includes not only vehicle capacity and customer time windows, but also real-time traffic conditions and loading feasibility. Let the depot be node 0, the customer set be N = {1, 2, ..., n}, and the vehicle set be K. Vehicle route planning must decide which customers each vehicle visits and in what order. A route can be expressed as:

$$R_k = (0, i_1, i_2, \dots, i_m, 0), \quad k \in K, \tag{1}$$

where $R_k$ denotes the route of vehicle k. Each route must satisfy the capacity constraint:

$$\sum_{i \in R_k} q_i \le Q_k, \tag{2}$$

where $q_i$ is the demand of customer i, and $Q_k$ is the capacity limit of vehicle k.

Customers may also require service to start within a time window $[a_i, b_i]$. If a vehicle arrives too early, it may have to wait; if it arrives too late, it may cause service failure, customer complaints, or penalty costs. The time-window constraint can be written as:

$$a_i \le s_i \le b_i, \quad \forall i \in N, \tag{3}$$

where $s_i$ denotes the service start time of customer i.

Meeting these conditions still does not guarantee that the plan can actually be carried out. For vehicle k, suppose the cargo assigned to it is the set $P_k$. The total weight of the goods in Pk can stay within the capacity limit, but the items might still not fit inside the compartment. Take two long items as an example. Their combined volume could be smaller than the compartment volume. But if each item is longer than the vehicle width and rotation is not possible, they still cannot sit side by side. Mixing heavy items with fragile ones can be hard to do in practice because stacking rules limit what can go where. So, this paper handles the three-dimensional loading problem as a route-planning constraint from the start, not as something to check afterward.

## 3. Research Objectives and Overall Approach

This paper aims to develop an agent that improves multi-scenario logistics transportation strategies. The agent should produce practical vehicle routes and cargo loading plans, even with complex constraints. This goal breaks down into four smaller objectives.

A vehicle route planning model is built as the starting point. It covers the basics, like making sure each customer is visited only once and that every vehicle leaves and returns as required. The routes need to stay

continuous. Vehicle capacity limits are included, along with customer time windows and transportation cost. The approach depends on how big the problem is. Small- and medium-scale problems can be converted into QUBO and solved with simulated annealing. For larger problems, heuristics and metaheuristics are used.

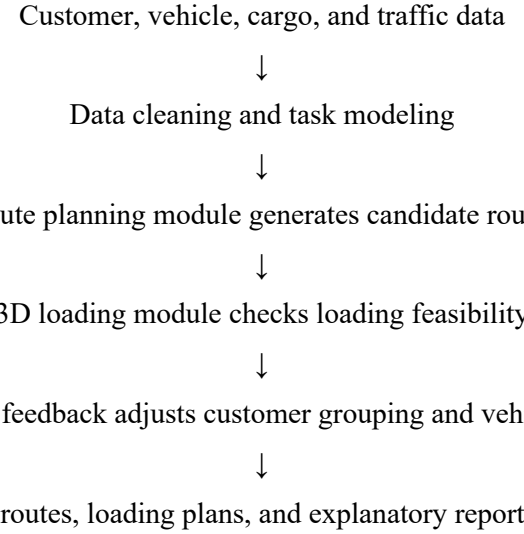
A three-dimensional loading optimization model is built next. It needs to capture cargo dimensions and weight. Rotation direction matters, and so do stacking restrictions. The model must reflect the vehicle's internal space and its maximum payload. It then checks whether the goods assigned to a specific route can actually be loaded. The loading module should report more than a simple yes or no. It should return cargo coordinates and orientations, and it should explain why loading is not feasible when the plan fails.

Third, a feedback loop between route planning and loading is established. Route planning and loading optimization are strongly coupled: the route determines the cargo set loaded in the same vehicle, and loading feasibility in turn affects route grouping. This paper converts loading infeasibility into feedback signals so that the route module avoids infeasible combinations in the next optimization round.

Fourth, the paper analyzes applications in Singapore land logistics. By introducing real-time traffic, urban delivery, port transfer, airport cargo transportation, cold-chain logistics, and large-item delivery, it explains the decision-making value of the agent in actual operations.

From a system perspective, the core logic of the agent is summarized in Figure 1.

*Figure 1: Closed-loop workflow of the proposed logistics optimization agent*




This process distinguishes the agent from an ordinary optimization algorithm. A normal algorithm usually solves only one fixed mathematical model, while the agent acts as a coordinator. It judges which algorithm is suitable for the current problem, decides when to call the routing module and when to call the loading module, explains constraint conflicts, and reorganizes the solution process according to failure feedback.

To express this collaborative process mathematically, let the route planning objective be $Z_{route}$ and the loading infeasibility penalty be $F_{load}$. The final objective of the agent can be written as:

$$\min Z = Z_{\text{route}} + F_{\text{load}} + F_{\text{time}} + F_{\text{carbon}}, \tag{4}$$

where $F_{time}$ denotes the time-window violation penalty, and $F_{carbon}$ denotes the carbon-emission or energy-consumption cost. If all loading plans are feasible, then $F_{load} = 0$. If some routes are not loadable, this term increases and pushes the route planning module to search again.

This modeling method is highly extensible. If driver working hours, vehicle restrictions, cold-chain temperature control, loading-equipment constraints, or carbon-emission quotas are introduced in the future, corresponding terms can be added to the objective function or constraint set, and the agent can still solve the problem in a modular way.

## 4. Agent System Architecture

The logistics optimization agent proposed in this paper consists of five main parts: the data perception layer, task modeling layer, algorithm decision layer, optimization execution layer, and explanation-feedback layer. Each layer corresponds to a key link in an actual logistics system.

The data perception layer receives external data, including customer coordinates, order demand, time windows, service duration, vehicle type, compartment dimensions, maximum payload, cargo length, width, height, weight, rotatability, stackability, real-time traffic conditions, and road restrictions. For the Singapore scenario, it can also include port queue status, airport cargo-station operating windows, commercial-area access restrictions, and electronic road-pricing information.

The task modeling layer converts raw data into an optimization problem. Customer nodes are abstracted as points on a graph, road traffic conditions are abstracted as edge weights, and vehicles and cargo are abstracted as constraint parameters. If road cost changes over time, the edge weight can be written as $c_{ij}(t)$. If vehicle types differ, each vehicle has different $Q_k$, $L_k$, $W_k$, $H_k$, and $G_k$ values.

The algorithm decision layer selects the solving strategy. If the number of customers is small and the variable scale is manageable, the agent selects QUBO modeling and simulated annealing. If the number of customers is large, the time windows are complex, or real-time response is required, it gives priority to Solomon I1 insertion, local search, ALNS, or GA. If a route is repeatedly rejected because of loading infeasibility, the algorithm decision layer increases the loading penalty weight or requires the route module to form special groups for large items, heavy goods, or cold-chain cargo.

The optimization execution layer includes two core modules: the route optimization module and the 3D loading module. The route optimization module outputs vehicle routes:

$$R = \{R_1, R_2, \dots, R_m\}. \tag{5}$$

The loading optimization module checks the cargo set $P_k$ corresponding to each route $R_k$ and outputs:

$$\Phi_k = (\phi_k, C_k, B_k), \tag{6}$$

where $\varphi_k \in \{0,1\}$ denotes whether loading is feasible, $C_k$ denotes the set of cargo coordinates and orientations, and $B_k$ denotes infeasibility causes or risk descriptions.

The explanation-feedback layer converts optimization results into a plan that dispatchers can understand. For example, the system should not only state that "vehicle 3 visits customers 12, 29, 24, and 26"; it should also explain why this route is selected, what the estimated travel time is, whether there is lateness risk, what the vehicle load factor is, which goods are placed at the bottom of the compartment, which goods cannot be stacked, and what backup option should be used in case of congestion.

The responsibilities of the agent modules are summarized in Table 1.

*Table 1: Responsibilities of the logistics optimization agent modules*

| Module | Input | Output | Function |
|---|---|---|---|
| Data perception | Orders, vehicles, cargo, traffic | Standardized data | Build optimization instances |
| Route planning | Customers, edge weights, time windows, capacity | Candidate routes | Determine visiting sequence |
| Loading optimization | Cargo, compartment, route grouping | Coordinates, orientations, feasibility | Verify loading executability |
| Feedback coordination | Route results, loading results | Adjustment strategy | Handle constraint conflicts |
| Explanation output | Final plan | Reports and visualization | Support human review |

The advantage of this architecture is that modules are both independent and mutually connected through feedback. The route module does not need to handle all three-dimensional spatial details, and the loading module does not need to search all route combinations independently. The agent connects the two complex problems through feedback coordination, thereby reducing overall solving difficulty.

## 5. Vehicle Route Planning Model

The basic task of the route planning module is to determine customer visiting sequences and vehicle assignment. Let the depot be node 0, the customer set be N = {1, 2, ..., n}, the full node set be V = N ∪ {0}, and the vehicle set be K. Define the binary variable:

$$x_{ijk} = \begin{cases} 1, & vehicle\ k\ travels\ from\ node\ i\ to\ node\ j, \\ 0, & otherwise. \end{cases} \tag{7}$$

Also define the vehicle usage variable:

$$y_k = \begin{cases} 1, & vehicle\ k\ is\ used, \\ 0, & vehicle\ k\ is\ not\ used. \end{cases} \tag{8}$$

If the objective is to minimize total travel time, the route cost is:

$$Z_{\text{travel}} = \sum_{k\in K}\sum_{i\in V}\sum_{\substack{j\in V \\ j\neq i}} c_{ij}\, x_{ijk}. \tag{9}$$

Here, $c_{ij}$ is the travel time from node i to node j. In a dynamic traffic scenario, it can be written as $c_{ij}(t)$, indicating the estimated travel time when departure occurs at time t.

The customer-visit uniqueness constraint is:

$$\sum_{k\in K}\sum_{\substack{j\in V \\ j\neq i}} x_{ijk} = 1, \quad \forall i \in N. \tag{10}$$

Vehicles depart from and return to the depot:

$$\sum_{j\in N} x_{0jk} = y_k, \quad \sum_{i\in N} x_{i0k} = y_k, \quad \forall k \in K. \tag{11}$$

For every customer node, the number of vehicle arrivals must equal the number of vehicle departures:

$$\sum_{\substack{i\in V \\ i\neq h}} x_{ihk} = \sum_{\substack{j\in V \\ j\neq h}} x_{hjk}, \quad \forall h \in N,\ \forall k \in K. \tag{12}$$

The capacity constraint is:

$$\sum_{i\in N} q_i \sum_{\substack{j\in V \\ j\neq i}} x_{ijk} \le Q_k, \quad \forall k \in K. \tag{13}$$

The time-continuity constraint ensures that the route order is consistent with service time. If vehicle k travels from i to j, then the service start time at customer j cannot be earlier than the time when the vehicle completes service at customer i and travels to j:

$$s_{jk} \ge s_{ik} + \tau_i + c_{ij} - M_{ij}\left(1 - x_{ijk}\right), \quad \forall i,j \in V,\ i \neq j,\ \forall k \in K. \tag{14}$$

Here, $M_{ij}$ is a sufficiently large constant. To improve model quality, a tighter pair-dependent Big-M can be used:

$$M_{ij} = \max\{0, b_i + \tau_i + c_{ij} - a_j\}. \tag{15}$$

The hard time-window constraint is:

$$a_i \le s_{ik} \le b_i, \quad \forall i \in N,\ \forall k \in K. \tag{16}$$

In urban delivery, hard time windows apply to strict scenarios such as port cutoff times, airport cargo handovers, and shopping-mall receiving windows. Soft time windows apply to ordinary e-commerce delivery, retail replenishment, or orders whose delivery time can be negotiated.

## 6. Soft Time Windows and Multi-Objective Optimization

In actual logistics, time windows are not always absolutely inviolable. Some customers allow early or late service, but waiting costs, breach-of-contract costs, or customer satisfaction losses may arise. Therefore, this paper introduces a soft time-window model. Let the service start time of customer i be $s_i$, and let its time window be $[a_i, b_i]$. If the vehicle arrives early, the early-arrival amount is:

$$v_i^- = \max(0, a_i - s_i). \tag{17}$$

If the vehicle arrives late, the late-arrival amount is:

$$v_i^+ = \max(0, s_i - b_i). \tag{18}$$

The soft time-window penalty can be written as:

$$F_{\text{time}} = \alpha \sum_{i \in N} v_i^+ + \beta \sum_{i \in N} v_i^-, \tag{19}$$

where $\alpha$ is the late-arrival penalty coefficient, and $\beta$ is the early-arrival waiting-cost coefficient. If the enterprise attaches more importance to punctual delivery, a larger $\alpha$ can be used. If waiting is allowed but lateness is not, a smaller $\beta$ and larger $\alpha$ can be used.

Combining vehicle usage, travel time, and time-window penalties, a multi-objective function can be constructed:

$$\min Z = \lambda_1 \sum_{k \in K} y_k + \lambda_2 \sum_{k \in K} \sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} c_{ij}\, x_{ijk} + \lambda_3 \sum_{i \in N} v_i^+ + \lambda_4 \sum_{i \in N} v_i^-. \tag{20}$$

Here, $\lambda_1$, $\lambda_2$, $\lambda_3$, and $\lambda_4$ reflect managerial preferences regarding vehicle count, transportation time, lateness, and waiting. If the goal is to first minimize the number of vehicles and then minimize travel time under the fixed vehicle count, a two-stage optimization can be used.

Stage 1:

$$\min Z_1 = \sum_{k \in K} y_k. \tag{21}$$

Stage 2, under the condition $\sum_{k \in K} y_k = m^*$:

$$\min Z_2 = \sum_{k \in K} \sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} c_{ij}\, x_{ijk}. \tag{22}$$

Here, $m^*$ is the minimum number of vehicles obtained in the first stage.

The soft time-window model matters for managers because it puts numbers on the trade-off between cost and service level. Very tight customer time windows often force the operation to add more vehicles just to stay on schedule. Some moderate lateness, on the other hand, can cut the vehicle count and bring total travel cost down a lot. Penalty coefficients can be adjusted across customer classes. High-value customers, port cutoff tasks, and cold-chain orders should carry high penalties, Regular replenishment work and more flexible orders get smaller penalties. So the system does more than generate a route. It spells out why some orders are handled on time, while others are pushed to a cheaper, more flexible schedule.

## 7. QUBO Modeling Method

QUBO stands for Quadratic Unconstrained Binary Optimization. Its standard form is:

$$\min_{x \in \{0,1\}^m} H(x) = x^\top Q x, \tag{23}$$

where x is a vector of binary variables and Q is a coefficient matrix. The key feature of QUBO is that all variables can only take values 0 or 1, and all constraints are transformed into penalty terms in the objective function. The final model is therefore an unconstrained binary optimization problem.

For vehicle route planning, the most direct variable definition is $x_{ij}$, indicating whether the vehicle travels from node i to node j. For multi-vehicle problems, it can be extended to $x_{ijk}$. If the visiting order is considered, a position variable can also be used:

$$z_{ip} = \begin{cases} 1, & customer\ i\ is\ assigned\ to\ position\ p\ in\ the\ visiting\ sequence, \\ 0, & otherwise. \end{cases} \tag{24}$$

With position variables, each customer appears in exactly one position:

$$\sum_{p=1}^{n} z_{ip} = 1, \quad \forall i \in N. \tag{25}$$

Each position is assigned exactly one customer:

$$\sum_{i \in N} z_{ip} = 1, \quad \forall p = 1,2,\dots,n. \tag{26}$$

These two constraints can be transformed into penalty terms:

$$H_1 = A \sum_{i \in N} \left( \sum_{p=1}^{n} z_{ip} - 1 \right)^2, \tag{27}$$

$$H_2 = B \sum_{p=1}^{n} \left( \sum_{i \in N} z_{ip} - 1 \right)^2. \tag{28}$$

The route-cost term can be expressed through customer combinations at adjacent positions:

$$H_{\text{cost}} = \sum_{p=1}^{n-1} \sum_{i \in N} \sum_{j \in N} c_{ij}\, z_{ip} z_{j,p+1} + \sum_{i \in N} c_{0i}\, z_{i1} + \sum_{i \in N} c_{i0}\, z_{in}. \tag{29}$$

Capacity and time-window constraints can be handled similarly. Taking capacity as an example, if the customer set of a vehicle is represented by binary variables, slack variables can be introduced to transform an inequality into an equality, which is then squared as a penalty:

$$H_{\text{cap}} = C \left( \sum_{i \in N} q_i\, u_i + \sum_{\ell=0}^{L} 2^{\ell}\, r_{\ell} - Q \right)^2. \tag{30}$$

Here, $u_i$ indicates whether customer i is assigned to the vehicle, and $r_\ell$ is a binary slack variable.

Picking the penalty coefficients is what really determines whether a QUBO model behaves well. Penalties that are set too low let the solver break constraints just to cut route cost. But penalties that are set too high take over the objective function. Then the search process struggles to tell which feasible solutions are actually better. In practice, A, B, C, and the related parameters are usually tied to the size of the transportation costs. Any constraint violation should trigger a penalty that is larger than the biggest route-cost improvement the model could get.

Within the agentic framework, QUBO works well for routing tasks that stay in the small-to-medium range. It fits cases where the system has to rapidly optimize about 10 to 20 customer nodes in one area. It can handle quick local changes too, like reordering a vehicle's nearby stops. But once the problem becomes global and involves more than 50 customers, a straight QUBO setup can balloon into too many variables. Breaking the problem up tends to work better. Customers can be grouped using clustering or heuristics, and then QUBO can be used to optimize each subgroup.

## 8. Simulated Annealing Algorithm

The simulated annealing algorithm is inspired by the physical annealing process. At high temperature, particles in a metal move actively. When the temperature is reduced slowly, the material can gradually form a

low-energy and stable structure. Simulated annealing in combinatorial optimization follows this idea: at the beginning of the search, it allows broad random exploration and even accepts some worse solutions; as the temperature decreases, randomness is gradually reduced and the algorithm tends to converge [6].

Let the current solution be x, the candidate solution be x′, and the objective function be H(x). If the candidate solution is better, that is, H(x′) < H(x), it is accepted directly. If it is worse, it is accepted with probability:

$$P(x \to x') = \exp\left(-\frac{H(x') - H(x)}{T}\right). \tag{31}$$

When the temperature T is high, even a worse candidate solution may be accepted, which helps the algorithm escape local optima. When the temperature is low, the probability of accepting worse solutions decreases, and the algorithm gradually stabilizes in a better region.

In a QUBO routing problem, a candidate solution can be generated by flipping a binary variable or by exchanging the visiting positions of two customers. If position variables $z_{ip}$ are used, common neighborhood operations include:

1. Swapping the positions of two customers in the sequence.
2. Moving a customer from its current position to another position.
3. Reversing a segment of the customer sequence.
4. Randomly reordering a local customer subset.

The basic simulated annealing process is:

Input: initial solution x, initial temperature T0, cooling rate eta, terminal temperature Tmin

Output: best solution xbest found so far

1. Initialize xbest = x

2. While T > Tmin:

2.1 Generate candidate solution x' near the current solution

2.2 Compute Delta = H(x') - H(x)

2.3 If Delta < 0, accept x'

2.4 Otherwise, accept x' with probability exp(-Delta / T)

2.5 If x is better than xbest, update xbest

2.6 T = eta * T

3. Return xbest

The advantage of simulated annealing is that it is simple to implement, broadly applicable, and naturally compatible with QUBO. For small- and medium-scale routing problems, it can quickly generate good solutions. For large-scale problems, it can serve as an acceptance criterion inside ALNS or GA. For example, in ALNS, after destroy and repair operators generate a new solution, if the new solution is worse than the current solution, it can still be accepted according to the simulated annealing probability, preventing premature search stagnation.

In Singapore's dynamic traffic environment, simulated annealing can also support rolling optimization. When real-time traffic conditions change, the system does not need to solve all routes from scratch. Instead, it can use the current route as an initial solution and conduct local perturbation and fast annealing around affected areas. This improves response speed while reducing frequent disruptions to drivers' execution plans.

## 9. Heuristic and Metaheuristic Routing Algorithms

When the number of customers increases to 50 or more, the number of variables in a complete QUBO model grows rapidly and the solving difficulty rises. Therefore, the agent needs to combine heuristics and metaheuristics for large-scale instances. The materials referenced in this paper include Solomon I1 insertion heuristics, 2-opt, Relocate, ALNS, and GA. These methods can play different roles in different stages.

Solomon I1 insertion is a classical constructive heuristic for VRPTW [4,5]. It usually selects a seed customer to start a route and then inserts unassigned customers into feasible positions in the current route. The insertion cost can generally be written as:

$$\Delta c(i,p) = c_{p,i} + c_{i,next(p)} - c_{p,next(p)}, \tag{32}$$

where p denotes the node before the insertion position, and next(p) denotes the node after the insertion position. If the route remains feasible after insertion with respect to capacity and time windows, the insertion is feasible. The algorithm continuously selects low-cost or well-evaluated insertions until no more customers can be inserted into the current route, after which a new route is opened.

2-opt is an intra-route improvement algorithm. It selects two edges in a route and attempts to reverse the path segment between them to reduce crossings and detours. If the original route segment is:

$$\cdots \rightarrow i \rightarrow i+1 \rightarrow \cdots \rightarrow j \rightarrow j+1 \rightarrow \cdots, \tag{33}$$

2-opt attempts to replace it with:

$$\cdots \rightarrow i \rightarrow j \rightarrow \cdots \rightarrow i+1 \rightarrow j+1 \rightarrow \cdots. \tag{34}$$

If the replacement remains feasible and reduces total cost, the operation is accepted.

The Relocate operator is used for cross-route adjustment. It attempts to move a customer from one route to a position in another route to reduce total cost or balance vehicle load. This operator is especially useful for situations in which one vehicle is overloaded, another is underused, or a route suffers from severe time-window conflicts.

ALNS stands for adaptive large neighborhood search. Its core idea is to repeatedly destroy part of the current solution and rebuild it using repair operators. Destroy operators may randomly remove several customers or remove customers with the largest contribution to the objective function. Repair operators may use greedy insertion or Regret-2 insertion. Regret-2 insertion prioritizes customers that will become much more costly if they are not inserted immediately. The regret value can be written as [7,11]:

$$regret_2(i) = c_2(i) - c_1(i), \tag{35}$$

where $c_1$(i) and $c_2$(i) are the best and second-best insertion costs for customer i.

Genetic algorithms are suitable for soft time windows and multi-objective trade-offs [10]. A chromosome can be represented as a customer permutation. During decoding, the sequence is divided into routes according to capacity and time-window rules. The fitness function can be set as:

$$fitness(s) = \frac{1}{Z(s) + \epsilon}, \tag{36}$$

where Z(s) is the total cost of solution s, and ε is a small constant used to avoid division by zero. Through selection, crossover, mutation, and embedded local search, GA can explore diverse solutions in a large search space.

In the agent, these algorithms are not mutually exclusive. Solomon I1 can quickly generate an initial solution, 2-opt and Relocate can perform local improvement, ALNS can restructure routes globally, GA can handle soft constraints and multi-objective trade-offs, and QUBO with simulated annealing can solve small- and medium-scale local subproblems.

## 10. Three-Dimensional Loading Problem Modeling

Three-dimensional loading optimization determines whether cargo can be placed inside a vehicle compartment while satisfying all constraints. Let the internal space of vehicle k be a rectangular box with dimensions ($L_k$, $W_k$, $H_k$) and maximum payload $G_k$. Let the cargo set assigned to vehicle k be $P_k$. For each cargo item p ∈ $P_k$, its dimensions are ($l_p$, $w_p$, $h_p$), its weight is $g_p$, and its feasible orientation set is $O_p$.

If cargo item p chooses orientation $o_p$, the rotated dimensions are:

$$\left(l_p^{o_p}, w_p^{o_p}, h_p^{o_p}\right). \tag{37}$$

The left-rear-bottom coordinate of the cargo inside the compartment is:

$$\left(u_p, v_p, z_p\right). \tag{38}$$

The boundary constraints require that the cargo must not exceed the compartment:

$$0 \le u_p, \quad u_p + l_p^{o_p} \le L_k, \tag{39}$$

$$0 \le v_p, \quad v_p + w_p^{o_p} \le W_k, \tag{40}$$

$$0 \le z_p, \quad z_p + h_p^{o_p} \le H_k. \tag{41}$$

The total weight constraint is:

$$\sum_{p \in P_k} g_p \le G_k. \tag{42}$$

Any two cargo items must not overlap in space. For items p and q, at least one of the following six separation conditions must hold:

$$\begin{aligned} u_p + l_p^{o_p} \le u_q \ \vee\ u_q + l_q^{o_q} \le u_p \ \vee\ v_p + w_p^{o_p} \le v_q \ \vee\ v_q + w_q^{o_q} \le v_p \ \vee\ z_p + h_p^{o_p} \\ \le z_q \ \vee\ z_q + h_q^{o_q} \le z_p. \end{aligned} \tag{43}$$

In real transportation, more constraints are also needed. First, there are rotation constraints. Some goods cannot be turned upside down or laid on their side, such as liquids, precision instruments, glass products, and cold-chain boxes. If item p cannot be rotated, then $O_p$ contains only its original orientation. Second, there are stacking constraints. If the maximum weight that item p can bear above it is $G_p^{stack}$, then the total weight of goods placed above it must not exceed this value. Third, there are center-of-gravity constraints. Heavy goods should be placed at the bottom and near the center of the compartment to reduce rollover risk during transportation. Fourth, there are unloading-sequence constraints. If a vehicle unloads goods according to the route sequence, goods for later customers should not block goods for earlier customers; otherwise, the driver must repeatedly move cargo.

Therefore, loading optimization is not only a geometric problem but also an execution problem. A loading plan that considers only space utilization but ignores unloading sequence may cause large amounts of extra handling during actual delivery. During loading verification, the agent should consider route sequence and place earlier-unloaded goods closer to the door or in easier-to-access positions.

## 11. Extreme Point Loading Algorithm

The extreme point algorithm is a commonly used engineering heuristic for three-dimensional bin packing. Its core idea is to maintain a set of candidate points where the next item may be placed. Initially, the compartment is empty, and the only candidate point is the left-rear-bottom corner: [12]

$$E = \{(0,0,0)\}. \tag{44}$$

When item p is placed at ($u_p$, $v_p$, $z_p$) with dimensions ($l_p$, $w_p$, $h_p$), three new candidate points are generated:

$$e_1 = \left(u_p + l_p, v_p, z_p\right), \tag{45}$$

$$e_2 = (u_p, v_p + w_p, z_p), \tag{46}$$

$$e_3 = (u_p, v_p, z_p + h_p). \tag{47}$$

These three points are located to the right, in front, and above the placed item. The algorithm then removes candidate points that are out of bounds, occupied, or dominated. A dominated point is a point that is no better than another candidate point in all three coordinate dimensions and therefore usually does not need to be retained.

The extreme point algorithm proceeds as follows:

Input: vehicle dimensions, cargo set, rotation rules, stacking rules

Output: loading coordinates and orientations, or infeasibility causes

1. Initialize candidate point set E={(0,0,0)}

2. Sort cargo by volume, weight, destination order, or priority

3. For each cargo item p:

3.1 Enumerate feasible orientations o

3.2 Enumerate candidate points e

3.3 Check boundary, overlap, weight, stacking, and unloading sequence

3.4 Select the best feasible position

3.5 Update the candidate point set

4. If all items are successfully placed, return the loading plan

5. If some item cannot be placed, return the reason for failure

The candidate-position evaluation function can combine multiple indicators:

$$score(p, e, o) = \theta_1 U_{\text{space}} + \theta_2 U_{\text{stability}} + \theta_3 U_{\text{unload}} + \theta_4 U_{\text{balance}}, \tag{48}$$

where $U_{space}$ denotes space utilization, $U_{stability}$ denotes stability, $U_{unload}$ denotes unloading convenience, and $U_{balance}$ denotes center-of-gravity balance. The weights θ1, θ2, θ3, θ4 can be adjusted according to different scenarios.

For large-item delivery, the weights of stability and unloading convenience should be increased. For e-commerce parcels, space utilization should receive more weight. For cold-chain transportation, goods in the same temperature zone should be placed together to avoid excessive temperature fluctuation caused by frequent door opening.

The extreme point algorithm is fast, interpretable, and easy to connect with the route planning module. If a cargo item cannot be loaded, the algorithm can clearly indicate the reason: dimensions exceed the compartment, weight exceeds the limit, candidate space is insufficient, rotation is not allowed, stacking is not allowed, or the unloading sequence is blocked. These reasons can be converted by the agent into feedback for the next round of route planning.

## 12. Coupling Between Routing and Loading

The coupling between route planning and three-dimensional loading is the most important research point in this paper. Traditional methods often solve the route first and then check loading. If loading fails, humans must manually adjust the route. This approach is inefficient, and the cause of failure is hard to use systematically. This paper argues that loading feasibility should be continuously considered during route planning.

Let the route planning module generate the route set:

$$R = \{R_1, R_2, \dots, R_m\}. \tag{49}$$

For route $R_k$, its customer set corresponds to cargo set $P_k$. The loading module provides the feasibility function:

$$\phi_k(R_k) = \begin{cases} 1 & \text{the cargo on route } R_k \text{ can be loaded,} \\ 0, & \text{the cargo on route } R_k \text{ cannot be loaded.} \end{cases} \tag{50}$$

If there exists $\varphi_k(R_k) = 0$, the whole plan is not executable. To allow the route planning module to perceive loading failure, a loading penalty can be defined:

$$F_{\text{load}}(R) = \mu \sum_{k \in K} \left(1 - \phi_k(R_k)\right) + \nu \sum_{k \in K} \Delta_k, \tag{51}$$

where $\Delta_k$ denotes the degree of loading failure. It can consist of several components:

$$\Delta_k = \rho_1 \Delta_k^{weight} + \rho_2 \Delta_k^{volume} + \rho_3 \Delta_k^{shape} + \rho_4 \Delta_k^{stack} + \rho_5 \Delta_k^{unload}. \tag{52}$$

Here, $\Delta_k^{weight}$ denotes weight excess, $\Delta_k^{volume}$ denotes volume or space excess, $\Delta_k^{shape}$ denotes conflict between cargo shape and compartment dimensions, $\Delta_k^{stack}$ denotes stacking conflict, and $\Delta_k^{unload}$ denotes unloading-sequence conflict.

The updated route objective is:

$$Z'_{\text{route}} = Z_{\text{route}} + F_{\text{load}}(R). \tag{53}$$

Loading failure is not just a final sign that a solution does not work anymore. In route search, it can be treated as feedback the agent learns from. For instance, routes that keep failing because too many large items get packed onto them push the agent to shift some customers to other vehicles. A customer with long cargo that cannot be rotated makes the agent favor a vehicle with a longer compartment. And if goods for an earlier customer end up blocked behind goods meant for a later customer, the system can adjust the visiting sequence or rearrange loading orientations.

This kind of feedback loop is useful in day-to-day operations. Fewer routes need to be changed by hand. It also raises the odds that the plan works on the first run. In a high-timeliness logistics setting like Singapore, changes made after a delivery plan is already in motion can get expensive fast. So checking loading feasibility early matters.

## 13. Agent Feedback Loop Algorithm

To achieve closed-loop coordination between routing and loading, this paper designs the following agent algorithm:

Input:

Customer set N

Vehicle set K

Cargo set P

Traffic cost matrix C(t)

Time windows [ai, bi]

Maximum number of iterations Imax

Output:

Executable routes R

Loading coordinates Cload

Explanation report Report

1. Initialize penalty weights and algorithm parameters
2. Generate initial groups according to customer location, demand, and time windows
3. Call the route planning module to generate candidate routes R
4. For each route Rk:

   4.1 Extract the corresponding cargo set Pk

   4.2 Call the 3D loading module

   4.3 Obtain feasibility φk and failure reason Bk
5. If all φk = 1:

   Output routes, loading plans, and explanation report
6. Otherwise:

   6.1 Construct Fload according to failure reasons

   6.2 Adjust customer grouping, vehicle selection, or route penalties

   6.3 Return to Step 3
7. If the maximum number of iterations is reached and no feasible plan is found:

   Output the best approximate plan and manual intervention suggestions

In this algorithm, the agent needs three types of judgment.

One part of the system is figuring out how big the problem is. With a small customer set, it can call QUBO and simulated annealing right away. A larger customer set goes through regional clustering before local optimization runs. The clustering step can use geographic location and time-window similarity. It can look at cargo dimensions too, along with vehicle compatibility.

For example, customer similarity can be defined as:

$$sim(i,j) = \eta_1 sim_{\text{geo}}(i,j) + \eta_2 sim_{\text{time}}(i,j) + \eta_3 sim_{\text{cargo}}(i,j). \tag{54}$$

Another requirement is figuring out why loading failed. A loading failure should not just come back as “infeasible.” It needs a specific label. Overweight issues can be fixed by switching to a larger vehicle or splitting the orders. Too much length calls for a longer vehicle. Stacking conflicts mean the cargo mix needs to change. Unloading-sequence conflicts might be handled by adjusting the placement orientation or changing the visiting order.

The agent needs a clear stopping point. It cannot run forever. A run can end once all constraints are met. It can stop too if objective improvement drops below a set threshold, if it hits the maximum number of iterations, or if it reaches the time limit specified by the user. In real-time scheduling, the system often has only a short window to produce a plan that can actually be executed. A “good enough and executable” result matters more than a theoretically optimal one.

The agent’s explanation report should spell out the route for every vehicle. It should show estimated departure and arrival times, along with total travel time. Time-window risk needs to be noted too. The report should cover the vehicle load factor and space utilization. Cargo placement coordinates should be included, plus notes for non-stackable items. Backup routes should be listed. It should end with suggestions for handling exceptions. Clear explanations like this can build dispatchers’ trust in the system. They can support enterprise review and continuous improvement as well.

## 14. Singapore Land Logistics Application Scenarios

Singapore is a typical high-density urban logistics environment. Its logistics system has several distinct characteristics: road resources are limited, traffic conditions change quickly, port and airport freight activities are concentrated, commercial and residential delivery demand is frequent, and vehicle scheduling must balance efficiency, punctuality, and urban management rules. The agent proposed in this paper can be applied in the following scenarios.

First, urban instant delivery. E-commerce platforms, supermarkets, and catering supply chains must complete large numbers of orders within a short time. Customer demand is dispersed and time windows are short, so routes need to be dynamically adjusted according to real-time traffic. The agent can continuously update $c_{ij}(t)$ and perform rolling optimization on local routes.

Second, retail store replenishment. Convenience stores, supermarkets, and shopping centers usually have fixed receiving periods, and vehicles must complete delivery within those windows. Cargo categories are diverse and sizes differ significantly, so loading sequence is closely related to unloading sequence. The agent can arrange goods near the vehicle door according to store visiting order, reducing repeated handling by drivers.

Third, port and airport cargo transfer. Port and airport tasks are strongly time-sensitive, such as cutoff times, flight loading times, and pickup windows. These tasks have high late-arrival penalties, so a larger $\alpha$ should be used in the model. At the same time, roads around container areas may be congested, requiring real-time updates to traffic costs.

Fourth, cold-chain transportation. Cold-chain goods require specific temperature zones and cannot be mixed freely with ordinary goods or exposed to frequent door opening. The agent should treat temperature-control capability as a vehicle selection constraint and keep goods in the same temperature zone together in the loading module.

Fifth, large-item delivery. Furniture, appliances, and equipment are often large and may not be rotatable, tiltable, or unloadable by one person. Route planning cannot consider weight only; it must also consider compartment length, width, and unloading sequence.

Sixth, green logistics and low-carbon delivery. Singapore has strong requirements for sustainable urban development, and logistics enterprises also need to reduce energy consumption and carbon emissions. The agent can add a carbon-emission term to the objective function:

$$F_{\text{carbon}} = \sum_{k\in K}\sum_{i\in V}\sum_{\substack{j\in V\\ j\neq i}} e_k\, c_{ij} x_{ijk}, \tag{55}$$

where $e_k$ denotes the unit-distance emission coefficient of vehicle k. If electric vehicles are used, battery constraints and charging-station access constraints can also be added.

In these scenarios, the value of the agent is not simply to compute the shortest path. Rather, it integrates traffic, vehicles, cargo, customer time windows, and operational goals into a unified decision process. For dispatchers, the system should output a complete operational plan rather than an abstract path.

## 15. Routing Case and Result Analysis

The vehicle routing case in the reference material contains 50 customer nodes and a total demand of 271 units. If vehicle capacity is Q = 60, the lower bound on the number of vehicles from capacity alone is:

$$\left\lceil \frac{271}{60} \right\rceil = 5. \tag{56}$$

However, the actual solution shows that 9 vehicles are required to obtain a feasible plan under hard time-window constraints. This indicates that time windows significantly limit the ability to merge routes. Some customers' service periods conflict with each other; even if vehicle capacity remains available, they cannot be placed on the same route.

The first method in the reference material uses Solomon I1 insertion to construct an initial solution and then applies 2-opt and Relocate for local improvement. The result uses 9 vehicles with a total travel time of 129.0. Route loads range from 18 to 37, all below the capacity limit of 60. Route travel times range from 10 to 18, and all vehicles can return to the depot within the required time.

The comparison is shown in Table 2.

*Table 2: Comparison of routing methods in the 50-customer case*

| Method | Total Travel Time | Vehicles | Description |
|---|---|---|---|
| Solomon I1 | 135.0 | 9 | Constructs an initial feasible solution |
| Solomon I1 + local search | 129.0 | 9 | Removes local detours and balances routes |
| ALNS | 92.0 | 9 | Large-neighborhood destroy and repair |

The improvement from 135.0 to 129.0 is:

$$\frac{135-129}{135}\times 100\% \approx 4.4\%. \tag{57}$$

After ALNS is applied, the total travel time decreases from 129.0 to 92.0. The improvement is:

$$\frac{129-92}{129}\times 100\% \approx 28.7\%. \tag{58}$$

The advantage of ALNS lies in its ability to remove and reinsert multiple customers simultaneously, thereby restructuring routes globally. Compared with simple local exchange, ALNS is more likely to escape local optima. For example, some customers may be assigned to routes outside nearby geographic regions in the original solution. Local search may make only small adjustments, while ALNS can directly destroy multiple routes and reorganize customer assignment.

However, none of the three methods reduces the number of vehicles below 9. This shows that the vehicle count is constrained by time-window conflicts rather than capacity alone. For the agent, this result is important: when the system attempts to reduce the number of vehicles, it should first identify customers with conflicting time windows, instead of simply increasing vehicle capacity or compressing route distance.

## 16. Soft Time Windows and Sensitivity Analysis

Under hard time-window constraints, all customers must be served within specified periods, which increases vehicle demand. If soft time windows are allowed, the system can trade cost against service level through lateness penalties.

The reference material scans the penalty coefficient and reports the sensitivity results shown in Table 3.

*Table 3: Soft time-window sensitivity results*

| alpha | Travel Time | Lateness Penalty | Total Cost | Vehicles | Violated Nodes |
|---|---|---|---|---|---|
| 0.5 | 127.0 | 150.0 | 277.0 | 5 | 22 |
| 1.0 | 123.0 | 255.0 | 378.0 | 5 | 22 |
| 2.0 | 127.0 | 600.0 | 727.0 | 5 | 24 |
| 5.0 | 124.0 | 1285.0 | 1409.0 | 5 | 24 |
| 10.0 | 141.0 | 2820.0 | 2961.0 | 5 | 23 |
| 20.0 | 132.0 | 5820.0 | 5952.0 | 5 | 28 |
| 50.0 | 132.0 | 14550.0 | 14682.0 | 5 | 28 |

When $\alpha = 0.5$, the model gives the lowest total cost, which is 277.0. In this case, travel time comes to 127.0 and the lateness penalty is 150.0. The plan uses 5 vehicles, and 22 nodes violate time windows. This option fits situations where delivery times are flexible and the enterprise cares more about vehicle utilization and keeping total cost down. With higher α values, the lateness cost increases fast. The system shifts toward punctuality, but the total cost goes up a lot too.

The vehicle-capacity sensitivity analysis shows a clear pattern as Q rises. Increasing Q from 15 to 35 cuts the number of vehicles from 19 to 10. Total travel time drops too, from 174.0 to 114.0. After that, the gains level off. For $Q \geq 40$, the solution stays at 9 vehicles, and total travel time stays at 129.0. Past that capacity

point, raising vehicle capacity does not do much for the route. The limiting factor shifts away from capacity and toward the time windows.

At Q = 35, the total travel time comes out to 114.0. That is lower than the 129.0 recorded at Q = 40, and it is not a contradiction. With a smaller capacity, the system ends up using more vehicles, so routes stay shorter and more compact. Total travel time drops. With a larger capacity, more customers can be grouped into one vehicle. But time-window conflicts can force extra detours. So Q=35 looks like an economic capacity turning point.

The time-window width sensitivity analysis finds that total travel time is relatively low at 114.0 with a time-window scaling factor of γ = 1.2. Loosening the time windows a bit gives the schedule more room to work. But stretching them too far can blow up the search space and make the algorithm less stable.

A Monte Carlo simulation was run by adding a ±30% random change to demand and a ±15% change to travel time. In total, 200 simulations were completed. Across these runs, the average total travel time comes out to 113.6. The standard deviation is 6.65, and the coefficient of variation is 5.85%. The 95% confidence interval falls between [102, 129]. Overall, the route planning solution stays fairly robust under this level of random disturbance.

## 17. Extended Significance of Loading Feedback in the Case

The above case mainly verifies the route planning module and does not directly include real cargo dimensions or 3D loading data. If it is placed into the agentic framework proposed in this paper, each customer node must be associated with a cargo set. For example, the cargo set of customer i can be expressed as:

$$P_i = \{p_{i1}, p_{i2}, \dots, p_{ir_i}\}. \tag{59}$$

The total cargo set of vehicle k is:

$$P_k = \bigcup_{i \in R_k} P_i. \tag{60}$$

The route planning result is truly executable only when every $P_k$ can be loaded into vehicle k. If a route is feasible from the routing perspective but the loading module returns $\varphi_k = 0$, the agent must analyze the cause of failure.

Assume route $R_5$ contains several large-item customers. The vehicle payload is still below $Q_k$, but the compartment length is insufficient, causing the longest item to be impossible to place. In this case, the feedback should not simply be "route failed"; it should be "the customer combination contains long-size cargo conflict and requires a longer vehicle or customer splitting." The corresponding penalty can be written as:

$$\Delta_k^{shape} = \max_{p \in P_k}\left(0, l_p^{min} - L_k\right), \tag{61}$$

where $l_p^{min}$ denotes the minimum length requirement of item p across all allowed orientations.

If the failure reason is a stacking conflict, such as fragile goods being pressed by heavy goods, the cargo combination or placement layer should be adjusted. The corresponding penalty can be written as:

$$\Delta_k^{stack} = \sum_{p \in P_k} \max\left(0, G_p^{above} - G_p^{stack}\right), \tag{62}$$

where $G_p^{above}$ denotes the total weight placed above item p.

If the failure reason is an unloading-sequence conflict, both the route module and the loading module may need adjustment. The route module can change the customer visiting sequence, and the loading module can move goods for earlier customers closer to the door. Unloading convenience can be represented by a sequence penalty:

$$\Delta_k^{unload} = \sum_{p,q \in P_k} I\left(order(p) < order(q)\right) I\left(block(q,p)\right), \tag{63}$$

where order(p) denotes the unloading order of the customer corresponding to item p, and block(q,p) denotes whether item q blocks item p.

Through this type of feedback, route planning no longer groups customers only according to coordinates and time windows. It gradually learns which customers are unsuitable for same-vehicle delivery, which orders require special vehicle types, and which goods should be placed earlier or later in the route. This is the core advantage of the agent compared with traditional optimization models.

## 18. System Output and Explainability Design

The output of a logistics scheduling system cannot be only a set of mathematical variables. Dispatchers and drivers need clear route, time, loading, and risk information. Therefore, the final output of the agent should include at least five types of information.

First, the vehicle-route output should be presented as a route table (Table 4), including depot, customer sequence, load, travel time, and time-window risk.

Second, the loading-plan output should be presented as a cargo table (Table 5), including vehicle assignment, coordinates, orientation, stacking status, and customer information.

*Table 4: Example vehicle-route output*

| Vehicle | Route | Load | Travel Time | Time-Window Risk |
|---|---|---|---|---|
| 1 | 0 -> 30 -> 20 -> 32 -> 10 -> 1 -> 0 | 35 | 10 | Low |
| 2 | 0 -> 40 -> 21 -> 22 -> 23 -> 39 -> 25 -> 4 -> 0 | 30 | 10 | Medium |

*Table 5: Example cargo-loading output*

| Cargo | Vehicle | Coordinate | Orientation | Stacking Status | Customer |
|---|---|---|---|---|---|
| p1 | 1 | (0, 0, 0) | Original | Bottom layer | 30 |
| p2 | 1 | (l1, 0, 0) | Rotated 90 degrees | Bottom layer | 20 |

Third, cost decomposition. The system needs to explain which components form the total objective:

$$Z = Z_{\text{travel}} + F_{\text{time}} + F_{\text{load}} + F_{\text{vehicle}} + F_{\text{carbon}}. \tag{64}$$

If one solution uses one more vehicle but significantly reduces lateness, the dispatcher should be able to see this trade-off.

Fourth, risk alerts. The system should mark traffic risk, lateness risk, loading risk, and abnormal vehicle utilization. For example, if a route passes through many congested roads, the agent should suggest “perform rolling optimization again 15 minutes before departure.” If a vehicle reaches 95% space utilization, it should warn that “loading tolerance is low; adding temporary orders is not recommended.”

Fifth, backup plans. In real execution, vehicle breakdowns, customer cancellations, and traffic accidents may occur. Therefore, the agent should keep several backup routes and loading adjustment plans. For example, if vehicle 3 cannot depart, the system should indicate which customers can be transferred to vehicle 5, which goods need to be reloaded, and how much the estimated cost will increase.

Explainability is very important for intelligent logistics systems. A black-box algorithm without explanation may be hard for dispatchers to trust even if its result is good. Through constraint explanations, failure-cause classification, cost decomposition, and backup plans, the agent makes optimization results easier for humans to understand and accept.

## 19. Deployment Process and Engineering Implementation Suggestions

A real logistics company could roll out the agent described in this paper in a few stages.

Work starts with data preparation. The enterprise pulls together customer addresses and past orders. It includes vehicle details, compartment dimensions, and cargo dimensions. Service time windows and driver shifts need to be in the same dataset. Road restrictions matter too, along with historical traffic data.

Data quality has a direct impact on how well the optimization works. Missing cargo dimensions force the 3D loading module to rely on rough estimates. Inaccurate customer addresses will throw off route planning, since the edge weights end up wrong.

The second stage focuses on basic route optimization. The enterprise can start by building a VRPTW model. Feasible routes can be generated using Solomon I1, local search, or ALNS, and then checked against the routes from manual dispatching. Performance can be judged by total travel time and the number of vehicles used. Punctuality rate matters too. Vehicle load factor and driver execution feedback should be included as well.

Stage three focuses on integrating the loading module. Cargo-dimension data needs to be set up for each order. Then the extreme point algorithm can be applied to produce the loading plans. Early on, the system can start with standard boxes, regular parcels, and large items. More complicated constraints can come later. Fragile goods, cold-chain goods, non-invertible goods, and other complex rules can be added step by step.

Stage four focuses on integrating a feedback loop. The reasons for loading failures are sent back to the route planning module. This lets the system adjust how it groups customers and which vehicles it picks on its own. Penalty weights need careful tuning here. Too much weight on loading feasibility can hurt routing efficiency. Too much weight on routing efficiency can trigger repeated loading failures.

Stage five focuses on real-time traffic and rolling optimization. Once the system brings in live road conditions, it can check routes again before the driver leaves, while deliveries are happening, and any time an exception comes up. Rolling optimization should limit how much the plan shifts so drivers are not constantly getting new tasks. A stability penalty can be set as:

$$F_{\text{change}} = \kappa \sum_{k \in K} d\left(R_k^{new}, R_k^{old}\right), \tag{65}$$

where $d(R_k^{new}, R_k^{old})$ denotes the difference between the new and old routes, and $\kappa$ controls the weight of route stability.

The sixth stage focuses on human-machine collaboration. The agent can generate a recommended plan, but dispatchers keep the final say. The system needs to let people lock in certain customer sequences. Dispatchers should be able to assign specific vehicles, block certain combinations, or require that particular customer needs are met. After those changes, the agent reoptimizes the rest of the plan around the human edits.

A modular setup makes sense for the engineering side of the system. The route optimization module can plug into the loading optimization module through APIs. The traffic prediction module can connect the same way, and the explanation report module can sit on top of those connections. Later changes stay contained. Simulated annealing could be swapped out for a quantum solver, or the extreme point algorithm could be replaced with a more complex mixed-integer model. The overall system structure would still stay the same.

## 20. Application Value and Managerial Implications

The agent proposed in this paper has multiple values for logistics enterprises.

Transportation costs can go down. Route optimization cuts detours and unnecessary repeat trips. Loading optimization makes better use of the vehicle's available space. The feedback loop limits rework and reduces second dispatches that happen after loading failures. In high-frequency urban delivery, small daily time savings per vehicle add up, and the cost advantage grows over time.

Delivery punctuality can improve too. The system uses time-window modeling and real-time traffic updates to spot lateness risk early. It can then suggest backup routes or a different departure time. For ports, airports, shopping malls, and cold-chain customers, being on time often matters more than the shortest distance.

Better vehicle utilization is another benefit. With traditional dispatching, capacity is usually guessed from total weight or how many orders are on the route. That approach can miss the three-dimensional space problem. The loading module in this paper plans at the level of cargo coordinates and orientations. This helps enterprises see more clearly whether a vehicle can take extra orders. It reduces situations where "weight remains available but space is insufficient" or "space remains available but weight is exceeded."

Another benefit is lower carbon emissions. Shorter travel time helps, and so do higher load factors and fewer ineffective deliveries. Together, these changes cut the energy used per order. The system can go further by taking in electric-vehicle battery levels. It can factor in charging-station locations and low-emission-zone rules. That supports greener logistics decisions.

Another benefit is clearer, more transparent management. The agent does more than give a final route. It provides explanations of the constraints and describes the risks. Managers can understand why a specific vehicle is chosen for a route. They can see why a customer ends up in a later batch. And they can check why one cargo item cannot be placed in the same vehicle as another. This kind of transparency helps enterprises build standardized scheduling processes.

It can be reused across different delivery and logistics settings. Urban delivery, e-commerce logistics, cold-chain transportation, port transfer, airport freight, store replenishment, and large-item delivery all look different on the surface. But they still come down to the same building blocks: routes, vehicles, cargo, time windows, and execution constraints. With a modular design, the agent can change the weights and constraints for each scenario without having to rebuild the full system.

From a manager's point of view, this study points to three practical takeaways. Logistics optimization works better when it is not driven by just one metric. Vehicle count, travel time, punctuality, loading rate, and carbon emissions all pull against each other at times. Algorithm outputs need to work in real operations. Loading feasibility and driver handling convenience should be built into the model from the start. And intelligent logistics systems should not remove people from the process entirely; Instead, they should help dispatchers make decisions. This matters most for tricky calculations and for flagging exceptions early.

## 21. Limitations and Future Research Directions

Although the agentic framework proposed in this paper is highly extensible, it still has several limitations.

First, the QUBO model has scale limitations. As the number of customers, vehicles, and time-encoding variables increases, the number of binary variables grows rapidly. For global routing problems with more than 50 customers, directly constructing a complete QUBO model is often unrealistic. Therefore, future work should study decomposed QUBO, in which regional partitioning, customer clustering, or vehicle assignment is performed first, followed by QUBO and simulated annealing for local subproblems.

Second, the three-dimensional loading model is still simplified. This paper mainly considers rectangular cargo, rotation, stacking, and non-overlap constraints. In reality, cargo may have irregular shapes, liquid sloshing, center-of-gravity offset, packaging-strength differences, and loading-equipment constraints. Future research can introduce mechanical stability models and vibration-risk evaluation during vehicle movement.

Third, traffic-state prediction still needs further improvement. This paper uses real-time traffic cost $c_{ij}(t)$ to represent dynamic road conditions, but actual deployment requires prediction of future traffic states. If only current congestion information is used, it may fail to accurately reflect traffic conditions when the vehicle reaches a road segment. Future models can combine historical traffic data, holiday factors, weather, and event information.

Fourth, the interaction mechanism between the agent and human dispatchers needs further refinement. In real enterprises, dispatchers may lock certain customer combinations or specify drivers based on experience. Future systems should support human constraint input and rapid reoptimization after manual modifications.

Fifth, the routing case in this paper mainly comes from existing materials, and the loading module has not yet been verified end-to-end with real cargo data. Future work should build an integrated dataset containing customers, orders, cargo dimensions, vehicle types, and real traffic, and compare "route-only optimization" with "route-plus-loading closed-loop optimization" in terms of execution success rate, vehicle utilization, and total cost.

Future research can proceed in three directions. First, in algorithm fusion, ALNS, GA, QUBO, and reinforcement learning can be combined so that the agent automatically selects algorithms based on instance characteristics. Second, in digital twins, a simulation environment for Singapore roads and logistics nodes can be established to test strategies virtually. Third, in green logistics, carbon emissions, vehicle battery levels, charging stations, and low-emission-zone rules can be incorporated so that the system supports both cost efficiency and sustainable development.

## 22. Conclusion

This paper proposes a logistics agent framework for collaborative optimization of route planning and three-dimensional loading in Singapore land logistics. The framework treats vehicle route planning and 3D loading optimization as two core problems and uses the agent as a coordinator to realize a closed-loop process involving data reading, algorithm selection, route generation, loading verification, failure feedback, and plan explanation.

For route planning, this paper establishes VRPTW/CVRPTW models with capacity and time-window constraints and discusses hard time windows, soft time windows, multi-objective functions, and two-stage optimization methods. For small- and medium-scale route problems, the model is converted into QUBO form. Customer-visit uniqueness, flow conservation, capacity, and time-window penalty terms are used to construct an unconstrained binary optimization objective, which can be solved by simulated annealing. For larger-scale problems, this paper introduces heuristic and metaheuristic methods such as Solomon I1, 2-opt, Relocate, ALNS, and GA.

For loading optimization, this paper models cargo placement as a three-dimensional bin packing problem and considers compartment dimensions, maximum payload, cargo size, rotation direction, stackability, non-overlap conditions, and unloading sequence. The extreme point algorithm maintains a set of candidate points, places goods one by one, generates new candidate positions, and can quickly judge loading feasibility while outputting specific coordinates.

The core contribution of this paper is the establishment of a feedback loop between route planning and loading optimization. If a route is feasible from the routing perspective but fails in loading, the agent converts the failure reason into a penalty signal and readjusts customer grouping, vehicle selection, or visiting sequence until an executable plan satisfying both routing and loading constraints is generated. This mechanism is more suitable for real logistics dispatching than the traditional “plan first, check later” approach.

Case analysis shows that in a 50-customer routing problem, Solomon I1 with local search obtains a feasible solution with 9 vehicles and a total travel time of 129.0. ALNS reduces the total travel time to 92.0 while keeping the same 9 vehicles, an improvement of about 28.7%. Soft time windows and sensitivity analysis further show that vehicle capacity, time-window width, customer number, and demand fluctuation all affect the final plan. If three-dimensional loading verification is further added, the system can more accurately judge whether a plan is executable.

Overall, the agent proposed in this paper can be applied to urban delivery, e-commerce logistics, cold-chain transportation, port and airport cargo transfer, retail store replenishment, and large-item delivery. For urban logistics environments such as Singapore, where traffic is dense and timeliness requirements are high, the agent can help reduce transportation cost, improve delivery punctuality, increase vehicle utilization, reduce carbon emissions, and promote logistics management from experience-driven dispatching toward real-time, data-driven, and explainable intelligent decision-making.

## References

[1] Group 4. (2026). Agentic AI for transportation [Course lecture material].

[2] MC2602690. (2026). Modeling and optimization of a vehicle routing problem with time-window constraints based on quantum computing [MathorCup Mathematical Application Challenge, Problem A material].

[3] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80-91. https://doi.org/10.1287/mnsc.6.1.80

[4] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 35(2), 254-265. https://doi.org/10.1287/opre.35.2.254

[5] Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation Science, 39(1), 104-118. https://doi.org/10.1287/trsc.1030.0056

[6] Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671-680. https://doi.org/10.1126/science.220.4598.671

[7] Pisinger, D., & Røpke, S. (2007). A general heuristic for vehicle routing problems. Computers & Operations Research, 34(8), 2403-2435. https://doi.org/10.1016/j.cor.2005.09.012

[8] Toth, P., & Vigo, D. (Eds.). (2002). The vehicle routing problem. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515

[9] Desaulniers, G., Desrosiers, J., & Solomon, M. M. (Eds.). (2005). Column generation. Springer. https://doi.org/10.1007/0-387-25486-2

[10] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley.

[11] Røpke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40(4), 455-472. https://doi.org/10.1287/trsc.1050.0135

[12] Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. INFORMS Journal on Computing, 20(3), 368-384. https://doi.org/10.1287/ijoc.1070.0250

## Funding

This research received no external funding.

## Conflicts of Interest

The authors declare no conflict of interest.

## Acknowledgment

This paper is an output of the science project.

## Copyrights