2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Dual-Process Optimization for Multi-Vehicle Route Planning and Parts Collection Sequencing

Ryota Higa[1,2]*, Takuro Kato[3]* and Florence Ho[1,2]

***Abstract*— We proposed a novel dual-process optimization approach for parts collection order and route planning in parts warehouses. Conventional multi-agent parts collection typically uses the vehicle routing problem (VRP), which focuses on minimizing the number of agents and costs. However, the model does not fully leverage the vehicle's potential. Moreover, multi-agent path finding (MAPF) focuses on route planning and avoiding path conflicts, ignoring the order of part collection. The proposed approach integrates algorithms from the traveling salesman problem (TSP) and path planning, and modifies them to suit the dynamic and complex environment of parts warehouses. This integration streamlines the collection process and considerably reduces the operational time. Thus, the study can improve automation and efficiency in parts warehouse management and improve optimization techniques. The proposed method achieved more than tenfold acceleration compared with the ideal centralized optimization, without cost increments. As the number of agents and part collections increases, centralized optimization requires a metaheuristic approach, which results in solution degradation. However, the proposed approach maintains over tenfold acceleration and produces solutions with shorter operational times. Furthermore, we conducted an ablation study comparing six methods, from entirely independent to centralized optimization, demonstrating that the proposed approach effectively balances computational time and solution accuracy.**

I. Introduction

Mobility automation and AI advancements have made warehouse optimization operations increasingly crucial. The push for automation in the logistics industry, led by industry leaders such as Amazon [1] and Toyota, has significantly improved efficiency and accuracy. For example, Amazon has implemented multi-agent path finding (MAPF) for efficient route planning while avoiding collisions between robots[2]. However, with MAPF being an NP-hard problem, this field is increasingly becoming complex [3]. For applications in the automotive industry, such as in Toyota's parts warehouses, efficient route planning is necessary to retrieve a variety of parts, which requires a complex NP-hard structure similar to that of the traveling salesman problem (TSP)[4], going beyond simple pickup and delivery problems (PDPs).

Generally, two approaches are adopted for route planning and parts collection order optimization: centralized and distributed optimization. Well-known route planning algorithms include A*, cooperative A*(CA* or STA*)[5], and conflict-based search (CBS)[6], [7]. To determine the order of parts retrieval, centralized optimization incorporates approaches such as the vehicle routing problem (VRP) and capacitated vehicle routing problem (CVRP), whereas distributed optimization typically relies on TSP. However, when addressing large-scale problems, combining centralized and distributed optimization is a feasible solution.

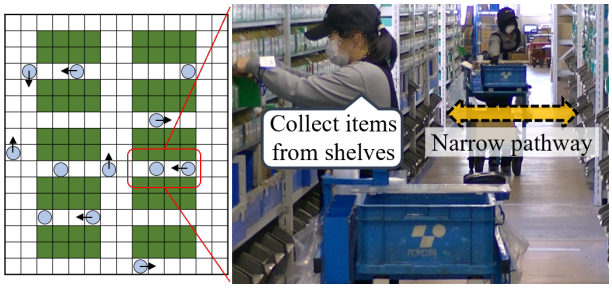

Fig. 1. Parts Warehouse Operations: Parts collection operation in parts warehouse. Each worker receives work instructions, collects all designated parts in the cart, and subsequently delivers them to the designated depot. Because this operation is performed in a shared space with narrow pathways, planning to avoid interference between workers is necessary. This is a dual optimization problem: the parts collection sequence and the paths between collection points.

Studies have investigated centralized optimization techniques, such as the extended time-dependent vehicle routing problem (ETDVRP) in the shared space; however, this method does not consider influential factors such as dynamic picking operation times [8]. Given that the ETDVRP was founded on the VRP, the contribution of the changes in the number of mobility units is significant to our experimental results. However, we have yet to thoroughly evaluate this aspect in our scenarios, which have a fixed number of units. Moreover, the ETDVRP, as a centralized optimization approach, encounters computational bottlenecks in warehouses handling many parts.

Therefore, in this study, we proposed a novel dual-process optimization method that maximizes the use of the given number of vehicles, quickly realizing efficient route planning for retrieving multiple types of parts without collisions. By independently planning the actions of each robot, this method achieves results close to those of centralized optimization. In this study, we enhanced the capability of the ETDVRP solver to handle dynamic obstacles and evaluated the balance between centralized and independent optimization through an ablation study. The proposed approach exhibits a well-balanced performance in real-time computation and has

*The two authors equally contributed to this work.

[1]Data Science Laboratories, NEC Corporation, 1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, Japan r-higaryouta@nec.com

[2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-4-7 Aomi, Koto-ku, Tokyo, Japan

[3]TICO-AIST Cooperative Research Laboratory for Advanced Logistics, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1, Umezono, Tsukuba, Japan takuro.katou@aist.go.jp

the potential to significantly improve operations in parts warehouses.

The contributions of this study can be summarized as follows:

- We proposed dual-process Optimization, an effective method for optimizing parts warehouses in which the overall cost optimization problem is divided into TSP and STA sub-problems, and revealed that this process is faster than previous methods and produces superior performance results.
- Compared with the previous centralized optimization method, the proposed method executes ten times faster and improves solution costs by 1.6%.
- We conducted six comparative experiments with methods ranging from entirely independent optimization to partially and fully optimal methods. The proposed approach maintained its speed and an appropriate balance between centralization and distribution.

II. Related Works

Warehouse optimization can be categorized into NP-hard combinatorial optimization problems such as MAPF and PDP. This field has been extensively studied and related studies include the following:

MAPF with Task Assignment: Studies on MAPF have typically focused on derivatives of algorithms such as A* and CBS. CBS can be accelerated by applying EECBS[9], a suboptimal algorithm. Other approaches, such as CBM[10], T-CBS[11], and ECBS-TA[12], have combined task assignment with CBS. For lifelong assignments, algorithms such as RHCR[13] and CCMP[14] are notable. Combining destination assignment with route planning alone is sufficiently challenging.

TSP, VRP, and PDP: The PDP is an NP-hard combinatorial optimization problem. To address the underlying time minimization problem in the ETDVRP, the TDVRP[15] can be used. Typical problems include the TSP and CVRP, which have evolved from phases of algorithm improvement to practical libraries. For example, Concorde[16] is the fastest library for obtaining exact TSP solutions. Regarding metaheuristics, LKH[17] is recognized for addressing various problem classes derived from the TSP and CVRP, including the time-dependent forms of PDP. Furthermore, methods more robust than RHCR have been proposed for multi-agent PDP, even in cases with agent speed and synchronization fluctuations [18].

III. Problem Definition

A. Time-Dependent Multi-Vehicle Parts Collection

We consider an undirected graph $G = (V, E)$ and a set of agents $A = \{1, \dots, N\}$. Each agent i has a start node $s_i \in V$ and a goal node $g_i \in V$ and is assigned M tasks denoted as $\Gamma_i = \{\gamma_i^1, \dots, \gamma_i^M\}$, where $\gamma_i^j \in V$. Here, we define sets of start nodes, goal nodes, and tasks as $\mathcal{S} := \{s_1, \dots, s_N\}, \mathcal{G} := \{g_1, \dots, g_N\}, \Gamma := \{\Gamma_1, \dots \Gamma_N\}$. To complete task set Γ_i, each agent i should visit all task locations $\gamma_i^j, j \in \{1, \dots, M\}$ and stay at each task locations for the service duration of τ time steps.

Let π_i represent the route in which agent i visits all its task locations Γ_i and arrives at the goal node g_i by a certain time T, including a service duration of τ time steps at each task location. Here, $\pi_i[t] \in V$ is the location of agent i at each time step t. We define the set of all agents' routes as $\Pi := \{\pi_1, \dots, \pi_n\}$. At each time step, agents can either move to an adjacent node or stay at their current node, that is, $(\pi_i[t], \pi_i[t+1]) \in E \vee \pi_i[t] = \pi_i[t+1]$. Agents should avoid following two types of conflicts with other agents at each time step $t \in \mathbb{Z}_{\geq 0}$.

- *Node Conflict:* $\pi_i[t] = \pi_j[t]$
- *Edge Conflict:* $\pi_i[t] = \pi_j[t+1] \wedge \pi_j[t] = \pi_i[t+1]$

We define the cost of each agent as T_i, where $T_i \in \mathbb{Z}_{\geq 0}$ is the earliest time such that $\pi_i[T_i] = \pi_i[T_i + 1] = \ldots = \pi_i[T] = g_i$.

The objective function is defined as follows:

$$\min \sum_{i=1}^{N} T_i. \tag{1}$$

Minimizing the overall NP-Hard cost in realistic computation time is difficult and requires an approximate method.

B. Time-dependent Task Sequencing

The planning problem of each agent's task sequence is defined as a time-dependent TSP (TDTSP).

$$\begin{aligned} &\min_x \sum_a \sum_b CTD(a, b, t_a) \cdot x_{a,b} \\ &\text{subject to} \\ &\sum_a x_{a,b} = 1 \ \forall b \in \Gamma_i \cup \{g_i\}, \\ &\sum_b x_{a,b} = 1 \ \forall a \in \{s_i\} \cup \Gamma_i, \\ &\quad t_{s_i} = 0, \\ &\quad t_b = \sum_a x_{a,b} \cdot (t_a + CTD(a, b, t_a)) \ \forall b \in \Gamma_i. \end{aligned} \tag{2}$$

Here, the cost function T_i is defined as $T_i := \sum_a \sum_b CTD(a, b, t_a) \cdot x_{a,b}$. The $x_{a,b} \in \{0, 1\}$ is a binary decision variable that considers the value 1 if the agent executes the task b following the task a, and it takes the value 0 otherwise. The $CTD(a, b, t_a)$ represents the time-dependent cost of the agent leaving task location a at time step t_a and moving to task location b.

C. Time-dependent Path Planning

Finding the shortest path between task locations and avoiding conflicts with other agents (i.e., dynamic obstacles) is given as the time-dependent shortest path problem.

$$\begin{aligned} f(n, t) &= g(n, t) + h(n, t), \\ n &\notin \text{reserved table}, \end{aligned} \tag{3}$$

where $g(n, t)$ is the cost of the path from the initial a node to n at time t, and $h(n, t)$ is the heuristic cost function from

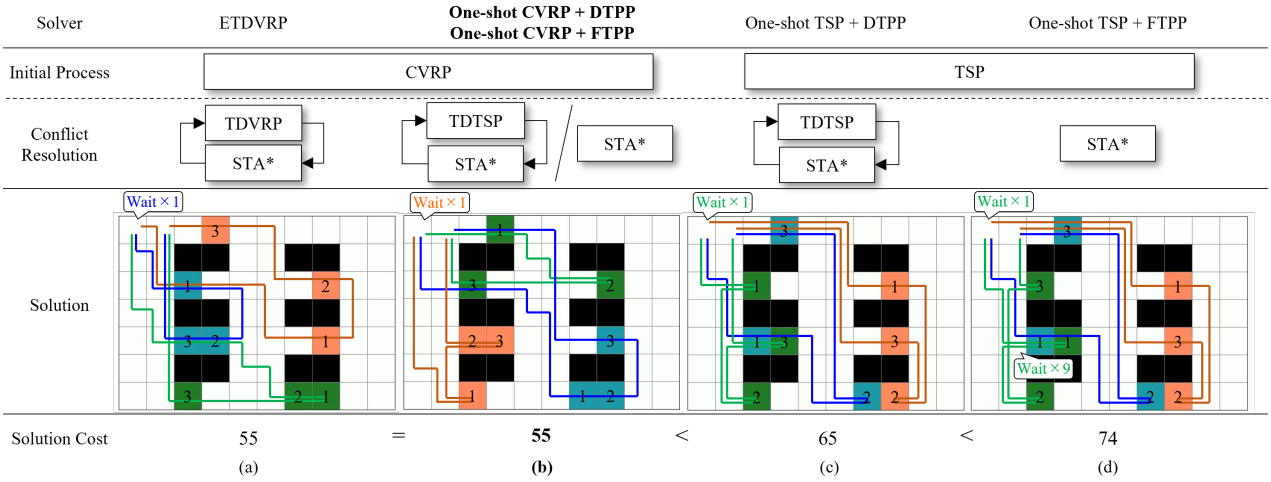

Fig. 2. Dual Process Optimization: (a) Extended time-dependent vehicle routing problem (ETDVRP) solver, a complete centralized optimization. (b) Dynamic task and path planning (DTPP) and fixed task sequence path planning (FTPP) solvers, both employing distributed optimization, are combined with centralized optimization by the CVRP solver. In this example, both found the same solution with a cost equivalent to that of the ETDVRP solver. (c) Adjusting task sequence and paths only by DTPP; the task allocation is not optimized. (d) FTPP only (path-only coordination); the green agent waits nine steps before task #1 owing to non-optimal task sequencing.

n to the final node b at time t. Node n can only be selected if a reservation table does not occupy it. The time-dependent shortest path problem can be solved by the algorithm space-time A* (STA*)[5], a modified A* with an extension in the time dimension.

IV. Dual-process Optimization Approach

Warehouse parts collecting and route planning necessitate optimizing both the part retrieval sequence and route. Specifically, the CVRP and MAPF should be dually optimized. A benchmark should be used for centralized optimization, considering the waiting time for picking tasks in the ETDVRP, as depicted in Fig.2 (a). This method incorporates the optimization of part retrieval by determining the cost management part of CBS through the TDVRP. The computational complexity of MAPF's route planning (left, right, up, down, wait), $O(5^N)$, has been partially solved using CBS. However, repeatedly solving the time-dependent CVRP until conflicts are resolved presents a bottleneck. In the worst-case scenario, a computational complexity of $X := MN, O(X!)$ is necessary. We demonstrated an algorithm intended to achieve a high speed by transitioning this bottleneck from the CVRP to the time-dependent TSP (TDTSP), thus solving $O(M!)$ even in the worst case, and also resolving the CBS, TDTSP, and STA to enhance solution performance.

A. Overview

As illustrated in Fig. 2 (b), (c), and (d), the proposed method computes solutions for the problem defined in Sec.III in two phases, namely initialization and conflict resolution, similar to the ETDVRP. During the initialization phase, routes with minimal travel costs are planned without considering inter-agent conflicts, which are then refined to be conflict-free paths in the conflict resolution phase. In the proposed approach, the conflict resolution phase decomposes the MAPF into a decentralized planning problem by using first-in-first-out (FIFO) structure for each agent. In this approach, high-priority agents plan their task sequence and paths first, marking their routes as reservations within the space-time grids. The following (i.e., low-priority) agents regard these reservations as dynamic obstacles while calculating their task sequences and paths. The details of initialization and conflict resolution algorithms are described in subsections IV-B, IV-C and IV-D.

The prioritization of each agent is a hyperparameter of this method and can be randomly chosen or determined based on the urgency of tasks in picking operations. Furthermore, seeking solutions under multiple prioritizations and adopting the best value as the final solution is acceptable.

The proposed method uses a decentralized approach based on individual agent planning optimization to resolve conflict resolution. However, both decentralized and centralized methods can be applied to the initialization. In the ETDVRP, the initial solution is centrally planned by the CVRP, followed by resolving conflicts through the iterative computation of the TDVRP, which is a type of centralized optimization. By contrast, the proposed method can either replace only conflict resolution or both initialization and conflict resolution with decentralized optimization.

B. Initialization

The initial plan Π_0 is established by combining A* with a one-shot TSP or CVRP solver, without considering inter-agent conflicts. An example of initialization with one-shot TSP is depicted in Fig.3.

One-shot TSP: The task sequence for each agent i is determined by solving the TSP, with start and goal nodes s_i, g_i, and task locations Γ_i. Although Euclidean or Manhattan distances are typically used to compute the cost matrix of TSP, the precise pre-computation of travel costs between

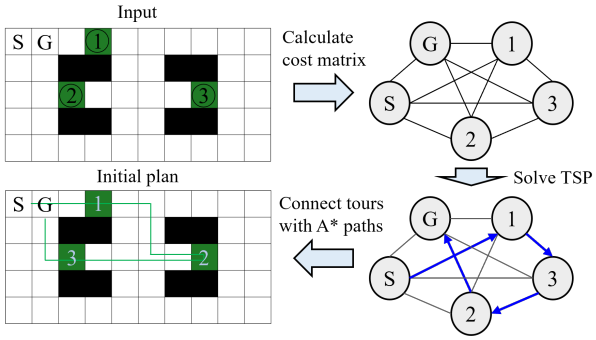

Fig. 3. Example of the procedure for computing the initial solution with one-shot TSP. *S* and *G* represent start and goal depot, respectively. A cost matrix based on distances between the start node, task locations, and goal node is used to solve TSP and determine the task sequence. The initial plan is subsequently formed by linking the tour obtained as the TSP solution with A* paths.

task locations by A* is possible in warehouse operations. Therefore, cost matrices that are based on A* paths are used in our method. For every distinct pair of locations $(a, b) \in \{s_i\} \cup \Gamma_i \cup \{g_i\}, a \neq b$, the path from node a to b obtained by A* and its cost are stored in $C_i(a, b)$ and $P_i(a, b)$, where C_i and P_i represents the cost matrix and associate paths for agent i , respectively. The initial plan for agent i is subsequently obtained by connecting the routes obtained by solving TSP with A* paths, referencing P_i. By applying the same initialization to all agents $i \in A$, a list of initial plans Π_0 is obtained.

One-shot CVRP: The procedure for planning the initial route using CVRP is identical to the initial processing in ETDVRP. Specifically, paths between each pair of different locations $(a, b) \in \mathcal{S} \cup \Gamma \cup \mathcal{G}, a \neq b$ are obtained by A*. The costs and the paths are stored in $C_{All}(a, b)$ and $P_{All}(a, b)$, where C_{All} and P_{All} represent the cost matrix and associate paths to solve the CVRP, respectively. Solving the CVRP with C_{All} yields each agent's route, which is subsequently connected using P_{All} to form the initial plan Π_0. Cost and path matrices for each agent C_i and P_i are obtained from C_{All} and P_{All}.

C. Conflict Resolution Through Dynamic Task and Path Planning (DTPP)

We proposed a novel algorithm that resolves route conflicts by adjusting task sequences and paths for each agent. The inputs are cost matrices, associate paths, and initial plans for each agent, denoted as C, P, and Π_0, respectively. Here, we define $C := \{C_1, \ldots, C_N\}$ and $P := \{P_1, \ldots, P_N\}$. In this approach, each initial agent i sequentially plans its route π_i and reserves the nodes. Subsequent agents then plan their task sequences and paths, avoiding the reserved nodes.

The procedure is detailed in Algorithm1. The individual planning for each agent has a two-level structure consisting of TDTSP and STA, as depicted in Fig.2 (b). The set $reserved$, which records the nodes and corresponding timestamps occupied by each agent, is initialized as empty. Furthermore, the solution list Π is initialized as empty [L2-3]. Elements are added to $reserved$ and Π as planning for each agent is finished.

Algorithm 1 Dynamic Task and Path Planning (DTPP)

Input: C, P, Π_0 ▷ Initial costs and associate paths, initial plan
Output: Π ▷ Set of conflict-free routes

1: **procedure** DTPP
2: $reserved \leftarrow \varnothing$
3: $\Pi \leftarrow \{\}$
4: **for each** i in $\{1, \ldots, N\}$ **do**
5: $\pi_i \leftarrow \Pi_0[i]$ ▷ Initial solution
6: $constraints \leftarrow reserved$ ▷ Constraints for STA
7: $C_i^{TD}, P_i^{TD} \leftarrow \{\}, \{\}$ ▷ Time-dependent cost and path
8: $conf \leftarrow$ First conflict between π_i and $reserved$
9: **while** $conf \neq \varnothing$ **do**
10: `/* Resolve conflict by STA */`
11: UpdateConstraint($conf$, $constraints$)
12: $a, b \leftarrow$ Start and end node of conflict path
13: $t_a \leftarrow$ Time when agent i departed from a
14: $new\_path \leftarrow$ FindNewPath$(a, b, t_a, constraints)$
15: `/* Re-plan task sequence */`
16: $C_i^{TD}\{a, b, t_a\} \leftarrow$ length(new_path)
17: $P_i^{TD}\{a, b, t_a\} \leftarrow new\_path$
18: $tour \leftarrow$ SolveTDTSP(C_i, C_i^{TD})
19: `/* Update route plan */`
20: $\pi_i \leftarrow$ LinkPaths$(tour, P_i, P_i^{TD})$
21: $conf \leftarrow$ First conflict between π_i and $reserved$
22: **end while**
23: Π.append(π_i)
24: $reserved \leftarrow reserved \cup \pi_i$
25: **end for**
26: return Π
27: **end procedure**

The solver begins the route planning of each agent i by detecting the first conflict between initial solution $\Pi_0[i]$ and $reserved$ [L5-9]. The set $constraints$ comprises the constraints for STA and is initialized with $reserved$. The conflict resolution starts with generating a novel constraint of STA based on the detected conflict in *UpdateConstraint* [L11]. For a conflict detected while stopping at a task location, $(\pi[t'], t')$ is added to $constraint$, where t' denotes the time of arrival at the task location. For a conflict detected while moving, an additional constraint is not added because the constraints for avoiding that conflict are already included in $reserved$ and $constraint$.

Based on the updated constraints, the solver plans an alternative path that avoids the detected conflict [L14]. In *FindNewPath*, the solver initially solves Eq.3 by running STA to find the shortest path that departs node a at time t_a and moves to node b, where a and b represent the start and goal nodes of the path containing the first conflict (i.e., the task locations from which agent i departs and its intended next destination), respectively. Subsequently, the path is expanded by the addition of stopping waypoints at node b location for a service duration of τ.

The new_path and its cost are subsequently added to time-dependent path dictionary P_i^{TD} and cost dictionary C_i^{TD} [L16-17]. Using the initial costs C_i and the time-dependent costs C_i^{TD}, the solver re-plans the task sequence, solving the TDTSP in *SolveTDTSP* [L18] to minimize the cost function

Algorithm 2 Fixed Task Sequence Path Planning (FTPP)

Input: C, P, Π_0 ▷ Initial costs and associate paths, initial plan
Output: Π ▷ Set of conflict-free routes

1: **procedure** FTPP
2: $reserved \leftarrow \varnothing$
3: $\Pi \leftarrow \{\}$
4: **for each** i in $\{1, \ldots, N\}$ **do**
5: $\pi_i \leftarrow \Pi_0[i]$ ▷ Initial solution
6: $constratins \leftarrow reserved$ ▷ Constraints of STA
7: $conf \leftarrow$ First conflict between π_i and $reserved$
8: **while** $conf \neq \varnothing$ **do** ▷ Conflict resolution
9: `/* Resolve conflict by STA */`
10: UpdateConstraint($conf$, $constraints$)
11: $a, b \leftarrow$ Start and end node of conflict path
12: $t_a \leftarrow$ Time when agent i departed from a
13: $new\_path \leftarrow$ FindNewPath$(a, b, t_a, constraints)$
14: `/* Update route plan */`
15: $t_b \leftarrow$ Time when agent i departed from b
16: $\pi_i[t_a; t_b] \leftarrow new\_path$
17: **end while**
18: Π.append(π_i)
19: $reserved \leftarrow reserved \cup \pi_i$
20: **end for**
21: return Π
22: **end procedure**

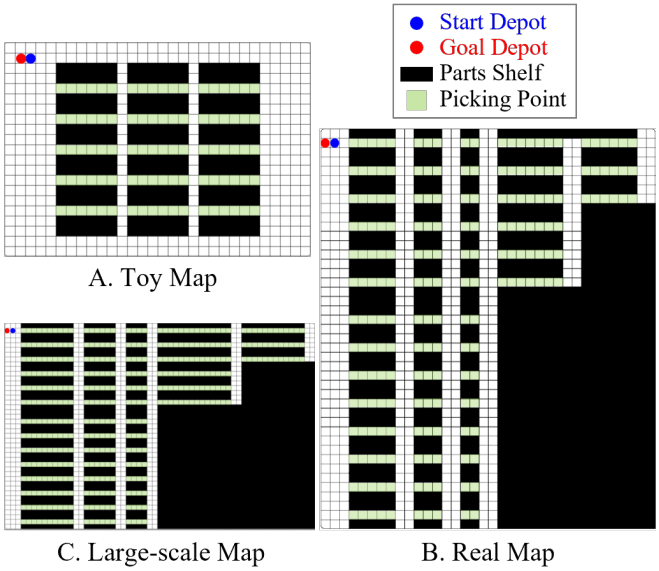

Fig. 4. Experimental setup with three maps. Map A has 90 picking points. Map B is based on the topographical information of an actual warehouse of a logistics center in Japan, comprising 200 picking points. Map C is an expanded version of Map B, featuring an increased number of picking points, totaling 400.

in Eq.2. Here, time-dependent costs in Eq.2 are obtained from C_i and C_i^{TD}. For the travel cost from node a to node b at time t_a, if $C_i^{TD}\{a, b, t_a\}$ has a value, the time-dependent cost is $C_i^{TD}\{a, b, t_a\}$. If $C_i^{TD}\{a, b, t_a\}$ is empty, then the cost defaults to $C_i(a, b)$. The new plan π_i is formed by connecting the tours obtained as a solution of TDTSP with paths, using P_i and P_i^{TD}.

The conflict resolution and re-planning of task sequences and paths are performed repeatedly. When the route plan π_i is modified to be conflict-free, the solver adds π_i to the list of solutions Π, adds the route $(\pi_i[t], t) \forall t \in \{0, \ldots, T\}$ to the set $reserved$, and proceeds to plan for the next agent $i + 1$ [L23-24].

D. Conflict Resolution Through Fixed Task Sequence Path Planning (FTPP)

We introduced a novel conflict-free path-planning method under a fixed task sequence. Compared with the ETDVRP and the DTPP, which address conflict resolution by adjusting task sequences and paths, this approach simplified the process by only adjusting paths. This approach is an extension of the existing CA*, with the following two enhancements: 1) the approach accommodates scenarios in which each agent has multiple destinations, and 2) the approach ensures that during the execution of each task, an agent is stationary for a certain service duration and does not conflict with higher-priority agents. The FTPP solver can be constructed by eliminating the step of resolving the TDTSP in the conflict resolution loop in the DTPP.

The pseudocode is described in Algorithm 2. The initial route planning and framework remain the same as those of the DTPP. The only difference is that when solving the STA [L13], the solver updates the current plan π_i by directly substituting the new_path for the conflicted segment from $\pi_i[t_a]$ to $\pi_i[t_b]$, without overhauling the entire plan [L15-16].

E. Examples

Fig.2 shows a summary of the dual process optimization solvers and the results of solving a simple example problem using each solver. The test instance was generated with the condition $N = 3, M = 3, \tau = 10$. The start and goal nodes $s_i, g_i, i \in A$ were placed side by side in the upper-left corner of the map. Black cells denote obstacles, whereas the colored grids indicate task locations Γ. Tasks with the same color are assigned to the same agent. The initial assignment of tasks to each vehicle is identical to that in the one-shot TSP + DTPP and one-shot CVRP + FTPP solvers' solutions.

In the solution obtained by the one-shot TSP + FTPP solver, the green agent waits for the blue agent to complete task #1. If the green agent were to bypass the blue agent and move to the location of task #1 from the opposite direction, the blue agent's movement from task #1 to task #2 would be obstructed. Therefore, this situation cannot be resolved only through path adjustment. By contrast, the one-shot TSP + DTPP solver adjusts the routes as well as the task sequence, yielding a lower-cost solution. Because multiple combinations of task order and paths exist with the same cost available for the green agent, the combination that does not result in waiting owing to conflicts with the blue agent is appropriately selected.

The one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers achieved lower costs because task assignments were optimized in the initial process. Because the size of the example problem was small, both solvers output the same solution with a cost equivalent to that of ETDVRP's solution.

TABLE I
EXPERIMENT RESULTS WITH TOY MAP AND REAL MAP

Method	Toy Map			Real Map		
	Cost	**Time[s]**	**Gap[%]**	**Cost**	**Time[s]**	**Gap[%]**
ETDVRP [Centralized]	486	657.6	0	1088	1191.1	0
One-shot CVRP + DTPP	**481**	22.4	**-1.0**	**1071**	13.3	**-1.6**
One-shot CVRP + FTPP	482	10.9	-0.8	1083	12.5	-0.5
One-shot TSP + DTPP	669	63.3	37.7	1690	46.9	55.3
One-shot TSP + FTPP	710	22.0	46.1	1700	14.8	56.3
TSP+A* [Decentralized]	824	**5.5**	69.5	1807	**3.7**	66.1

V. EXPERIMENTAL SETTINGS

We used three distinct maps, as depicted in Fig.4. The black cells represent the shelves (static obstacles) over which the agents cannot pass. We designed narrow pathways between the shelves to maximize warehouse storage efficiency so that the agents cannot pass each other in these aisles. Regarding the task design, we set all agents to start simultaneously from a common starting node (blue circle). They visited their respective task locations to pick up items from shelves and subsequently returned to the goal node (red circle). We used the *Toy map* and *Real map* to solve the problems within a scale manageable by all solvers, enabling comparisons of performance and computation time. We specifically designed the *Large-scale map* to investigate the solvers' capabilities with an increase in total task number to explore the extent of each solver's solutions.

We set the service time τ to 10 in the experiments and the number of agents N to 10 for the *Toy map* and 15 for the *Real map* and "Large scale map." We determined these parameter configurations based on actual logistics operations and randomly assigned task locations to agents from the green grid areas in each map for each test instance in Fig.4.

In addition to the methods introduced in Sec.IV, we implemented the "TSP+A*" solver. This solver plans each agent's task sequence by solving the conventional TSP and its paths by linking task locations with standard A* sequentially, without considering conflicts between agents. This approach is completely decentralized.

All solvers were implemented in Python 3.8 and run on a laptop computer (Intel Core i7-1195G7 CPU @2.90 GHz and 32GB RAM). We used the open-source library Python-TSP[1] for the standard TSP solver, and Google OR- tools[2] for the standard CVRP solver. The TDVRP solver incorporated the iterated local search (ILS) solver described in [8]. We solved the TDTSP by adapting this TDVRP solver with the vehicle number constraint set to one. We set the hyperparameter for the number of iterations in ILS N_{iter} to one, as preliminary experiments revealed that increasing the number of iterations did not significantly improve the cost of the solution.

[1]https://pypi.org/project/python-tsp/

[2]https://developers.google.com/optimization/

VI. RESULTS AND DISCUSSION

A. Experiments with Toy Map and Real Map

To compare the costs of the solutions and the computation time of each method, we conducted experiments in which common test instances were solved by each solver. The map data used were *Toy map* and *Real map*, as depicted in Fig.4, with the number of tasks for each agent M set to five.

Results: Table I summarizes the experimental results, and the solutions obtained by the one-shot TSP + DTPP and one-shot CVRP + DTPP are shown in Fig.5 as an illustrative example. The metric "Cost" indicates the number of waypoints in the plans obtained by each solver. Service time waypoints on task locations are excluded from this metric as they are constant regardless of the solver used and depend only on the setting of the service time τ. The solution obtained by TSP+A* differs from those of other solvers as it includes agent collisions in the plan. Therefore, the waiting time because of collisions is estimated by simulation and added to the cost. "Time" indicates the CPU time each solver requires to complete the calculations. The computation times of the cost matrices are not included here because cost matrices are assumed to be provided in advance. "Gap" depicts the percentage increase in cost for other solvers compared to the results obtained by the ETDVRP solver, which is a completely centralized solver and searches low-cost solutions aggressively.

Table I reveals that the ETDVRP solver required the longest computational time. By contrast, the one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers generated solutions equivalent to that of the ETDVRP solver in cost and were 10–60 times faster, respectively. One-shot TSP + DTPP and one-shot TSP + FTPP solvers improved costs by approximately 6–19% compared with the completely decentralized approach (TSP+A*). However, a gap of approximately 28–37% exists among solution costs compared with that of the solvers with one-shot CVRP.

Computational Times: The difference in the computational time between the ETDVRP solver and the one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers can be attributed to the difference in the computational load of conflict resolutions. The one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers optimize the task allocation in the initial process and subsequently address the remaining conflict resolution in a decentralized manner. By contrast, the ETDVRP solver repetitively solves the central optimization

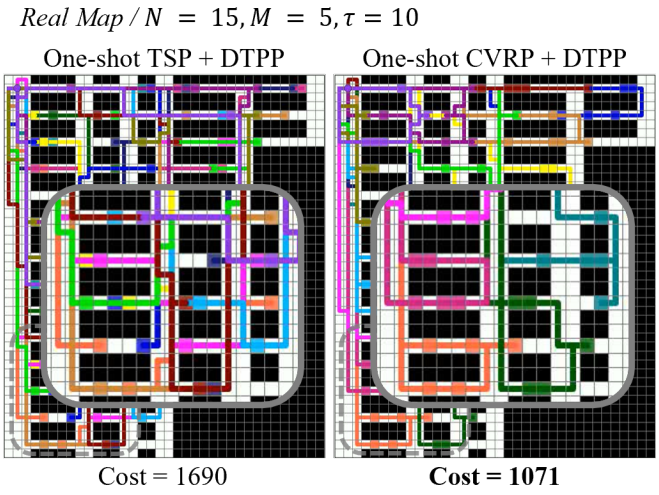

Fig. 5. Example of plans obtained by solvers with DTPP. Colored grids represent task locations. Different colors represent different agents assigned to the tasks. Colored lines represent the paths of each agent. The one-shot TSP + DTPP solver does not optimize task allocation to agents; therefore, the solution contains many overlaps between each agent's path. By contrast, one-shot CVRP + DTPP reduces the number of overlaps.

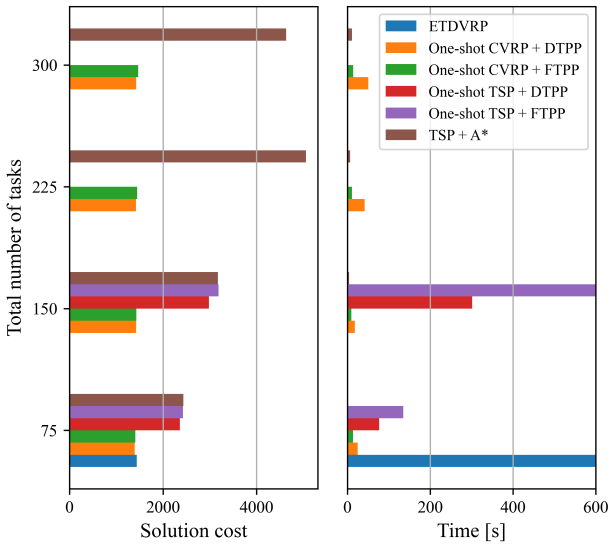

Fig. 6. Costs of solutions obtained by each solver and processing time. The right subplot's y-axis is limited to 350 s. The ETDVRP solver required 1697.7 s when the total number of tasks was 75 and failed to determine solutions with more than 150 tasks. The one-shot TSP + FTPP solver took 961.8 s for 150 tasks. Both one-shot CVRP + DTPP and one-shot TSP + FTPP solvers could not find solutions with more than 225 tasks.

TDVRP during conflict resolution. The TDVRP requires a higher number of computations until convergence. Profiling conducted on the *Toy map* experiments revealed that the ETDVRP, one-shot CVRP + DTPP, and one-shot CVRP + FTPP solvers computed the conflict resolution loops 290, 33, and 19 times, respectively. In the *Real map* experiments, the corresponding counts were 1010, 42, and 33 times, respectively.

When comparing the DTPP and FTPP solvers with one-shot TSP and one-shot CVRP, the computational time is shorter with CVRP. As shown in the left image of Fig.5, when CVRP is not solved, each agent's path frequently intersects with the other agents when moving between task locations. By contrast, when CVRP is solved, the task locations of each agent are grouped in close proximity, as depicted in the right image of Fig.5. Therefore, each agent's path only collides with other agents' paths when moving from the start depot to the first task location and from the last location to the goal depot, thereby reducing the total computational load.

Cost Improvements: The result indicates that the one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers exhibit excellent heuristics compared with the ETDVRP. The ETDVRP solver can leave unnecessary waits in the solution, whereas the proposed methods do not. Furthermore, through comparisons between DTPP and FTPP, with one-shot TSP and with one-shot CVRP, the effectiveness of including the task sequence as an optimization variable in conflict resolution is demonstrated. In experiments conducted using the *Toy map* and *Real map*, solvers with DTPP consistently revealed lower-cost solutions than the solvers with FTPP.

B. Experiments With Large Scale Map

To identify the problem size that each solver can compute, we conducted experiments by increasing the number of tasks for each agent M and solving common test instances using each solver. We used the *Large-scale map* depicted in Fig.4 when conducting experiments with many task locations. We set the runtime limit to 60 min for each instance.

Results: Fig.6 depicts the solution costs obtained by each solver for each test instance and the corresponding computation times. The one-shot CVRP + DTPP, one-shot CVRP + FTPP, and TSP+A* solvers determined solutions in all cases. The ETDVRP solver could not obtain solutions when the total number of tasks $N \times M$ exceeded 150, and both DTPP and FTPP were not successful when the task count exceeded 225. A trade-off was observed between performance and processing time between one-shot CVRP + DTPP and one-shot CVRP + FTPP solvers.

Capability: The size of the cost matrix for the TDVRP, solved by the ETDVRP in conflict resolution, is $(N \times M)^2$. By contrast, the proposed method decomposes conflict resolution into decentralized optimization. For DTPP, the size of each agent's cost matrix for the TDTSP is M^2. The worst-case computational complexity of TDVRP is $O((N \times M)!)$, whereas that of solving TDTSP with N agents is $O(N \times (M!))$. Consequently, DTPP and FTPP effectively solve large-scale problems that cannot be solved using the ETDVRP solver.

Furthermore, when comparing solvers with one-shot TSP and with one-shot CVRP, solvers exhibit a sharp increase in computation time and cannot obtain solutions as the total number of tasks increases. By contrast, the one-shot TSP

exhibits a controlled increase in the computation time and obtains solutions in all cases, as depicted in Fig.6. This phenomenon can be attributed to the reduced number of conflicts to be resolved due to the decreased overlap in agents' routes because of the optimization of task assignments in CVRP, as discussed in Sec.VI-A.

Cost Improvements: When comparing solvers with DTPP and FTPP, those of DTPP consistently find solutions with lower costs than those obtained using FTPP. Similar to the results in Sec.VI-A, this phenomenon indicates the effectiveness of optimizing route adjustments and including the task order during individual agent planning.

VII. Conclusion

We proposed a novel dual-process optimization approach for parts collection orders and route planning in parts warehouses. We identified these challenges as two distinct NP-hard problems, the TSP and route planning, and subsequently developed a novel dual-process optimization method. The proposed method outperformed previous centralized optimization approaches, achieving over ten times higher execution speeds and cost improvements of up to 1.6%, compared with the previous meta-heuristics. We demonstrated that the proposed approach effectively balances performance and speed and maintains an efficient equilibrium between centralization and distribution through six comparative experiments, involving independent to fully optimal methods.

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9, Mar. 2008.

[2] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. Conflict-based search with optimal task assignment. In *Proc. of International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 757–765, 2018.

[3] H. Ma, J. Li, T. S. Kumar, and S. Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proc. of International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 837–845, 2017.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[5] D. Silver. Cooperative pathfinding. In *Proc. of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 117–122, 2005.

[6] G. Sharon, R. Stern, A. Felner, and N. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, pages 563–569, 2012.

[7] G. Sharon, R. Stern, A. Felner, and N. Sturtevant. Meta-agent conflict-based search for optimal multi-agent path finding. In *Proc. of the International Symposium on Combinatorial Search (SoCS)*, pages 97–104, 2021.

[8] A. Aggarwal, F. Ho, and S. Nakadai. Extended time dependent vehicle routing problem for joint task allocation and path planning in shared space. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12037–12044, 2022.

[9] J. Li, W. Ruml, and S. Koenig. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 12353–12362, 2021.

[10] H. Ma and S. Koenig. Optimal target assignment and path finding for teams of agents. In *Proc. of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1144–1152, 2016.

[11] C. Henkel, J. Abbenseth, and M. Toussaint. An optimal algorithm to solve the combined task allocation and path finding problem. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4140–4146, 2019.

[12] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. Conflict-based search with optimal task assignment. In *Proc. of International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 757–765, 2018.

[13] J. Li, A. Tinka, S. Kiesel, J. D. Durham, T. S. Kumar, and S. Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11272–11281, 2021.

[14] C. Leet, C. Oh, M. Lora, S. Koenig, and P. Nuzzo. Task assignment, scheduling, and motion planning for automated warehouses for million product workloads. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7362–7369, 2023.

[15] C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[16] D. L. Applegate, R. E. Bixby, V. Chvátal, W. Cook, D. G. Espinoza, M. Goycoolea, and K. Helsgaun. Certification of an optimal tsp tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15, 2009.

[17] K. Helsgaun. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical report*. Roskilde Universitet, 2017.

[18] M. Yuki, Y. Tomoki, and S. Toshiharu. Distributed planning with asynchronous execution with local navigation for multi-agent pickup and delivery problem. In *Proc. of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 914–922, 2023.