Cell-Based Local Search Heuristics for Guide Path Design of Automated Guided Vehicle Systems With Dynamic Multicommodity Flow

Tatsushi Nishi, *Senior Member, IEEE*, Shuhei Akiyama, Toshimitsu Higashi, *Member, IEEE*, and Kenji Kumagai, *Member, IEEE*

***Abstract*— This article discusses the guide path design of automated guided vehicle (AGV) systems for which we propose a model that incorporates a dynamic multicommodity flow with capacity constraints from a pickup to a delivery point. The problem is formulated as the selection of a guide path connecting a given set of pickup/delivery points. A cell-based local search that is based on an original neighborhood technique is developed. In the proposed method, the guide path is effectively optimized based on a local search of cell-based neighborhood search preserving connectivity constraints and the subsequent solution of a dynamic multicommodity flow problem. The performance of the proposed method is strengthened by the redundant elimination of arcs and intensification of search according to the concept of the flow-concentrated cell. The effectiveness of the proposed method is demonstrated by comparing it with a general-purpose solver and recent algorithms for fixed-charge capacitated multicommodity network design problems. A real case study is presented to demonstrate the applicability of the proposed method by using simulation software.**

***Note to Practitioners*—Due to the recent growth of online shipping, the design of automated warehousing systems is receiving much attention. Most automated warehousing systems introduce multiple automated guided vehicles (AGVs) for transportation. This article presents an efficient optimization of guide path design considering congestions and dynamic multicommodity flow routing. We develop cell-based local search heuristics for the guide path design problem for AGV systems that can be applied to a large-scale transportation model to solve the problem efficiently. Our major finding is that the guide path design with the dynamic multicommodity flow can significantly reduce the delays caused by congestions and improve the efficiency of the warehousing systems. The effectiveness of the derived guide path design is evaluated by simulation software. The feasibility of the guide path is confirmed in several case studies.**

***Index Terms*— Automated guided vehicles (AGVs), block layout, dynamic multicommodity flow, flow path design, heuristics.**

Manuscript received January 25, 2019; revised May 18, 2019, September 5, 2019, and October 26, 2019; accepted October 29, 2019. Date of publication December 13, 2019; date of current version April 7, 2020. This article was recommended for publication by Associate Editor R. Cordone and Editor J. Li upon evaluation of the reviewers' comments. This work was supported in part by the Japan Society for the Promotion of Science under Grant 17K18951 and in part by the Japan Keirin Autorace Association (JKA) under Grant 2017M-158. This article was presented in part at the 2017 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore, 2017. *(Corresponding author: Tatsushi Nishi.)*

T. Nishi and S. Akiyama are with the Division of Mathematical Science for Social Systems, Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, Osaka 560-8531, Japan (e-mail: nishi@sys.es.osaka-u.ac.jp).

T. Higashi and K. Kumagai are with Murata Machinery, Ltd., Kyoto 612-8418, Japan (e-mail: toshimitsu.higashi@akl.muratec.co.jp; kenji.kumagai@hkt.muratec.co.jp).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2019.2952920

I. Introduction

With the increasing demands of online shopping, automated guided vehicle (AGV) systems are widely used in warehousing, container terminals, and flexible manufacturing systems. To achieve cost-effectiveness, labor-saving, efficiency, and increased space utilization, the design of a guide path layout is one of the critical features of an AGV system [17]. The main problem addressed by this article is the determination of the connections or the selection of guide path lanes when the pickup/delivery (P/D) points are given. In practice, this determination or selection is made by human operators with much experience and knowledge of transportation systems design by considering various factors, such as constraints, specifications, cost minimization, and several uncertainties, by using discrete-event simulators. However, simulation-based optimization techniques are very time-consuming and require a lot of computational effort when the number of decision variables and design parameters increases. It is highly desirable to derive a near-optimal solution with higher accuracy for reducing the total time required for the layout design process. The objective of this article is to develop an efficient model and algorithm to solve the guide path layout design problem of AGV systems.

The network flow model has been used to represent the guide path design of AGV systems. A common approach for guide path design is to find an optimal block layout that consists of the set of cells dividing the whole guide path. Kim and Tanchoco [16] applied a mathematical model to the design of material flow paths in a single-loop AGV system. Guide path layouts are classified into single- or tandem-loop types. An example of an AGV system with multiple layout types is shown in Fig. 1(a), and the schematic layout of the application of a guide path design to a warehousing system is shown in Fig. 1(b). A single-loop layout comprises a single forward circuit and its combinations, whereas a tandem-loop consists of a multicycle loop that has some P/D points with a transfer path that connects two-cycle loops. The tandem loop has become one of the major guide path layouts in recent

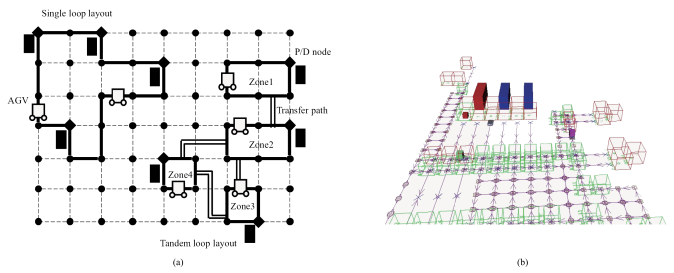


Fig. 1. Example of AGV system with multiple layout types and the guide path design of the transport system in a warehouse. (a) Guide path example including single and tandem loops. (b) Guide path design of the transport system in warehouse.

years. For example, the tandem-loop layout is regarded as more effective than conventional guide path models in [8]. The definition of a guide path also has unidirectional or bidirectional lanes.

Generally, bidirectional lanes are more complex than unidirectional lanes. Most of the related works deal with the latter. Asef-Vaziri and Goetschalckx [3] presented a mathematical model with the dual-truck, which is the allocation of two parallel paths in the same place. Efficient solution algorithms for the mathematical model have also been studied by Asef-Vaziri *et al.* [5], who applied the ant colony optimization (ACO) method and Rubaszawski *et al.* [27] developed a genetic algorithm for the model. Research on material handling flow design has focused on a few types of guide path layouts, but most of the studies have been limited to block layouts (see [13], [16], and [29]). There is little research that has considered both the general flow path design problem and conflict-free routing problem. Most of the related works do not consider traffic congestions of vehicles in the guide path design problem. These congestions are caused by the dynamic behavior of the vehicles. The models used in the design of the guide path do not consider dynamic features of the system. Zhang *et al.* [35] developed a layout design problem taking into account the congestion of flow routing in a manufacturing/warehouse facility. In their formulation, a multicommodity flow problem with link capacity is incorporated into the flow path design problem. However, it becomes extremely difficult to obtain a near-optimal solution when the size of the system increases [36].

In this article, we propose an efficient algorithm for the guide path design of AGV systems with a dynamic multicommodity flow model, which has more generality where the set of nodes is freely located, while the routing considers dynamic behaviors. Our main objective is to integrate the flow path design and the guide path design by considering the dynamic behavior AGVs to avoid deadlocks and congestion. In the proposed approach, routing decisions are included in the design of the guide path layout problem considering the collisions and conflicts among vehicles in detail. The proposed model using a dynamic multicommodity flow model is compared with the detailed routing model, which considers the dynamic conflict-free routing problem in the design of the guide path. On the other hand, the proposed model considers the dynamic multicommodity flow and buffers for the selection of the guide path. We compare the performances of these two models for the design of a real AGV system.

To solve the problem efficiently, cell-based local search heuristics are applied. The proposed method is based on cell-based local search heuristics to decompose the problem into a dynamic multicommodity flow problem and a cell-based selection problem. The redundant arc elimination technique is used to intensify the search. The proposed method is applied to a real case study. The results of the guide path layout design derived by the proposed method are evaluated by simulation software (AutoMod). Then, the effectiveness of the proposed method is demonstrated by computational experiments. Our contributions are summarized as follows.

1) We have developed a dynamic multicommodity flow model for the guide path design taking into consideration the dynamic routing of AGV systems. While considering the traffic congestions of vehicles, the proposed model can generate a guide path design that is similar to those generated by human operators.
2) We propose cell-based local search heuristics to solve the guide path design problem efficiently. The local search is effectively conducted according to the concept of cells, which constitutes a guide path design with good accuracy.
3) The proposed method is compared with a general-purpose solver and recent heuristic algorithms for capacitated fixed cost multicommodity network design problem.
4) The derived guide path design has been confirmed to be able to derive a guide path design structure similar to one that can be derived by a human operator but at a lower cost.

The rest of this article is organized as follows. Section II introduces the related works. Section III explains the problem description and formulation. We propose cell-based local search heuristics to solve the problem in Section IV. Computational results are presented in Section V. Finally, Section VI concludes this article and states our future work.

II. Related Works

The design and control of AGV systems have been extensively reviewed in [17] and [33]. Gaskins *et al.* [13] developed a multicommodity flow for determining flow paths for free-ranging AGVs. The guide path layouts of AGV systems are classified into several types, among which are single-loop and tandem layouts. Introduced by Bozer and Srinivasan [8], the tandem layout has some advantages, such as the simplicity of the systems control and the avoidance of congestion and deadlocks. However, the efficiency of the layout depends significantly on the partitioning of the layout into several tandem configurations. For example, Fan *et al.* [12] developed a hybrid algorithm to design a tandem-loop layout based on a genetic algorithm and a simulated annealing algorithm. These works evaluate the performance of the layout following the costs of intrazone and interzone flows. A single-loop layout has features similar to a tandem one. Sinriech and Tanchoco [30] outlined an optimal procedure for designing a single-loop system and P/D station location. In their work, the design procedure has multiple steps, and the guide path design is evaluated only by the length of its arcs. In later years, several works on the single-loop layout have appeared. Caricato *et al.* [9] also presented a model to create a single-loop layout as well as proposed a branch and cut framework and tabu search heuristics. Asef-Vaziri *et al.* [5] presented a mathematical model and proposed heuristics based on an ant colony system to create a single-loop-based layout on a block layout. Due to the recent increases in the power of CPUs, the literature on concurrent design has increased. Asef-Vaziri and Goetschalckx [3] proposed a method of simultaneously designing a single-loop layout and the locations of the pickup and drop-off points. Asef-Vaziri and Kazemi [4] addressed the covering and connectivity constraints in the single-loop formulation of flow path network design problems. They provided a method with efficient subtour elimination to improve the performance of branch and cut algorithms. The abovementioned studies have evaluated the performance with the total flow traveling costs on the layout. The total flow traveling cost is the quantity of loaded and empty flows multiplied by the travel distance. The empty flow was introduced by Maxwell and Muckstadt [19] to represent dynamic vehicle movements in the static model, while the concept of empty flow has been used in many related works. On the other hand, we have extensively studied and published many works on efficient conflict-free routing methods for AGV systems (see [20]–[26], [31]). Loop-based layouts are commonly designed with a fixed layout candidate. A block layout is one of a major layout that comprises a set of fully packed adjacent polygons. Several loop-based layouts are used for selecting the paths on the block layout candidates [4]–[6], [9], [28]. The abovementioned studies on single-loops have also created the layouts on the block layout type. This article discusses the guide path design of AGV systems while considering conflict-free routing. The main feature of our proposed method is its consideration of the deadlocks and congestions caused by the dynamics of vehicles. If the length of each path is different, it is difficult to design layouts while considering vehicle dynamics. Although a single-loop layout is simpler than other layout types, the shortest single-loop configuration problem has proved to be NP-hard [15]. A multicommodity flow problem without considering the multiple time periods has also proved to be NP-hard [11].

Recently, many efficient algorithms for multicommodity network design problems have been reported. Crainic *et al.* [10] developed a slope scaling/Lagrangian perturbation heuristic (SS/LPH) with long-term memory for multicommodity capacitated fixed-charge network design problem. Boland *et al.* [7] proposed an iterative refinement algorithm using partially time-expanded networks that solve continuous-time service network design problems. Yaghini and Foroughi [34] developed an ACO-based neighborhood search algorithm for the problem. However, the dynamic multicommodity flow decision over time is not considered in these models. Continuous-time service network design problem [7] and uncapacitated time–space fixed-charge network flow problem [14] incorporate network design decision to the multicommodity flow overtime problem. These general models do not consider the connectivity constraints of the generated network that may appear in the practical guide path design of AGV systems.

An efficient algorithm considering the dynamic traffic flow of AGV systems is more challenging and required for general guide path design. Efficient algorithms for the guide path design problem have also been proposed. Generally, in the case of more than two elements being determined simultaneously, the mathematical model is divided into several parts and solved in step-by-step approaches. Asef-Vaziri *et al.* [5] addressed an ant colony-based heuristic algorithm to generate feasible loops. Lim *et al.* [18] developed a Q-learning algorithm to design the guide-path networks. Asef-Vaziri *et al.* [6] developed a two-step method. At first, valid and feasible loops are enumerated, and the solution on these layouts is derived by a mathematical model to obtain the optimal solution. To reduce computational efforts, we also propose efficient cell-based local search heuristics. The proposed method consists of two stages to solve the original problem sequentially. These stages are repeated until a near-optimal solution is derived. Our proposed model employs a more complex model since the dynamics of the vehicles are considered. Therefore, the exact algorithm may not be effective. In the proposed method, the search of the guide path is strengthened by updating the guide path with congested areas. Then, the solution of the dynamic multicommodity flow problem is used as an evaluation for the performance of the guide path design. To improve the efficiency of the search, we introduce the concept of a cell that is a set of arcs, and these arcs constitute a loop. The performance of the local search heuristics is evaluated by using a general-purpose solver. The validity of the derived guide path is evaluated by simulation software using real parameters.

III. Problem Description

Consider the situation of an outline of the guide path design being requested from a designer of an AGV system. A set of P/D locations, facility layouts, and demands of material flow from P/D locations are given. If the dynamic features of the system are neglected in the design of the guide path, there may be congestion among the vehicles. Then, the throughput of the system may be decreased. To address the dynamic features of the system, we can consider the following two types of problem formulations. The first formulation is a detailed model with dynamic vehicle routing, and the second one is the dynamic multicommodity flow model proposed in this article.

A. Problem Description of the Detailed Model With Dynamic Vehicle Routing

The transport system is represented by a set of nodes and arcs. Each node represents a pickup point, delivery points, and a candidate of intersection points, while each arc represents a candidate segment of a path between two adjacent nodes where a vehicle can travel. All possible selections of the guide paths are represented by a set of arcs.

Fig. 2(a) shows the outline of an AGV transport system model. The material flow requirements between the P/D points are represented by a from/to matrix. To ensure safety in transportation, all P/D nodes must be connected and subtours must not be permitted. To represent the avoidance of congestions and conflicts by vehicles, the vehicles cannot be on

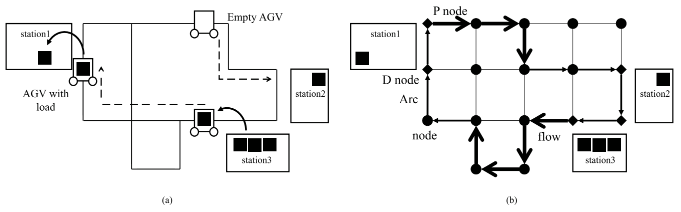


Fig. 2. Outline of transport system and the dynamic multicommodity flow path design. (a) Model of AGV transport system. (b) Dynamic multicommodity flow model of AGV transport system.

each arc or node at the same time. The traveling velocity is assumed to be constant. The total planning horizon is divided into several time periods. All transport demands must be completed by the end of the planning horizon. The P/D times are negligible with respect to traveling time.

The problem is to determine an optimal selection of the guide path such that the objective function is minimized. The objective function is the weighted sum of the total distance of flow and the fixed cost to select each lane. The input data of the detailed model are as follows: candidates of guide paths, a set of transportation demands, total planning horizon, location, number of P/D nodes, and number of vehicles.

To consider the dynamic feature of the routing model, we have developed a mixed-integer linear programming model that incorporates the routing problem in the design of guide paths [1]. However, a detailed design model considering the dynamic behavior of AGV routing is too complicated when the vehicle routing decisions are represented by integer variables. Therefore, in Section III-B, we propose a dynamic multicommodity flow model that can accommodate the dynamics of material flow to reduce computational complexity.

B. Dynamic Multicommodity Flow Model

In the proposed model, the dynamics of material flows are represented by a dynamic multicommodity flow shown in Fig. 2(b). The material flow starts from its pickup location and ends at its delivery location with the shortest time routing to satisfy the demands. The transportation demand constraints can be represented by the constraints on product buffers. The conflicts among vehicles are replaced with the capacity constraints on each node and arc. Two or more than two flows cannot be on the same node at the same time.

We set an unspecified number of vehicles, but we can estimate the fleet size from the total material flow and transportation capacity of the vehicles by $\sum_{l,m} f_{lm}(D_{lm}/S+L+E)/U$, where f_{lm} is trip frequency per shift on route (l,m) derived from dynamic multicommodity flow model, D_{lm} is the trip distance of route (l,m), S is the average travel speed, L is load/unload time per trip, E is trip inefficiency, which includes idle time and so on, and U is the average operational time per shift [16].

There are some changes from the detailed model described in Section III-A. The main change is the proposed models not considering routing empty vehicles that have delivered their loads. However, the base idea of the multicommodity flow balance and conflict avoidance constraints is the same as that of the detailed model. Our objective is to develop an optimal guide path. Therefore, these changes may have less impact on the optimality of the guide paths. We assume the following conditions in the problem definition.

1) All possible intersections, arcs, and P/D nodes are given.
2) Bidirectional or unidirectional lanes are assumed.
3) The guide path has connectivity in which all P/D nodes are connected.
4) Each flow can travel into a neighborhood node during a time period.

The constraints on the proposed model are listed as follows.

1) All P/D nodes are connected.
2) The transportation demand should be satisfied within the planning horizon.
3) There are no collisions or conflicts among flows.
4) The number of flow requirements is defined by the number of pairs of P/D points of the transportation demand.
5) There are no P/D points at the same location.

C. Problem Formulation

The mathematical representation of the proposed model is introduced by using the notation in Table I. The transportation system is represented as a graph $G=(N,A)$, where each node $i \in N$ represents an intersection where each material flow changes its directions, each directed arc $(i,j) \in A$ $(i \neq j)$ represents a lane where each flow can travel on unidirectional way, and each self-loop arc $(i,i) \in A$ represents waiting at node i. If both arcs (i,j) and (j,i) are selected as a lane, the lane is bidirectional. The specific nodes, pickup node (P node) and delivery node (D node), represent the source and sink nodes where each flow arises or disappears. We assume that the graph G has strong connectivity for all P and D nodes. Each arc has a unit per flow cost $l_{i,j}$, fixed cost $m_{i,j}$, and flow capacity of $c_{i,j}$. Each material flow has a single source $o_k \in N$ and a single sink $d_k \in N$. Let K be the total time horizon. A time–space network with discretization

TABLE I
NOTATION FOR THE DYNAMIC MULTICOMMODITY FLOW MODEL

Sets:	
A	set of arcs
F	set of flow requirements
H	set of time periods
N	set of nodes
$\delta^+(i)$	set of successors of node i
$\delta^-(i)$	set of predecessors of node i
Parameters:	
e_t^k	demand quantity of material flow k at node d_k in time t
f_t^k	supply quantity of material flow k at node o_k in time t
$c_{i,j}$	capacity on arc (i,j)
o_k	pickup node for material flow k
d_k	delivery node for material flow k
K	total time horizon
$l_{i,j}$	travel cost per unit flow on arc (i,j)
$m_{i,j}$	fixed cost of arc (i,j)
w_1, w_2	weighting factors for fixed costs and travel costs
Decision variables:	
$x_{i,j,t}^k$	quantity of material flow k from node i to node j in time t
b_t^k	quantity of material flow k on node d_k in time t
$y_{i,j}$	binary variables: 1, if arc (i,j) is selected to a guide path and 0 otherwise.

$\{t = 0, 1, 2, \ldots, K\} \in H$ is used to represent dynamic multicommodity flow. Without loss of generality, we assume that the traveling time for each material flow between neighbor two nodes is one. The conflicts among the material flows are avoided by imposing flow capacity constraints, and the flow velocity is assumed to be constant. The set of flow requirements is represented by F. Each flow requirement $k \in F$ consists of the node numbers of P node (o_k) and D node (d_k) and flow requirement. f_t^k is the quantity of flow entered from P node (o_k) at time t. e_t^k is the demand of flow entered into D node (d_k) at time t. The material flow is represented by continuous variable $x_{i,j,t}^k \in \mathbb{R}$, the quantity of material flow k on arc (i, j) at time t, where $\mathbb{R}$ is the set of real numbers. The quantity of material flow k on node d_k at time t is represented by continuous variable $b_t^k \in \mathbb{R}$, and the demand e_t^k is satisfied from node d_k. The variable $y_{i,j} \in \{0, 1\}$ takes 1 if arc (i, j) is selected as a lane and 0 otherwise. The problem is to determine the guide paths for the commodifies to minimize the total weighted sum of the fixed and flow traveling costs, which can be formulated as the following mixed integer linear programming problem:

$$\min\left(w_1 \sum_{(i,j)\in A} m_{i,j} y_{i,j} + w_2 \sum_{t\in H}\sum_{k\in F}\sum_{(i,j)\in A} l_{i,j} x_{i,j,t}^k\right). \tag{1}$$

The objective function (1) is to minimize the weighted sum of the fixed costs for selecting an arc to a guide path and the total flow traveling costs of material flows over the time horizon, which increases with respect to the flow traveling time because the cost of waiting at node i is incurred by $l_{i.i}$ if $x_{i,i,t}^k > 0$ at time t

$$\sum_{j\in\delta^-(i)} x_{j,i,t}^k - \sum_{m\in\delta^+(i)} x_{i,m,t+1}^k = \begin{cases} -b_t^k + b_{t+1}^k + e_{t+1}^k, & (i = d_k) \\ -f_t^k, & (i = o_k) \\ 0, & (i \neq o_k, d_k) \end{cases}$$
$$(\forall i \in N \quad \forall k \in F\ \forall t \in H). \tag{2}$$

$\delta^+(i) = \{j | (i, j) \in A\}$ and $\delta^-(i) = \{j | (j, i) \in A\}$. Constraints (2) are the material flow balancing constraints, indicating that the total quantity of incoming flows into a node i is equal to the total quantity of outgoing flows from node i to its neighbor nodes. If node (i) is D node (d_k), the demand of material flow k disappears at the node d_k. The change of material flow at node d_k between time t and time $t + 1$ ($b_{t+1}^k - b_t^k$) is equal to the change of input material flow minus output demand e_{t+1}^k If node (i) is P node (o_k), material flow k appears at the node o_k outgoing into neighborhood nodes

$$\sum_{k\in F} x_{i,j,t}^k \le c_{i,j} y_{i,j} \quad (\forall i \in N \quad \forall j \in N\ \forall t \in H). \tag{3}$$

Equation (3) ensures that the material flow quantity is equal or less than its capacity $c_{i,j}$ if the corresponding arc (i, j) is selected to a guide path

$$y_{i,j} \le \sum_{l\in\delta^+(j)|l\neq j} y_{j,l} \quad (\forall i \in N \quad \forall j \in N\ i \neq j) \tag{4}$$

$$y_{i,j} \le y_{j,j} \quad (\forall i \in N \quad \forall j \in N,\ i \neq j) \tag{5}$$

$$.\sum_{j\in\delta^+(i)} y_{i,j} \ge 1 \quad (\forall i = o_k, d_k) \tag{6}$$

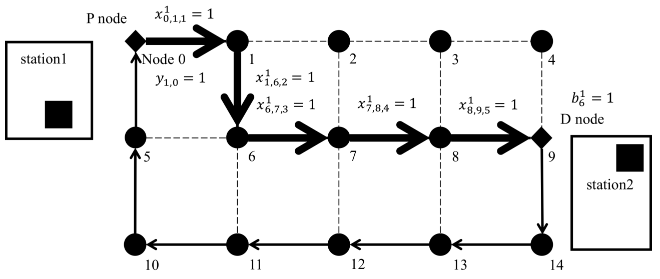


Fig. 3. Example of the solution of a guide path.

Constraints (4) indicate that if the lane from node i to node j is adopted, at least one arc from node i to node l ($l \neq i$) should be adopted to maintain the connectivity. Constraint (5) enforces that $y_{j,j} = 1$ if $y_{i,j} = 1$ such that material flow can wait at node j. Constraint (6) ensures that each P/D node is connected and both ends of each arc on the guide path are connected with other arcs

$$x_{i,j,0}^k = 0 \quad (\forall i \in N \quad \forall j \in N\ \forall k \in F) \tag{7}$$

$$b_0^k = 0 \quad (\forall k \in F) \tag{8}$$

$$x_{i,j,t}^k \ge 0, \quad y_{i,j} \in \{0, 1\} \quad (\forall i \in N \quad \forall j \in N\ \forall t \in H) \tag{9}$$

$$b_t^k \ge 0 \quad (\forall k \in F \quad \forall t \in H) \tag{10}$$

Equations (7) and (8) are the initial conditions for each flow on each arc. Equations (9) and (10) are the nonnegative constraints for each variable.

D. Example of a Solution of the Guide Path

Consider an example of the guide path shown using only unidirectional arcs in Fig. 3. The guide path contains $|N| = 15$ nodes and $|A| = 37$ arcs that consist of 22 arcs and 15 self-loop arcs that represent waiting at each node. The total time horizon is $K = 10$. There is only one commodity flow requirement $|F| = 1$, which travels from a pickup node (node 0) to delivery node (node 9) with its demanded quantity being $f_0^1 = e_9^1 = 1$. The flow travels to node 0 at time period 0. The flow travels into node 9. If the flow variable is $x_{i,j,t}^k = 1.0$, the guide path on the route is $y_{i,j} = 1$, and other variables are $y_{i,j} = 0$ to maintain the connectivity. Finally, in time period 9, $b_8^1 = 1.0$ and (2) is satisfied. In this case, if variables $l_{i,j}$, w_1 and w_2 are equal to 1, $m_{i,j} = 1 (i \neq j), m_{i,i} = 0 (i \in N)$, and the optimal value of the objective function is equal to $w_1 \sum_{(i,j)\in A} m_{i,j} y_{i,j} + w_2 \sum_{t\in H}(\sum_{(i,j)\in A} l_{i,j} x_{i,j,t}^1) = 1 \times 12 + 1 \times 5 = 17$ because the fixed costs are 12 unidirectional arcs and the flow traveling costs are 5.

IV. Solution Approach

We propose cell-based local search heuristics to solve the problem efficiently. In this section, we define the notion of the cell and k-neighborhood strategy in the proposed cell-based local search heuristics.

A. Cell-Based Neighborhood Search Algorithm

A cell-based local search is introduced to solve the problem. To explain cell-based neighborhoods easier, a grid network

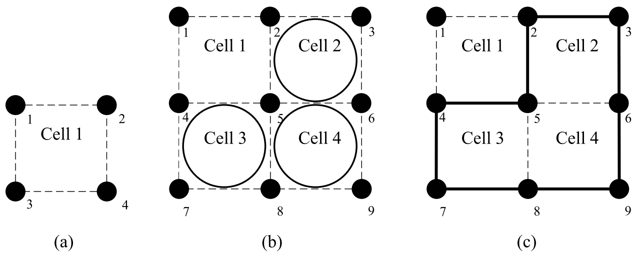


Fig. 4. Definitions of a cell, k-cell, and outside loop. (a) Example of a cell. (b) Example of a k-cell ($k = 3$). (c) Outside loop of k-cell.

is used without loss of generality. The cell has been widely used to represent block layouts for facility design and material handling networks [2]. We use the cell-based local search heuristics to generate a variety of candidates of the guide path satisfying connectivity constraints. A cell c is defined as a set of four nodes and four edges, as shown in Fig. 4(a). The guide path layout is divided into some cells without any overlaps.

The following definitions are used to develop k-neighborhood of the cell-based local search heuristics based on the use of the notion of cells.

A unit cell is defined as follows.

Definition 1: A cell $c = (e_1, e_2, e_3, e_4)$ is defined by a set of four nodes and a set of four edges, which constitutes a loop in a grid network.

A set of cells is defined.

Definition 2: A k-cell $C_k = \{c_1, c_2, \ldots, c_k\}$ is defined by k-adjacent cells in which each cell shares an edge with at least one other cell in C_k.

Definition 3: An outside loop $O_k = \{e \in A | \ \exists(c_1, c_2) \in C_k \times C_k,\ e \in c_1 \text{ and } e \notin c_2\}$ is defined by a set of edges that comprise the bound of k-cell C_k and its complement of C_k. These edges belong to only one cell in k-cell.

In Fig. 4, there are four nodes $\{1, 2, 3, 4\}$ and four edges $\{(1, 2), (2, 4), (4, 3), (3, 1)\}$, which constitutes a loop. Fig. 4(b) shows k-cell C_k in which $k = 3$ and $C_k = \{c_2, c_3, c_4\}$ and $|C_k| = 3$.

From the definition, an arbitrary cell of k-cell C_k is adjacent to the other cell; in other words, each cell shares at least one edge with another cell. For example, cell c_3 and cell c_4 are adjacent because they share an edge $(5, 8)$. Fig. 4(c) shows the outside loop of k-cell $C_k = \{c_2, c_3, c_4\}$. The outside loop is a set of edges $O_k = \{(2, 3), (3, 6), (6, 9), (9, 8), (8, 7), (7, 4), (4, 5), (5, 2)\}$ because these edges belong to only one cell of k-cell C_k, whereas edges $\{(5, 6), (5, 8)\}$ are not in the outside loop because edge $(5, 6)$ belongs to c_2 and c_4 and edge $(5, 8)$ belongs to c_3 and c_4. By using the set of cells, we have the following proposition.

Proposition 1: All selections of the guide path can be represented by choosing a set of cells and their corresponding edges.

Proof: From the assumption of strong connectivity of the graph $G = (V, A)$, for all nodesm $a, b \in V$ has a minimal loop that includes nodes a and b. A cell is minimal if it does not have a proper subset of graphs that have a cell. Since every subset of guide path selection $G' \subseteq G$ has strong connectivity, G' can be partitioned into a set of minimal loops. Therefore, all selections of the guide path can be represented by a set of cells when each minimal loop is regarded as a cell. ■

From the notion of k-cell, we develop a neighborhood search based on k-neighborhood defined as follows.

Definition 4: A key cell is selected randomly from the current set of cells that constitute a guide path. k-neighborhood is defined as the new guide path generated by joining the current guide path and k-cell C_k, including the key cell (choosing k cells from the von Neumann neighborhood of the key cell in the context of cellar automata [32]). In other hands, k-neighborhood is joined from the edge set E_G of current guide path and the edge set E_k, which is the outside loop of the k-cell, and an outside loop is represented by a set of edges, which is the outside loop of the k-cell O_k.

Proposition 2: All of the new guide paths derived by k-neighborhood have strong connectivity.

Proof: From Proposition 2, the solution derived from k-neighborhood with the set of cells G_2 has always strong connectivity because the derived set of cells has one large cycle loop. From the assumption, graph G_2 and current guide path layout graph G_1 are overlapped, and there are some common nodes and one of them is named $b \in G_1 \cup G_2$. Since the graphs G_1 and G_2 have strong connectivity, for all nodes, $a \in G_1$ and the node b has a closed loop that includes nodes a and b. Also, for all nodes, $c \in G_2$ and the node b has a closed loop that includes nodes c and b. Therefore, for all nodes, $a \in G_1$ and $c \in G_2$ have a closed loop via node b. ■

If the guide path is updated only for a single node or a single edge, the search space becomes extremely large. The convergence to a near-optimal solution becomes slower. Therefore, we update the guide path of k-cell. In the method, the set of cells C to partition the whole guide path is enumerated in advance. First, the set's neighborhood size k has to be 1. Then, select a key cell from C by Algorithm 3 to generate k-cell. The k-neighborhood is enumerated, and then, evaluate each new guide path of the k-neighborhood until the current best value of the objective function is improved. If there is no improvement from the current best solution, k is updated $k := k+1$. If the best value of the objective function is updated, k is initialized to be 1.

The overall procedure of the cell-based local search heuristics is structured as follows. The flowchart of the algorithm is shown in Fig. 5(a). In the algorithm, the dynamic multicommodity flow routing and the cell-based guide path selection are carried out sequentially, as shown in Fig. 5(b) and (c).

Cell-Based Local Search Heuristics Method:

1) *Step 1*: Generate an initial guide path G, and the initial optimal guide path is set $(f(G_{\text{best}}) = \infty)$ and set $k = 1$ and set $C_{\text{neighbor}} = \phi$. C is the current set of cells.
2) *Step 2*: Solve the dynamic multicommodity flow problem on the condition that the current guide path G is fixed.
3) *Step 3*: Eliminate the redundant path from the current guide path G by Algorithm 2.
4) *Step 4*: Calculate the objective function $f(G)$.

Algorithm 1 Cell-Based Local Search Heuristics

```
1: Initizalize $G$
2: $f(G_{best}) \leftarrow \infty$
3: $k \leftarrow 1$
4: for $itr = 1, itr < max\_iteration, itr = itr + 1$ do
5:     $IsUpdate \leftarrow$ False
6:     Solve dynamic multi-commodity flow problem on the fixed layout $G$
7:     Algorithm 2: Redundant elimination of arcs ($G$)
8:     Calculate $f(G)$
9:     if $f(G) < f(G_{best})$ then
10:        $G_{best} \leftarrow G$
11:        $IsUpdate \leftarrow$ True
12:    else
13:        $G \leftarrow G_{best}$
14:    end if
15:    if $IsUpdate ==$ True then
16:        $k \leftarrow 1$
17:        $C_{neighbor} \leftarrow \phi$
18:        $c_{key} \leftarrow$ Algorithm 3: Selection of a key cell
19:    end if
20:    if $C_{neighbor} == \phi$ then
21:        $C_{neighbor} \leftarrow k - neighborhood(c_{key})$
22:    end if1
23:    $C_k \leftarrow$ Pop($C_{neighbor}$)
24:    $G \leftarrow G \cup outsideloop(C_k)$
25: end for
```

Algorithm 2 Redundant Elimination of Arcs

```
1: $e, R, route \leftarrow \phi$
2: for all $(i, j) \in A$ do
3:     $f \leftarrow f + \sum_{t,k} x^k_{i,j,t}$
4:     if $f = 0$ then
5:         add $(i, j)$ to $e$
6:     end if
7: end for
8: for all $(i, j) \in e$ do
9:     if $\sum_{l \in \delta^-(i),(l,i) \notin e} y_{l,i} = 0$ then add $i$ to $R$
10:    end if
11:    if $\sum_{l \in \delta^+(i),(j,l) \in e} y_{j,l} \geq 2$ then add $j$ to $R$
12:    end if
13: end for
14: for all $r \in R$ do
15:    $i \leftarrow r$
16:    while $\sum_{l \in \delta^+(i),(i,l) \in e,(i,l) \notin route} y_{i,l} \geq 1$ do
17:        $route \leftarrow (i, l)$
18:        $i \leftarrow l$
19:    end while
20:    if guide path layout $G/route$ do not violate (4) - (6) then
21:        for all $(i, j) \in route$ do
22:            $y_{i,j} = y_{j,i} \leftarrow 0$
23:        end for
24:    end if
25: end for
```

5) *Step 5*: If the value of the objective function satisfies $f(G) < f(G_{\text{best}})$, then update $G = G_{\text{best}}$ and set $k = 1$ and set $C_{\text{neighbor}} = \phi$.
6) *Step 6*: Go to Step 10 if the number of iterations reached a specified number of iterations.
k-Neighborhood Search of the Cell-Based Local Search:
7) *Step 7:* Select a key cell from C by Algorithm 3.
8) *Step 8:* If $k \leq 4$ (the maximum neighborhood size is four directions for grid graph), then all possible set of k-neighbor cells C_{neighbor} are enumerated by adding k-cell C_k, including the key cell.
9) *Step 9:* If $C_{\text{neighbor}} \neq \phi$, select a k-cell C_k from C_{neighbor} and $C_{\text{neighbor}} := C_{\text{neighbor}} \backslash C_k$, and these two operations are similar to the Pop operation in stack and queue. This procedure is implemented by Pop(C_{neighbor}) at the step 23 in the pseudocode of Algorithm 1. Then, apply k-neighborhood to generate a new guide path G and go to Step 2. If $C_{\text{neighbor}} = \phi$, then $k := k + 1$ and return to Step 9.
10) *Step 10:* Subtour elimination algorithm (see Algorithm 4) is conducted if the derived tour has subtour.

The proposed method consists of two stages: solving the dynamic multicommodity flow at Step 2 and the cell-based neighborhood search of guide path at Steps 7–10. The generation of an initial solution is explained in Section IV-B. To expedite the search, the redundant path is eliminated from the current cell-based guide path by Algorithm 2. This is explained in Section IV-D. The k-neighborhood search is strengthened by using an intensive cell-based search around a key cell. The selection of the key cell is explained in Section IV-E. The subtour elimination algorithm is conducted if the derived tour has subtours. The detailed algorithm is explained in Section IV-F.

Algorithm 3 Selection of a Key Cell

```
1: $c_{max} \leftarrow 0$
2: if random[0, 1] $< \gamma$ then
3:     for all $c \in C$ do
4:         $x^+_c \leftarrow \sum_t \sum_k \sum_{((i,j)|(i,j) \notin c \wedge (j,l) \in c)} x^k_{i,j,t}$
5:         if $c_{max} < x^+_c$ then
6:             $c_{key} \leftarrow c$
7:             $c_{max} \leftarrow x^+_c$
8:         end if
9:     end for
10: else
11:    $c_{key} \leftarrow$ random[1, |C|]
12: end if
```

Fig. 6(a) illustrates an example of k-neighborhood. Fig. 6(b) shows the new guide path generated from the 2-neighborhood. The current guide path is represented by the set of cells $C = \{c_3, c_5, c_6, c_7, c_8\}$ and let c_6 be a key cell. Then, the set of neighbor cells $C_{\text{neighbor}} = \{\{c_2, c_6\}, \{c_5, c_6\}, \{c_6, c_7\}, \{c_6, c_9\}\}$. Now, let 2-cells is $\{c_6, c_7\}$. The 2-neighborhood is the union of the outside loop of the current guide path $E_G = \{(6, 7), (7, 8), (8, 3), (3, 4), (4, 9), (9, 10), (10, 15), (15, 14), (14, 13), (13, 12), (12, 11), (11, 6)\}$ and the outside loop

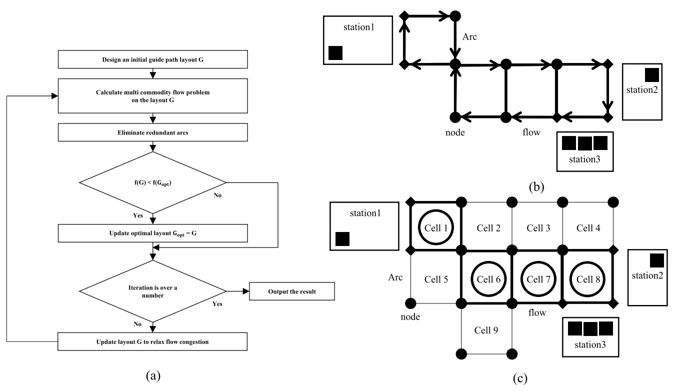


Fig. 5. (a) Flowchart of the proposed algorithm. (b) Dynamic multicommodity flow problem. (c) Guide path selection problem as subproblem.

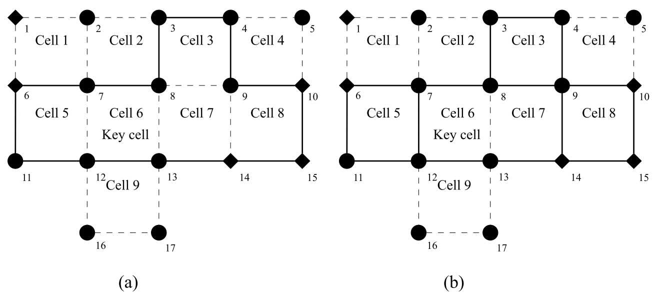


Fig. 6. (a) Guide path with cell and key cell. (b) Example of 2-neighborhood.

of the newly selected 2-cell $\{c_6, c_7\}$, including the key c_6: $E_2 = \{(7, 8), (8, 9), (9, 14), (14, 13), (13, 12), (12, 7)\}$. A newly generated guide path $E_{G_{new}}$ is $\{(6, 7), (7, 8), (8, 9), (9, 14), (12, 7), (8, 3), (3, 4), (4, 9), (9, 10), (10, 15), (15, 14), (14, 13), (13, 12), (12, 11), (11, 6)\}$.

The efficiency of the search can be improved when the cell is used to search the guide path because a variety of guide path can be generated easily without violating connectivity and loop constraints. The optimal solution can also be represented by a set of cells.

B. Generation of an Initial Solution

An initial guide path is generated such that all P/D nodes are connected to each other and the guide path is strongly connected. To obtain an initial solution, we introduce the concept of cells. First, we search for the set of cells that cover all P/D nodes and are adjacent to each other by the depth-first search. Then, whenever candidates are found, we add the outline of the cell sets and the number of cells into guide path layout candidates. Finally, we define the optimal candidate, which has minimal cells as the initial guide path.

C. Solving the Dynamic Multicommodity Flow Problem

The dynamic multicommodity flow problem is based on the formulation in Section III-C. Removing some constraints about the connectivity of the guide path and fixing the current decision variable $y^*_{i,j}$ on the guide path layout, the problem is a linear programming problem that can be solved by the simplex algorithm. The formulation is provided as follows:

$$\min\ w_2 \sum_{t\in H} \sum_{k\in F} \sum_{(i,j)\in A} l_{i,j} x^k_{i,j,t} \tag{11}$$

$$\text{s.t.}\quad (2), (7), (8), (9), (10)$$

$$\sum_k x^k_{i,j,t} \le c_{i,j} y^*_{i,j}\ (\forall i \in N\ \ \forall j \in N\ \forall t \in H). \tag{12}$$

D. Redundant Elimination of Arcs

In the k-neighborhood, the current guide path is updated by adding some cells and its outline edges in the current guide path. However, in some cases, redundant edges may be generated during the k-neighborhood search. Therefore, in this step, multiple redundant arcs are removed during the iteration of the cell-based heuristics. From the results of the dynamic multicommodity flow problem, we define the criteria to eliminate these arcs. The arcs are removable when there is no flow on them in the solution of a dynamic multicommodity flow problem. To eliminate such arcs, we have to consider the constraints that all arcs should be connected. In the following procedure, we provide the redundant arc elimination algorithm to maintain the connectivity as follows. The overall procedure is described as follows, and the detailed algorithm is shown in Algorithm 2.

Redundant Elimination of Arcs Method:

1) *Step 1:* From the results of the problem, enumerate arcs that have no flow, and then, they are added to the set of redundant arcs e.
2) *Step 2:* The nodes that have no arc into the set of redundant arcs e are added into the set of nodes R. Then, add the nodes that have two or more than two arcs from the set of redundant arcs.
3) *Step 3:* From the set of nodes, enumerate all removable routes to the destination node from which all arcs of the set of redundant arcs leave.
4) *Step 4:* For each path, check whether the connectivity constraints are violated or not and the route has no P/D nodes, and then, remove arcs on the route from the current guide path.

We explain an illustrative example in Fig. 7, where $\sum_t x^k_{i,j,t}$ flow on each arc is depicted.

At the first step (a) in Fig. 7, we can see that five arcs have no flow in the set of selected arcs from the results. Then, in the second step (b) in Fig. 7, from the selected arcs, three root nodes are chosen by using Algorithm 2. In step (c) in Fig. 7, five routes from the selected root nodes are found. Finally, at the final step (d) in Fig. 7, each route is tested, and routes 3 and 5 are eliminated. After the elimination of routes 1, 2, and 4, violate the connectivity constraints. Therefore, routes 1, 2, and 4 are not eliminated in this example.

E. Intensification of Search for Cell-Based Heuristics

If the random selection of a key cell in the k-neighborhood is applied, it sometimes requires a lot of computing time to derive a near-optimal solution. The random selection has

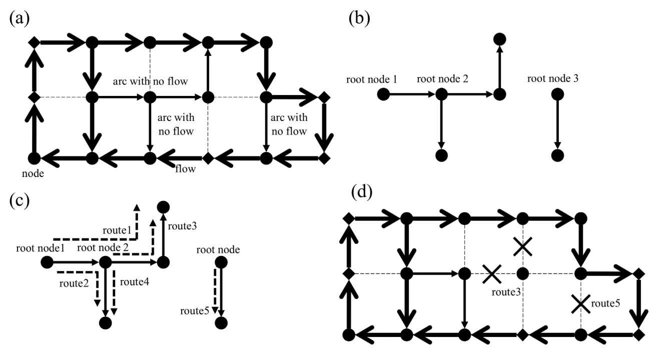


Fig. 7. Example of redundant arc elimination method.

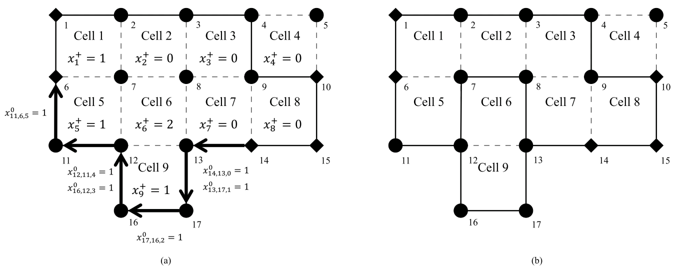


Fig. 8. Detection of the concentrated cells and the generation of new guide path. (a) Guide path with cell and key cell. (b) Example of the 2-neighborhood.

no use of the knowledge from the results of the evaluation from the dynamic multicommodity flow problem. Therefore, we introduce the concept of flow-concentrated cells in which the quantity of the flow is concentrated on the specific nodes for all time periods and all commodities to relax the concentration of the flow using the knowledge from the results. The most concentrated cell is defined that the total quantity of flow entering the cell for all periods is the highest. To compute the total quantity of entering the flow, we define the total entering flow x_c^+ into the cell c and x_c^+ is defined as follows:

$$x_c^+ = \sum_t \sum_k \sum_{((i,j)|(i,j)\notin c \wedge (j,l)\in c)} x_{i,j,t}^k \ (\forall c \in C). \tag{13}$$

The computation is carried out before selecting a key cell during the search. The key cell is determined with the most concentrated cell by probability γ. Otherwise, the key cell is selected randomly.

Fig. 8 illustrates an example of the current guide path represented by the set of cells $\{c_1, c_2, c_3, c_5, c_6, c_7, c_8, c_9\}$. The guide path of our example containing $|C| = 9$ cells' flow on each arc $x_{i,j,t}^k$ is depicted in Fig. 8(a). If (13) is computed for each cell $c \in C$, we find the most concentrated cell c_6 in which the sum of the entering flow is $x_6^+ = x_{16,12,3}^0 + x_{14,13,0}^0 = 2$ and then, let c_6 be the key cell $c_{key} = c_6$. Then, the set of neighbor cells $C_{\text{neighbor}} = \{\{c_2, c_6\}, \{c_5, c_6\}, \{c_6, c_7\}, \{c_6, c_9\}\}$. Now, let 2-cell is $\{c_6, c_9\}$ and $C_{\text{neighbor}} = C_{\text{neighbor}} \backslash \{c_6, c_9\}$. After selecting the 2-cell $\{c_6, c_9\}$, adopt arcs on the outside loop of a large cell of the set shown in Fig. 8(b). In many cases, the procedure may generate a shortcut path for each flow to improve the value of the objective function.

F. Subtour Elimination Method

There is a possibility that the guide path layout has multiple disconnected loops violating subtour elimination constraints, i.e., all P/D nodes are not connected to each other. It is necessary to eliminate these subtours because all transportation, which has a variety of destination and origin points, must be achieved. To solve this problem, we consider two methods. In the first method, if the guide path lane is restricted into the unidirectional arc, we can adopt the following subtour elimination constraints into the formulation in which we select a node $v \in \{o_k\} \cup \{d_k\}$ that is the P/D node because P/D nodes must be all connected to the guide path and a fixed number M is the large number for big-M

$$u_v = 0 \tag{14}$$

$$u_i + M(1 - y_{i,j}) \geq u_j \ (i \neq v) \ (\forall (i,j) \in A). \tag{15}$$

In the second method, when the guide path type is not restricted, we cannot embed these constraints into the formulation. Therefore, we propose the following iterative algorithm to maintain the connectivity of the guide path in our mathematical model.

Algorithm 4 [Subtour Elimination Algorithm]

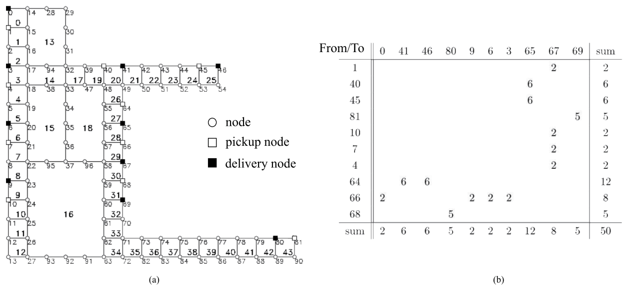
```
Step 0  Set the solution of $y_{i,j}$ derived from the cell-based
        local search heuristics.
Step 1  Search the nodes that are connected to the arcs from
        each pickup and delivery point, then provide label $l_i$
        searched nodes with value for each independent loop.
Step 2  Complete the algorithm if there is only one type of
        label value.
Step 3  Enumerate the set of nodes $V_{l_i}$ for each independent
        loop with label $l_i$.
Step 4  Add the following subtour elimination constraints for
        $V_{l_i}$ in the original problem and solve the problem.
        Then return to Step 0.
```

$$\sum_{i \in U_{l_i}} \sum_{j \in N \backslash U_{l_i}} y_{i,j} \geq 1$$
$$(\forall U_{l_i} \subseteq V_{l_i},\ U_{l_i} \neq \phi,\ N \backslash U_{l_i} \neq \phi\ \forall l_i). \tag{16}$$

Steps 2 and 4 of the algorithm run in $O(|A|)$ because there are only $|A|$ arcs. Equation (16) ensures that there is at least one arc from each independent loop to another loop to eliminate subtour. In the problem definition, all P/D nodes must be in a loop. In the worst case, the number of the additional constraints is $O(2^{(|P|+|D|)})$ because the maximum number of the labels is $|P|+|D|$. Therefore, the computational complexity of subtour elimination algorithm is $O(2^{|P|+|D|})$ which is exponential to the number of P/D nodes. However, in practical implementation, the algorithm requires much fewer iterations, e.g. a few iterations to eliminate subtours from several preliminary experiments.

V. Computational Study

Computational experiments are conducted to confirm the effectiveness of the proposed method in this section. The results of the guide path derived from the proposed method are evaluated by the transport simulations for a real case study.



From/To	0	41	46	80	9	6	3	65	67	69	sum
1									2		2
40								6			6
45								6			6
81										5	5
10									2		2
7									2		2
4									2		2
64		6	6								12
66	2				2	2	2				8
68				5							5
sum	2	6	6	5	2	2	2	12	8	5	50

Fig. 9. (a) Candidates of the guide path. (b) From/To chart.

A. Problem Instances

We apply the proposed method to a real-world instance of a transport company. A candidate for the design of the guide path layout is shown in Fig. 9(a). It contains $|N| = 101$, $|A| = 371$ arcs, and $|P| = |D| = 10$ P/D nodes illustrated by the white and black squares, and the guide path is divided into $|C| = 44$ cells. The weighting factors of the flow traveling costs and the fixed costs, which are represented by w_1 and w_2, respectively, in the objective function were set to $w_1 = 1$ and $w_2 = 100$ because the weight of the flow traveling costs is much more important than that for the fixed costs in designing a guide path in general. The total planning horizon is $K = 60$ periods. The transport demands are also shown in Fig. 9(b). Each row is the pickup node (from), each column is the delivery node (to), and the value is the quantity of each demand. In the proposed method, the maximum number of iterations is set to 300. The program used for the computation was coded in Microsoft Visual C++ 2017 Edition. Tests were carried out on an Intel Core i7-2700K 3.50 GHz and 8.0-GB memory. A general-purpose solver IBM ILOG CPLEX 12.6 was used for solving the mixed-integer and linear programming problems.

B. Comparison of the Detailed and the Proposed Models

We examine the difference in the performance between the detailed model that was developed in [1] and the dynamic multicommodity flow model proposed in this paper. These two models are solved by IBM ILOG CPLEX 12.6, where $w_1 = 100$ and $w_2 = 1$. The main difference between the proposed model and the detailed model is summarized as follows.

1) The conflicts and interference between vehicles are represented by the constraints of flow restriction instead of collision avoidance constraints between vehicles.
2) The P/D is expressed by the appearance and disappearance of flows instead of P/D operations at these locations.
3) The traveling of empty vehicles in the proposed model is omitted.

The total computation time and the total costs are compared by solving the detailed and proposed models under the same transport tasks. Table II summarizes the comparison of those two models. The total computation time to derive a solution is significantly reduced by the proposed method. The detailed model cannot derive a solution when the number of demands is greater than 10. There are differences in the flow traveling costs in these models. This is caused by the difference since the empty flow of vehicles is neglected in the proposed model. However, the results indicate that the derived guided path has similar layouts that have the same fixed costs. From these results, the proposed model is applicable to the instances with larger demands with good accuracy.

TABLE II
COMPUTATIONAL RESULTS BETWEEN THE DETAILED AND THE PROPOSED MODELS

	Detailed model				Proposed model			
demands	time [s]	obj	flow travel	fixed	time [s]	obj	flow travel	fixed
7	394	7369	73	69	26	6982	82	69
7	478	8075	75	80	32	5740	140	56
10	2311	22173	221	73	39	10272	102	72
13	–	–	–	–	83	49367	493	67

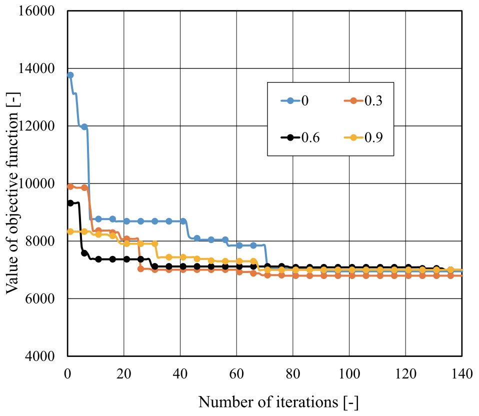


Fig. 10. Transition of the value of the objective function.

C. Effects of the Probability to Update Concentrated Cells

To evaluate the performance of the proposed algorithm, tests were conducted by changing the probability γ of selecting the concentrated cells when the guide path was updated in the cell-based neighborhood search procedure. It was the main part of the heuristics and the change may signify the strength of the k-neighborhood search.

Fig. 10 shows the computational results when the parameter γ is changed. The value of the objective function decreases as the probability γ rises, but when the probability γ is near to the value of 1.0, the value of the objective function increases because the solution converges quickly to a bad local optimum solution. The curve of the objective function has a minimum point with respect to the probability of γ. Also, the computational time increases when γ is increased. Therefore, it is noted that the parameter γ should be set around 0.5 according to preliminary experiments. As we can see from Fig. 11 that the convergence of the proposed algorithm is much faster when the most concentrated cells are selected as the key cell than the case when a cell is selected randomly.

D. Comparison of General-Purpose Solver and the Cell-Based Local Search Heuristics

To evaluate the performance of the proposed method, we compared the performance of the proposed method with that of a general-purpose solver (CPLEX) under various conditions. Test instances were generated changing the total quantity of demands from 2 to 101 when the weights of the flow traveling costs and the fixed costs are given.

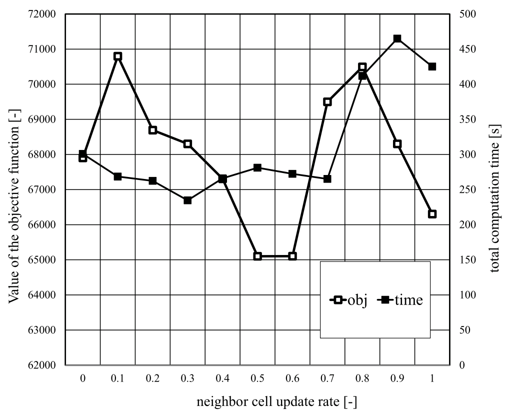


Fig. 11. Effects of the probability γ of updating concentrated cells to the solution.

TABLE III
COMPARISON OF THE PROPOSED METHOD WITH THE GENERAL PURPOSE SOLVER (CPLEX)

	General purpose solver (CPLEX)				Proposed method			
demand	time [s]	obj	flow travel cost	fixed cost	time [s]	obj	flow travel cost	fixed cost
2	3.82	2684	26	84	73.48	2791	27	91
11	24.14	13381	133	81	72.99	13482	134	82
16	15.39	19383	193	83	92.12	19788	197	88
28	75.81	26189	261	89	269.74	27709	276	109
50	4400.98	64889	648	89	281.08	65101	650	101
78	10801.9	132378	1323	78	1122.68	141599	1415	99
52	-	-	-	-	997.04	83000	829	100
88	-	-	-	-	2423.72	191011	1909	111
101	-	-	-	-	4702.2	167107	1670	107

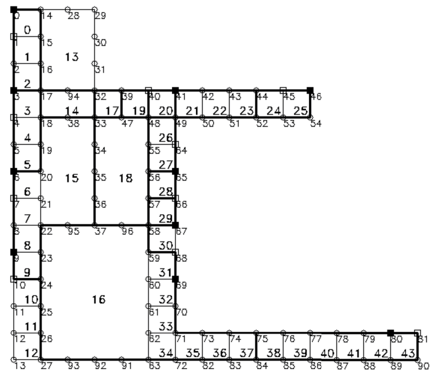


Fig. 12. Result of the guide path design derived from the proposed method.

The computational results of the performance evaluation of the proposed method are also shown in Table III. Fig. 12 shows a result of the obtained guide path by the proposed method. In Fig. 12, the candidates of the arc and the created guide path are depicted by the thin and bold lines, respectively. Each node is drawn as a circle, and the P/D nodes are depicted by the squares. The node number is written in the lower right of the node, and the cell number is written in the center of the cell. We can see from Fig. 12 that all constraints are satisfied. For example, all P/D nodes are connected by some arcs.

In Table III, demand is the total quantity of demand flow for all P/D nodes. The quantity of initial flow is at time 0. The total traveling cost is the total flow travel distance multiplied by the flow quantity. The results show that the proposed method was able to derive a solution within approximately 73 s in the shortest demand case. The quality of the solutions derived from the proposed method is quite similar to the optimal solution derived by CPLEX. However, the total computation time of the proposed method is significantly reduced when the number of demands is larger than 50. For fewer demand cases, the proposed method requires larger computational efforts than does CPLEX. This is because the proposed method uses the density of the traveling flows. Therefore, in smaller demand cases, a flow rarely encounters other flows, so it is not efficient for the proposed method to update the guide path because the proposed method utilizes the information of flow congestions. However, for larger demand cases, the proposed method can derive a solution much faster than CPLEX. The gap between the objective values of the two methods is within 7%. CPLEX cannot derive even a feasible solution when the demand is greater than 78. The performance of the proposed method is not better than CPLEX for smaller cases. For larger cases, the performance of the proposed method is compared with other methods in Table V. If the value of travel cost is higher, the gap becomes much higher as in the case when the demand is equal to 78 because we set $w_1 = 1$ and $w_2 = 100$. If the ratio of the layout and transport costs in the objective function is changed, the total computation time of the general-purpose solver significantly increases as the ratio of the weighting factor to the fixed cost increases; however, the total computation time of the proposed method is less influenced by the ratio of the weighting factors because the proposed method does not need to consider the tradeoff between layout and flow traveling costs during the computation of the dynamic multicommodity flow problem.

For larger demand cases with 78 demands, CPLEX cannot derive even a feasible solution because the number of decision variables for dynamic multicommodity flow is significantly increased. The proposed method can derive a feasible solution even for the cases with 100 demands. From the results, the effectiveness of our proposed method is confirmed.

E. Comparison With Static and Dynamic Multicommodity Flow Models

The performance of the dynamic multicommodity flow model is evaluated by comparing it with that of the static flow model. The static multicommodity flow model is constructed based on the formulation in Section III-C, and the time index is removed from the original formulation. The decision variable of $x_{i,j}^k$ is used without $x_{i,j,t}^k$. Feasibility on the same condition is assured in the problem definition. The detailed formulation is described as follows:

$$\min \left(w_1 \sum_{(i,j)\in A} m_{i,j} y_{i,j} + w_2 \sum_{k\in F} \sum_{(i,j)\in A} l_{i,j} x_{i,j}^k \right) \quad (17)$$

$$\text{s.t.} \quad (4), (5), (6)$$

$$\sum_{j\in\delta^-(i)} x_{j,i}^k - \sum_{m\in\delta^+(i)} x_{i,m}^k = \begin{cases} \sum_{t\in H} e_t^k \ (i = d_k) \\ \sum_{t\in H} -f_t^k \ (i = o_k) \\ 0 \ (i \neq d_k, o_k) \end{cases} \quad (18)$$

$$\sum_k x_{i,j}^k \le c_{i,j} y_{i,j} \ (\forall i \in N \quad \forall j \in N) \quad (19)$$

$$x_{i,j}^k \ge 0, \quad y_{i,j} \in \{0,1\} \ (\forall i \in N \quad \forall j \in N \ \forall k \in F). \quad (20)$$

TABLE IV
COMPARISON OF THE PROPOSED METHOD WITH STATIC FLOW MODEL

	General purpose solver			Static flow model			Proposed heuristics		
demand	obj	flow travel cost	fixed cost	obj	flow travel cost	fixed cost	obj	flow travel cost	fixed cost
2	2684	26	84	2784	27	84	2791	27	91
11	13381	133	81	13491	134	91	13482	134	82
16	19383	193	83	20378	203	78	19788	197	88
28	26189	261	89	26805	267	105	27709	276	109
50	64889	648	89	67290	672	90	65601	650	101
78	132378	1323	78	141589	1415	89	141599	1415	99
52	-	-	-	93986	939	86	83000	829	100
88	-	-	-	209487	2094	87	190111	1909	111
101	-	-	-	169299	1692	99	167107	1670	107
average	-	-	-	82789.9	827.0	89.9	79020.9	789.7	98.7

The computational results for static and dynamic models are shown in Table IV. To compare the performance of these models, the multicommodity flow formulation in Section III-C is solved with the fixed guide path generated from each model. From the results, the value of the fixed costs obtained from the static model is larger than that of the dynamic model. This is because the static model just derives the shortest path between P and D nodes without considering the flow congestions. The number of lanes derived from the static model is less than that of the dynamic model. Also, the number of congestions is sufficiently small in the smaller demand cases. Then, the fixed costs derived from the proposed method are almost the same that derived from the static model. The average of total costs for the dynamic model is significantly less than that of the static model even the fixed costs of the dynamic model are larger than the static model. The guide path design by utilizing the dynamic multicommodity flow model can reduce the transport delays caused by congestions.

F. Comparison With Recent Algorithms on Fixed-Charge Capacitated Multicommodity Network Design Problem

To confirm the efficiency of the proposed method, we compared the performance of the proposed method with that of the following two recent algorithms for the fixed-charge capacitated multicommodity network design problem. The one is ACO-based neighborhood search [34], and the other is SS/LPH [18]. Each algorithm is explained as follows.

G. ACO-Based Neighborhood Search

A new solution is generated by constructing new guide paths for the demands with continuous volumes delivered on the closed arc by using ACO. A submixed-integer programming model is used to generate a neighborhood. Each ant can reach any of multiple sink nodes corresponding to the source when a neighborhood network search is conducted.

H. SS/LPH Method

SS is an iterative scheme that consists of solving a linear approximation of the original formulation at each iteration. The costs of each linear approximation are adjusted to reflect the exact costs incurred by the solution at the previous iteration. The iterations proceed until two successive solutions are identical. At this point, the linear approximation costs correspond to the true objective function given by the sum of the transportation costs and the fixed costs. To improve the performance, a Lagrangian bounding procedure is performed concurrently with SS procedure. The procedure using long-term memory [18] cannot be adopted because the memory size, which is proportional to network size, becomes too huge.

TABLE V
COMPARISON OF THE PROPOSED METHOD WITH RECENT ALGORITHMS WITH ACO AND SS/LPH WITH THE SAME NUMBER OF ITERATIONS

		Proposed heuristics		ACO-based search		SS/LPH method	
$\|N\|$	demand	obj	time [s]	obj	time* [s]	obj	time* [s]
15	1	415	13.4	413	49.4	413	20.5
15	2	418	12.7	413	26.3	413	62.3
15	3	1017	13.5	915	95.2	917	32.1
15	5	1518	14.6	1519	202.0	1520	43.2
101	2	2791	73.5	2685	1469.9	2707	75.2
101	11	13482	72.9	13759	523.4	13481	221.7
101	16	19788	92.1	19782	10036.6	19684	338.5
101	28	27709	269.7	-	-	-	-

(time* is the computation time when the best solution is obtained)

TABLE VI
COMPARISON OF THE PROPOSED METHOD WITH RECENT ALGORITHMS WITH ACO AND SS/LPH WITH THE SAME COMPUTATION TIME

		Proposed heuristics		ACO-based search		SS/LPH method	
$\|N\|$	demand	obj	time [s]	obj	time [s]	obj	time [s]
15	1	415	13.4	414	13.5	415	14.4
15	2	418	12.7	415	14.0	416	14.3
15	3	1017	13.5	1039	14.5	1105	13.5
15	5	1518	14.6	1619	1576	1560	14.3
101	2	2791	73.5	2766	74.4	2707	75.2
101	11	13482	72.9	35715	74.2	17262	73.1
101	16	19788	92.1	38149	94.2	31149	93.2
101	28	27709	269.7	-	-	-	-

These methods can be applied only to the static multicommodity network design problem. To solve the dynamic multicommodity network design problem by these two methods, the original network is modified into the time–space network by the following conversion method.

Conversion Method for Original Network $G = (N, A)$ Into Time–Space Network $G_T = (N_T, A_T)$:

1) *Step 1:* Generate $|T|$ networks $G_T = (N_T, A_T)(T = 1, \dots, K)$ from the original network $G = (N, A)$.
2) *Step 2:* Connect the arcs with time t into the nodes with time $t + 1$.
3) *Step 3:* All the nodes on G_T that corresponds to the sink node on G are defined to the sink nodes.
4) *Step 4:* Solve the static network design problem for G_T.
5) *Step 5:* Convert the solution on G_T into G by the following equation:

$$\sum_t y^*_{(i,t),(j,t+1)} \geq 0 \rightarrow y^*_{i,j} = 1. \tag{21}$$

6) *Step 6:* If the network of the solution y^* has subtour, subtour elimination method in Section IV-F is executed to remove all subtours.

The comparison of the performance of the proposed method, ACO, and SS/LPH for small-/large-scale instances are shown in Tables V and VI. Table V shows the comparison when the number of iterations is set to 300 such that the values of the objective function for ACO and SS/LPH are sufficiently converged; time* in Table V is the computation time when the best solution is obtained.

For small-scale instances, the performance of the conventional methods is slightly better than the proposed method; however, those instances are impractical because the number of demands is less than 4. The performance of the proposed method is much better than conventional methods for large-scale instances. If the number of iterations is increased, the values of the objective function are the same for all methods, but the computation times for ACO and SS/LPH are much increased.

Table VI shows the comparison when the computation time for the proposed method and those of ACO and SS/LPH is almost the same. If the computation time for the proposed method and those of ACO and SS/LPH is the same, the values of the objective function for ACO and SS/LPH are slightly increased for small-scale instances, and they are much increased, especially for large-scale instances when $|N| = 101$. The proposed method was able to derive the same level of solutions with less computational effort than the conventional methods even for small-scale instances. For large-scale instance and the number of demand is equal or larger than 28, ACO and SS/LPH methods could not find a feasible solution, and the computation of ACO and SS/LPH are stopped due to memory over because the construction of time–space network requires larger computational efforts. In particular, ACO-based neighborhood search requires larger computational efforts. That is because the step of solving a mixed-integer program requires a lot of computational costs in the algorithm. The computational time of the mixed-integer program is proportional to the network size. The gap between the values of the objective function for those three methods is within 4%.

Our objective is to develop the proposed algorithm for practical cases $|N| = 101$ when the number of demand is around 100 or more. The comparable cases that can be shown in Table V are so limited. Most of the practical cases could not be solved by ACO and SS/LPH. The difference between the performance of the proposed method and ACO and SS/LPH is much smaller even for small-scale instances. From the discussion, the effectiveness of the proposed algorithm over other algorithms, such as CPLEX, ACO, and SS/LPH, is confirmed for practical cases.

I. Transport Simulation With Automod Software

To confirm the efficiency of the proposed method, we conducted a transport simulation for the derived guide path design. We used a commercial simulation software called Automod developed by Applied Materials, Inc., for the simulation of transport simulations considering acceleration, battery charging, and P/D times, among other physical parameters. The software was able to simulate a situation that was close to a real-world situation. The settings of the simulation software are listed in the following.

1) Each vehicle transports loads via the routing derived from Dijkstra's algorithm.
2) Each vehicle transport avoids conflict and deadlocks.
3) Physical parameters, such as acceleration, deceleration, picking up, dropping off, and rotating times, are considered.
4) Picking up, dropping off, and rotating times are considered.
5) Transportation requests are given by random numbers on each pair of picking up and dropping off points following a normal distribution.
6) Vehicle settings regarding batteries are ignored.

TABLE VII
SIMULATION RESULTS OF AUTOMOD FOR THE GUIDE PATH DESIGN DERIVED FROM THE PROPOSED METHOD

number of	load waiting time			lead time		
vehicles	mean	max	min	mean	max	min
1	593.5978	7237.6178	42.7059	651.0178	7279.8010	90.9699
2	128.0382	687.4610	11.6186	185.9726	742.2187	55.5382
3	63.6144	239.6156	10.0251	122.9115	310.4616	47.1747
4	56.5305	229.3709	10.0000	117.3527	299.3698	47.4932
5	48.5836	190.2717	10.0841	108.2366	258.7525	47.1747

By using the software, we evaluate the performance of the derived guide path with two criteria. The one is the load waiting time that is the time from loading operation to pickup operation. The second is the load time that means the same as the load-travel time including drop-off time. We investigate the effects of changing the fleet size of vehicles to estimate the optimal number of vehicles.

In the transport simulation, each P/D location was the same as those of the test cases. Each vehicle required 10 s for pickup and drop-off times. The simulation time was set to 3 h. The number of pickup loads per hour was fixed, and the destination of the load was random. Each vehicle was able to turn at 45Â°/s. The acceleration speed of each vehicle was 0.4 m/s^2, and the deceleration was 0.5 m/s^2. The results of the transport simulation on the two layouts are shown in Table VII. We can confirm the feasibility of the guide path derived by the proposed method for satisfying demands, even for real case studies in simulations. Each load is correctly transported on the guide path. If the number of vehicles is 1, each time on the right-hand side of the table is very large, signifying that the transportation could not meet the demands. If the number of vehicles is 3 or more, the improvement in transportation time is less. Therefore, we can estimate the optimal vehicle fleet size as being between 2 and 3. The simulations confirm the feasibility of the derived guide path.

VI. CONCLUSION

This article proposed the cell-based heuristics for a guide path design problem of AGV systems. A mathematical model employing a dynamic multicommodity flow problem considering the dynamic traveling conditions, such as congestions and conflicts, of vehicles was developed. To reduce computational efforts, we have provided a two-stage method by which a dynamic multicommodity flow problem and a cell-based guide path design problem were solved by a k-neighborhood search. The results of the computation show that the computation time of our proposed method, in cases when the demand, is larger is significantly shorter than that of a general solver. The duality gap of the proposed method in the worst case is 7%. The effectiveness of the dynamic model is confirmed by comparing the performance with the static model. The transport simulation has shown that transport is correctly done considering physical parameters. We can estimate the optimal fleet size of the AGVs. The main feature of the heuristics is the use of cells to create the guide path and the detection of the concentrated cells. The heuristics offer the advantage of lower computational effort. The proposed method can be easily extended to unidirectional lanes. The proposed method is compared with recent algorithms on the capacitated fixed-charge network design problem. From the results, our proposed method can be applied to solve large-scale capacitated fixed-charge network design problems with connectivity constraints. The future direction of this work is to develop a more efficient procedure for general dynamic multicommodity network design problems.

References

[1] S. Akiyama, T. Nishi, T. Higashi, K. Kumagai, and M. Hashizume, "A multi-commodity flow model for guide path layout design of AGV systems," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2017, pp. 1251–1255.

[2] A. Asef-Vaziri and G. Laporte, "Loop based facility planning and material handling," *Eur. J. Oper. Res.*, vol. 164, no. 1, pp. 1–11, 2005.

[3] A. Asef-Vaziri and M. Goetschalckx, "Dual track and segmented single track bidirectional loop guidepath layout for AGV systems," *Eur. J. Oper. Res.*, vol. 186, no. 3, pp. 972–989, 2008.

[4] A. Asef-Vaziri and M. Kazemi, "Covering and connectivity constraints in loop-based formulation of material flow network design in facility layout," *Eur. J. Oper. Res.*, vol. 264, no. 3, pp. 1033–1044, 2018.

[5] A. Asef-Vaziri, M. Kazemi, K. Eshghi, and M. Lahmar, "An ant colony system for enhanced loop-based aisle-network design," *Eur. J. Oper. Res.*, vol. 207, no. 1, pp. 110–120, 2010.

[6] A. Asef-Vaziri, G. Laporte, and R. Ortiz, "Exact and heuristic procedures for the material handling circular flow path design problem," *Eur. J. Oper. Res.*, vol. 176, no. 2, pp. 707–726, 2007.

[7] N. Boland, M. Hewitt, L. Marshall, and M. Savelsbergh, "The continuous-time service network design problem," *Oper. Res.*, vol. 65, no. 5, pp. 1303–1321, 2017.

[8] Y. A. Bozer and M. M. Srinivasan, "Tandem AGV systems: A partitioning algorithm and performance comparison with conventional AGV systems," *Eur. J. Oper. Res.*, vol. 63, no. 2, pp. 173–191, 1992.

[9] P. Caricato, G. Ghiani, A. Grieco, and R. Musmanno, "Improved formulation, branch-and-cut and tabu search heuristic for single loop material flow system design," *Eur. J. Oper. Res.*, vol. 178, no. 1, pp. 85–91, 2007.

[10] T. G. Crainic, B. Gendron, and G. Hernu, "A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design," *J. Heuristics*, vol. 10, no. 5, pp. 525–545, 2004.

[11] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, vol. 5, no. 4, pp. 691–703, 1976.

[12] X. Fan, Q. He, and Y. Zhang, "Zone design of tandem loop AGVs path with hybrid algorithm," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 869–874, 2015.

[13] R. J. Gaskins, J. M. A. Tanchoco, and F. Taghaboni, "Virtual flow paths for free-ranging automated guided vehicle systems," *Int. J. Prod. Res.*, vol. 27, no. 1, pp. 91–100, 1989.

[14] J. L. Kennington and C. D. Nicholson, "The uncapacitated time-space fixed-charge network flow problem: An empirical investigation of procedures for arc capacity assignment," *INFORMS J. Comput.*, vol. 22, no. 2, pp. 326–337, 2010.

[15] M. C. De Guzman, N. Prabhu, and J. M. A. Tanchoco, "Complexity of the AGV shortest path and single-loop guide path layout problems," *Int. J. Prod. Res.*, vol. 35, no. 8, pp. 2083–2092, 1997.

[16] K. H. Kim and J. M. A. Tanchoco, "Economical design of material flow paths," *Int. J. Prod. Res.*, vol. 31, no. 6, pp. 1387–1407, 1993.

[17] T. Le-Anh and M. B. M. De Koster, "A review of design and control of automated guided vehicle systems," *Eur. J. Oper. Res.*, vol. 171, no. 1, pp. 1–23, 2006.

[18] J. K. Lim, J. M. Lim, K. Yoshimoto, K. H. Kim, and T. Takahashi, "Designing guide-path networks for automated guided vehicle system by using the Q-learning technique," *Comput. Ind. Eng.*, vol. 44, no. 1, pp. 1–17, 2003.

[19] W. L. Maxwell and J. A. Muckstadt, "Design of automatic guided vehicle systems," *IIE Trans.*, vol. 14, no. 2, pp. 114–124, 1982.

[20] T. Nishi, M. Ando, and M. Konishi, "Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition and coordination technique," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1191–1200, Dec. 2005.

[21] T. Nishi, M. Ando, and M. Konishi, "Experimental studies on a local rescheduling procedure for dynamic routing of autonomous decentralized AGV systems," *Robot. Comput.-Integr. Manuf.*, vol. 22, no. 2, pp. 154–165, 2006.

[22] T. Nishi, M. Ando, M. Konishi, and J. Imai, "A distributed route planning method for multiple mobile robots using Lagrangian decomposition technique," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 3, Sep. 2003, pp. 3855–3861.

[23] T. Nishi and R. Maeno, "Petri net decomposition approach to optimization of route planning problems for AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 523–537, Jul. 2010.

[24] T. Nishi, S. Morinaka, and M. Konishi, "A distributed routing method for AGVs under motion delay disturbance," *Robot. Comput.-Integr. Manuf.*, vol. 23, no. 5, pp. 517–532, 2007.

[25] T. Nishi, K. Shimatani, and M. Inuiguchi, "Decomposition of Petri nets and Lagrangian relaxation for solving routing problems for AGVs," *Eur. J. Oper. Res.*, vol. 47, no. 14, pp. 3957–3977, 2009.

[26] T. Nishi and Y. Tanaka, "Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 5, pp. 1230–1243, Sep. 2012.

[27] J. Rubaszewski, A. Yalaoui, L. Amodeo, and S. Fuchs, "Efficient genetic algorithm for unidirectional flow path design," in *Proc. 14th IFAC Symp. Inf. Control Problems Manuf.*, Bucharest, Balkans, vol. 45, no. 6, 2012, pp. 883–888.

[28] M. S. Sedehi and R. Z. Farahani, "An integrated approach to determine the block layout, AGV flow path and the location of pick-up/delivery points in single-loop systems," *Int. J. Prod. Res.*, vol. 47, no. 11, pp. 3041–3061, 2009.

[29] Y. Seo and P. J. Egbelu, "Flexible guidepath design for automated guided vehicle systems," *Int. J. Prod. Res.*, vol. 33, no. 4, pp. 1135–1156, 1995.

[30] D. Sinriech and J. M. A. Tanchoco, "Impact of empty vehicle flow on performance of single-loop AGV systems," *Int. J. Prod. Res.*, vol. 30, no. 10, pp. 2237–2252, 1992.

[31] Y. Tanaka, T. Nishi, and M. Inuiguchi, "Dynamic optimization of simultaneous dispatching and conflict-free routing for automated guided vehicles—Petri net decomposition approach," *J. Adv. Mech. Des., Syst., Manuf.*, vol. 4, no. 3, pp. 701–715, 2010.

[32] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling*. Cambridge, MA, USA: MIT Press, 1987.

[33] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *Eur. J. Oper. Res.*, vol. 170, no. 3, pp. 677–709, 2006.

[34] M. Yaghini and A. Foroughi, "ACO-based neighborhoods for fixed-charge capacitated multi-commodity network design problem," *Int. J. Transp. Eng.*, vol. 1, no. 4, pp. 311–334, 2014.

[35] M. Zhang, R. Batta, and R. Nagi, "Modeling of workflow congestion and optimization of flow routing in a manufacturing/warehouse facility," *Manage. Sci.*, vol. 55, no. 2, pp. 267–280, 2009.

[36] G. Zhang, T. Nishi, S. D. O. Turner, K. Oga, and X. Li, "An integrated strategy for a production planning and warehouse layout problem: Modeling and solution approaches," *Omega*, vol. 68, no. 1, pp. 85–94, 2017.

Tatsushi Nishi (M'01–SM'19) received the B.S. degree in chemical engineering and the M.S. and Ph.D. degrees in process systems engineering from Kyoto University, Kyoto, Japan, in 1996, 1998, and 2001, respectively.

In 2001, he became a Research Associate with the Research Group of Systems Control and Engineering, Department of Electrical and Electronic Engineering, Okayama University, Okayama, Japan. He has been an Associate Professor with the Research Group of Systems Mathematics, Department of Mathematical Science for Social Systems, Graduate School of Engineering Science, Osaka University, Osaka, Japan, since March 2006. He has authored or coauthored over 200 technical publications. His research interests include discrete optimization, discrete event systems, production scheduling, decentralized optimization algorithms, supply chain optimization, multiple mobile robots control, and manufacturing systems.

Dr. Nishi has won several awards, including the National Instruments Corporation Young Investigator Award from the 2006 International Symposium on Flexible Automation, The 2010 Best Paper Award from the Society of Instrument and Control Engineers and 2007, the 2011 Best Paper Award from the Scheduling Society of Japan, the 2013 Best Paper Award from the Institute of Systems, Control and Information Engineers, and the Outstanding Paper Awards at the IEEE International Conference on Industrial Engineering and Engineering Management in 2014 and 2015.

Shuhei Akiyama received the B.S. and M.S. degrees from the Division of Mathematical Science and Social Systems, Graduate School of Engineering Science, Osaka University, Osaka, Japan, in 2017, 2019, respectively.

He was previously engaged in the research on the guide path design of AGV systems. Since 2019, he has been with Amazon, Tokyo, Japan.

Toshimitsu Higashi (M'99) received the M.E. degree from Waseda University, Tokyo, Japan, in 1992, and the D.E. degree from Nagoya University, Nagoya, Japan, in 2000.

He joined Murata Machinery, Ltd., Aichi, Japan, in 1992, where he is currently engaged in simulation and systems engineering at the Automated Systems Division. His research interests include self-organizing manufacturing systems, collective autonomy of distributed autonomous robotic systems, production scheduling, and optimization algorithms for large-scale production/material handling systems.

Kenji Kumagai (M'19) graduated from the Graduate School of Aerospace Engineering, National Defense Academy, Yokosuka, Tokyo, in 2004.

He joined Murata Machinery, Ltd., Tokyo, Japan, in 2009, where he is currently engaged in systems optimization in Research & Development Headquarters. His research interest is the optimization of large-scale production/material handling systems.