Int J Adv Manuf Technol (2008) 36:780–797
DOI 10.1007/s00170-006-0892-9

ORIGINAL ARTICLE

Development of an intelligent agent-based AGV controller for a flexible manufacturing system

Sharad Chandra Srivastava · Alok Kumar Choudhary · Surendra Kumar · M. K. Tiwari

Received: 30 June 2005 / Accepted: 17 November 2006 / Published online: 30 January 2007

Abstract Automated guided vehicles (AGVs) are the most flexible means to transport materials among workstations of a flexible manufacturing system. Complex issues associated with the design of AGV control of these systems are conflict-free shortest path, minimum time motion planning and deadlock avoidance. This research presents an intelligent agent-based framework to overcome the inefficacies associated with the aforementioned issues. Proposed approach describes the operational control of AGVs by integrating different activities such as path generation, journey time enumeration, collision and deadlock identification, waiting node location and its time estimation, and decision making on the selection of the conflict-free shortest feasible path. It represents efficient algorithms and rules associated with each agent for finding the conflict-free minimum time motion planning of AGVs, which are needed to navigate unidirectional and bidirectional flow path network. A collaborative architecture of AGV agent and its different modules are also presented. Three complex experimental scenarios are simulated to test the robustness of the proposed approach. It is shown that the proposed agent-based controller is capable of generating optimal, collision- and deadlock-free path with less computational efforts.

Keywords Automated guided vehicles (AGVs) · Flexible manufacturing system AGV control · Multi-agent system (MAS) · Deadlock avoidance

S. C. Srivastava · S. Kumar
Department of Production Engineering,
Birla Institute of Technology,
Mesra,
Ranchi 835 215, India

A. K. Choudhary
Wolfson School of Mechanical and Manufacturing Engineering,
Loughborough University,
Loughborough, UK

M. K. Tiwari (✉)
Department of Forge Technology, National Institute of Foundry and Forge Technology,
Ranchi 834003, India
e-mail: mkt09@hotmail.com

1 Introduction

Flexible manufacturing systems (FMS) aim to combine the productivity of flow lines with the flexibility of job shops, to attain very versatile manufacturing units achieving high operational efficiencies. They are particularly designed for low volume, high variety manufacturing, and good decision making and management are crucial to maximize the benefits that they offer. Stecke and Solberg [1] mentioned four decision stages for FMS, i.e., design, planning, scheduling and control. An FMS consists of a set of cells: material handling system (automated guided vehicles) and service centers, etc. Automated guided vehicle systems (AGVs) are advanced material handling devices extensively used in automated manufacturing systems (AMS) to transport materials among workstations [2]. The design and implementation of such AGVs require answers to a number of problems, such as guide path design, controller devices, and routing algorithms. In the recent past, with the emergence of distributive technologies [3] and advanced manufacturing paradigms, the AGV design and control problems attracted the attention of various researchers, including Tanachoco and Sinrich [4], Egbelu and Tanachoco [5], Egbelu [6], etc. AGVs are under computer control and they are the most flexible means to link all the locations of the shop floor [7–9], their operations must face some traffic control problem, such as collision prevention and deadlock avoidance [10, 11], and minimum time motion planning. The objective of this research is to provide an analytical treatment of some of these issues and to offer a novel perspective to resolve the issues related to collision- and deadlock-free vehicle routing with minimum time motion.

The collision and deadlock-free, shortest time, multiple automated guided vehicle system (MAGVs) are the need of today's FMS. There are varieties of AGVs available in the manufacturing system, serving a predefined flow path network, which consists of a set of nodes interconnected via set lanes (edge or links). Nodes represent sites for workstation and parking areas for vehicle and serve as pick-up and drop-off points for loads. A demand could be generated in the form of pick-up loads at any workstation and drop off a load to other workstations. The main problem to be tackled here is finding the shortest path of the AGVs from its present location to the destination station via intermediate pickup workstation. There are several possible paths through which the vehicle can travel, and among these shortest, a conflict- and deadlock-free path is to be adopted for routing. The vehicle's chosen path should be such that, it may not affect the others active travel schedule.

The main thrust of this research is to frame collaborative agent-based architecture for real time traffic control of AGVs, with a view to avoiding collisions and deadlocks and achieving minimum journey motion times in an FMS. It generates a conflict-free shortest path for the AGV in an effective manner and, therefore, can overcome the inefficacies that may arise in complex layout of manufacturing system. Agent controller, presented in this paper, controls the AGVs using modules. These modules are associated with rule-based system, heuristics and algorithms. Various agents used in this architecture are as follows: guide path agent, zone controller agent, journey time database agent, online traffic controller agent, AGV agent, order agent and interface agent. In addition, contract net protocols are used for fully automated competitive negotiation through the use of contracts among different agents.

The rest of the article is organized as follows: The literature relevant to collision- and deadlock-free minimum time motion planning with the application of artificial intelligence (AI) is reviewed in Sect. 2. Section 3 describes various complexity associated with the problem environment. An overview of agent technology and proposed structure of agent-based AGVs are mentioned in Sect. 4. Agent interaction approaches, various rules, heuristics, and algorithms associated with different agents are illustrated in Sect. 5. Section 6 discusses about the implementation aspects of proposed model. Three experimental scenarios are illustrated in Sect. 7. Results of simulation runs and effectiveness of the proposed approach are discussed next. Lastly, the paper ends with concluding remarks and future research directions.

2 Literature review

Research articles covering in depth knowledge in the broad domain of works related to collision- and deadlock-free, shortest time route and application of AI to the control aspect of AGVs, are considered here for discussion. Broadbent et al. [12] coined the concept of conflict-free shortest time route. A matrix has been generated by applying Dijkstra's shortest time path algorithm that describes the occupation time of a vehicle at a node. Conflicts at a node (junction) or catching-up conflicts can be resolved by slowing down the vehicle, which is yet to be scheduled. The procedure can be applied to any type of path, namely unidirectional and bidirectional, but the procedure does not guarantee an optimal solution of bidirectional models [13]. Introduced the concept of virtual tunnel for the bidirectional path consisting of several segment of multi-unidirectional paths. This concept allows multiple simultaneous crossing at an intersection.

Chang and Tanachoco [14] presented an algorithm for finding conflict-free shortest time route for AGVs in a network, which is based on Dijkstra's shortest path method. The main drawback of this procedure is its computational complexity, as the number of vehicles and number of nodes increases, the response time will increase. Narshimhan et al. [15] analyzed the rerouting of AGVs to encounter interruption via simulation. But, this procedure does not guarantee an optimal and conflict-free path. Oboth et al. [16] presented a route generation technique that provided conflict-free routes for multiple AGVs with varying speeds. It is unconfirmed whether this technique can guarantee the optimal conflict-free route. Recently, several researchers [17, 18] have attempted to solve the shop-floor control problems using the concept of agent technology.

Wallace [19] has presented novel agent approach, in which the rule-based intelligent agents act as traffic managers to allow or disallow mobile robots access to points (x-y coordinates within a two-dimensional world) and segments (lines connecting points) in a system. These rules and assessments define the interaction between the AGVs and guide-path that controls AGVs movement. As compared to this distributed approach, the proposed centralized approach to control the AGV movement, although less flexible, is more reliable and easier in implementation as well as it avoids many unnecessary agent-to-agent interactions. Lim et al. [20] suggested a construction algorithm to design a guide-path network for AGVs. Their study used the total travel time, including waiting and interference time, of the vehicle as the decision criteria for determining the direction of the path segment on the unidirectional guide-path layouts, and Q learning is utilized to estimate the travel time of vehicles on path segments. Fanti [21] has formulated a zone control scheme to tackle the traffic control problems, based on the knowledge of the AGVs' operative condition. Lee et al. [22] proposed a systematic two staged traffic control scheme to obtain collision free minimum time motion of AGVs along loop less path. Kim et al. [23] modeled an intelligent agent that lies in the two extremes i.e., heterarchical and hierarchical frameworks to solve an industrial warehousing problem. Yamashita [24] proposed two empty vehicle dispatching policies, and numerically calculated the waiting time distributions of the items for each policy using a state space reduction technique for Markov chains.

In this research, the main focus is to resolve some of the complex issues concerning the collision and deadlock avoidance with minimum time motion planning pertaining to operational control of AGVs. This paper presents an multi-agent-based framework representing a zone controlled AGV environment, incorporating various issues like path generation, link occupation time, collision and deadlock avoidance, and suggests the waiting node, waiting time, positioning of the idle AGVs, pick-up and drop-off nodes associated with an shortest conflict-free path selection. However the intelligent agent-based framework can be used to overcome the shortcomings associated with previous approaches and incorporate many beneficial facilities explained as follows.

i. It is a complete framework to develop a minimum time motion plan for AGVs and to govern the AGVs as same plan.
ii. The system provides collision- and deadlock-free AGV movement and can solve the breakdown problems (if any).
iii. This centralized MAS control is more reliable and easy to implement onto shop floor. The shop-floor controller acts as mediator among other shop-floor activities and AGV movements that can act to establish seamless co-ordination among each activities of the shop floor.
iv. Pre-detected time of reaching the product at its target is quite important for establishing co-ordination among shop-floor activities.
v. The system can not only serve the AGVs on complex guide-path but also efficiently control the movement of AGVs even if the number of AGVs is increased.

3 Shop-floor environment

The problem considered here is related to the dynamic environment of flexible manufacturing systems (FMS), where fixed numbers of AGVs are available to handle material transfer requests. An order or demand for a vehicle implies that the vehicle is to move from its present location (initial node) to a node of a pick-up workstation (intermediate node), where the jobs for loading are available. From there, the loaded AGV proceeds to a node representing a drop-off workstation (final node) for unloading, where the AGV is free to be routed again.

Distributive traffic control has been adopted in many practical situations, where the range sensors maintain a minimum headway between two adjacent vehicles. When the complexity of the guide path network is high, these distributive controls resulted in decreased efficiency of the system. In these situations, it is difficult to find a substitute path, and deadlock occurs more frequently with an increase in the number of AGVs. In this research, we have considered centralized traffic control scheme for multiple AGV systems with complicated bidirectional guide paths. On shop floors, the exact position of an AGV can be obtained only when they pass over pre-specified control points fixed on the guide path. Hence, the zone blocking technique, which permits only one vehicle in a given path segment at a time is suitable for multiple automated guided vehicles under centralized traffic control [25, 26]. In such a system, the guide path is divided into a set of many small path segments; these segments are termed as a link of the network. The link from node x to one of its adjacent node y is denoted as (x, y). For this link, node x is said to be adjacent to node y and vice versa. For each pair of adjacent nodes, link (x, y) is said to be unique. For bidirectional path segment two links (x,y) and (y,x) are considered separately, and (y, x) is said to be the inverse link of (x,y).

In this paper, we have adopted some implicit assumptions to map the shop-floor scenario. Vehicles can start and stop only at nodes. In order to avoid collision, temporary stay at some nodes is permitted. A spin turn of AGVs on a guide path is assumed to be avoided. It is also assumed that a vehicle path cannot contain any loops or partial paths, whose start node is the same as its goal node. A rapezoidal velocity profile is used, and the maximum speed for each profile is fixed at its maximum vehicle speed V_{max} multiplied by velocity parameter η_{xy} ($0<\eta_{xy}\leq 1$) that is assigned to each link. Acceleration and deceleration of AGVs are assumed to be constant, and can be denoted as α_{acc} and α_{dec}. Disruption and rescheduling of any active travel schedule for other vehicle is not permitted.

Journey time for a path P [I, F] can be determined using the aforementioned variables. Here (x, y)∈P [I, F], for link (x, y), when AGV move over it, the time require to clear (x, y) equals to $(\eta_{xy}.\ V_{max})^{-1}.\ l_{xy}$. Here, journey time for a path P is represented by:

$$\tau(p)^{\Delta} = \sum \left(\eta_{xy}\nu_{\max}\right)^{-1} \cdot l(x,y) + \sum_{i} \theta_i + \sum_{j} \varphi_j + \sum_{r} \xi_r \ldots \quad (1)$$

where

θ_i	Additional time for ith acceleration.
ϕ_j	Additional time for jth deceleration.
ξ_r	Additional time for rth temporary stay.
P [I, F]	Path P from I to F, where I is the initial or start node and F is the final or goal node
$\tau(P)$	Journey time for a path P
V_{max}	Maximum velocity of the AGV
η_{xy}	Velocity parameter associated with the link ($0<\eta_{xy}\leq 1$)
l_{xy}	Length of link (x,y).

Entry time and exit time, associated with each link, is known using journey time. Suppose an AGV is moving along a path P [I, F] then entry time of (x, y) is estimated using $\tau[\{I. . . x\}]$ while the exit time is calculated as $\tau[\{I,..P, y\}]$. A very small response time is also considered. This is the time from the start of the algorithm execution to the start of the vehicle movement. It includes the computational time of the algorithm and the communication delay needed to transmit the travel schedule to the vehicle.

4 Background information on agent technology

4.1 Definition of agent

It is well documented that there are sundry definitions of the term agent reported by several researchers in the recent past. Fisher [27] defined agent as an encapsulated entity with embedded AI capabilities. Jennings and Wooldridge [28] defined agent to be an autonomous problem-solving entity, which by nature is communicative, reactive, and goal oriented. Davidson et al. [29] mentioned that agent is an entity with an ability to interact independently with its environment through its own sensors and effectors [30]. Nwana and Ndumu [31] defined an agent to be a software and/or hardware component capable of acting rationally, in order to accomplish tasks on account of its user. Huang and Nof [32] suggest that an agent is a collaborative computing system capable of reacting autonomously and responding reflexly to the impacts from the environment in a goal oriented paradigm. Moreover, literature available on role and definition of agents are vast and beyond the scope of this section. More details about agents can be found in [33–37].

4.2 Agent properties and types

According to Jennings and Wooldridge [28], an agent can exhibit autonomy, social ability, responsiveness and proactiveness, in addition to adaptability, mobility, veracity and rationality. Furthermore, agents may have high and low level reasoning capabilities [29]. In this paper, we have defined an agent as a computational entity that acts on behalf of others is autonomous, both proactive and reactive, and exhibit certain degree of ability to learn; cooperate and can move after receiving the details of certain task. An agent acts autonomously following certain algorithm. Based on their skills, an agent tries to attain the goal defined by the assigned task in a proactive manner. In order to achieve the goal, agent must be mobile, collaborative, and communicative to share their knowledge.

4.3 Agent-based architecture of AGVs

Three basic elements of agents have been identified: network interface, local knowledge model and domain knowledge model. Network interface consists of the agents to the network. Various resources like Java classes, other agent's address, message language KQML, ontology and strategies comes into the purview of local knowledge model. Domain knowledge model comprises of dexterity essential for an agent to perform functional tasks and skills that may be the method for activating action corresponding to the received request. Figure 1 represents schematic structure of an agent.

In order to implement the agent characteristics in to AGVs following functional modules are required: communication interface, perception, social knowledge, self knowledge, domain knowledge, learning, reasoning, problem solving interface, coordination, planning scheduling and control, conflict management, application interface, etc. In this paper, we have developed a collaborative architecture incorporating the features from other architectures and thus provide powerful and promising capabilities. The internal architecture for AGV agent is shown in Fig. 2.

4.4 Communication protocols used in agent network

The protocol aims to efficiently and effectively distribute the task among the different agents to ensure the completion of task smoothly and efficiently. It allows agent to share the information therefore determine the overall behavior and organization of MAS (multi-agent system). Figure 3 shows the mechanism of sending and receiving information tasks through communication bus (internet/intranet).

In this proposed MAS, all agents communicate over the Internet/intranet via a bundle of knowledge query and manipulation language (KQML) messages to transfer data among each other. Agent communication language (ACL) acts as formalism for exchanging messages. It consists of three layers: a content layer, a message layer and a communication layer. The actual message is specified by

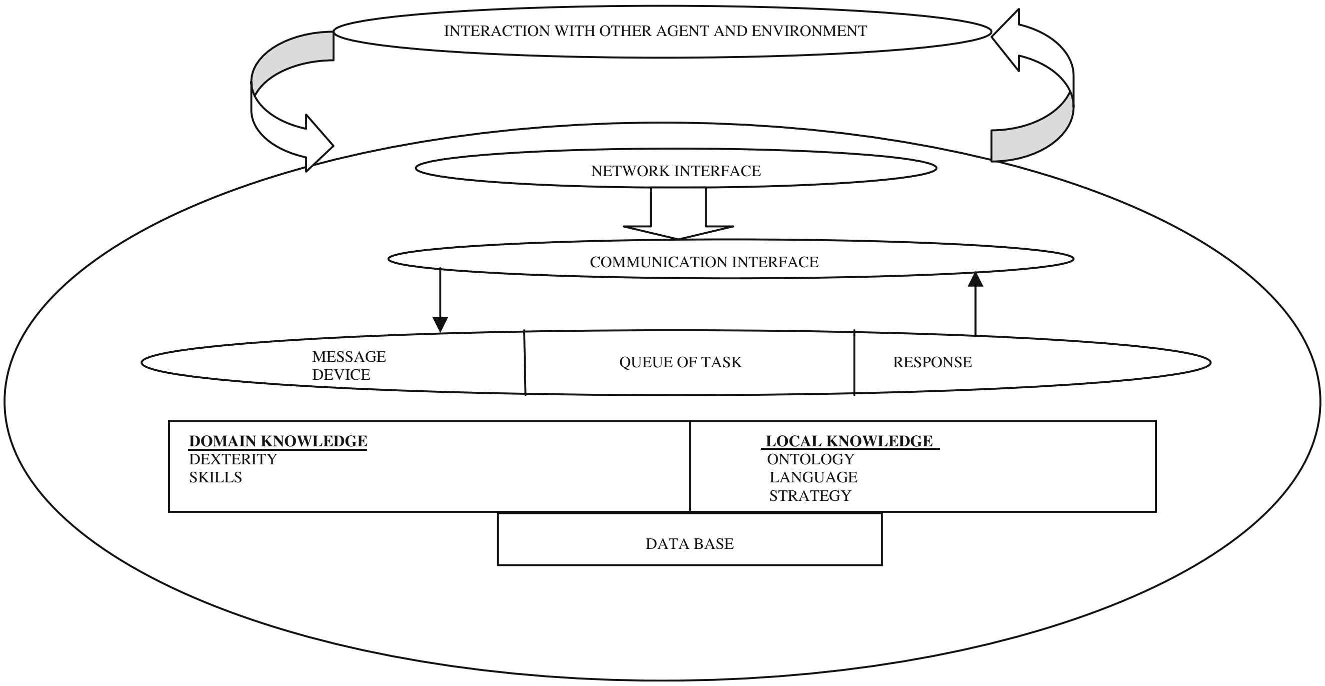

Fig. 1 Agent's architecture

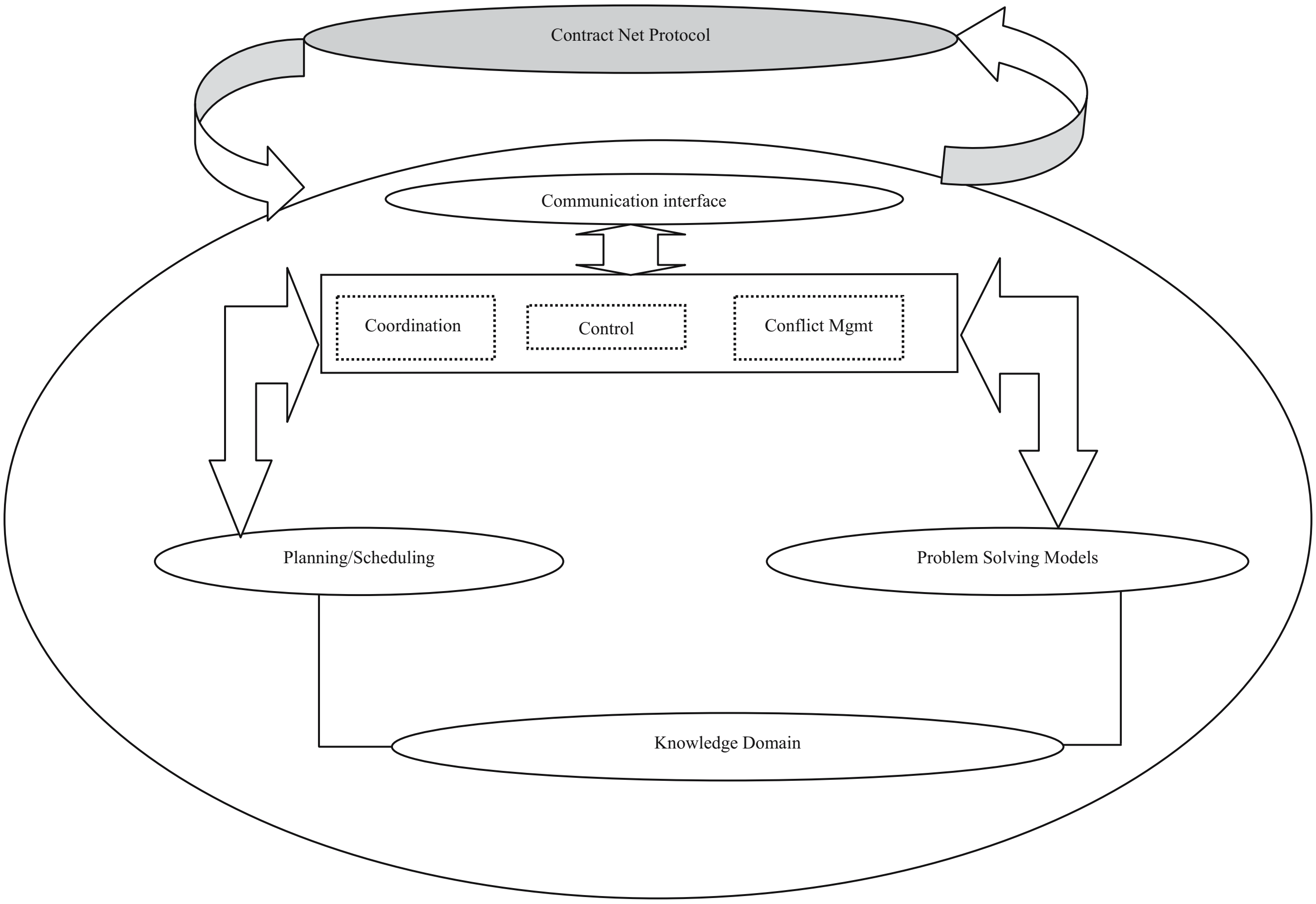

Fig. 2 Architecture for AGV agent

the content layer. The message layer comprises performatives which are provided by the language (e.g., tell, reply, decision making and ask-if), which correspond to the speech act from the speech act theory [36]. The performative header defines what the message means and what the recipient agent should do. For the implementation aspect of the proposed structure, small subsets of performatives have been customized, as described in Table 4. When a message is transmitted among the agents, it is wrapped by the KQML in a standard format. The destination agent can compose the KQML and retrieve the embedded message. The agent language (AL) is developed for each agent to understand the message it receive and execute the task as message 'tells'.

4.5 Agent's functionality

This paper presents an idea consisting of society agents to resolve the issue related to conflict, deadlock and interruption occurring in a guide path network. Six types of agent have been proposed to develop the architecture of the control mechanism of AGVs. Each agent is associated with modules and these modules follow some rule bases, heuristics and algorithms. Figure 4 represents the architecture of a proposed multi-agent system framework and coordination, and negotiation among these agents has been demonstrated in Table 4.

These agents are described as follows:

1. 4.6.1 Guide path (GP) agent 1. The guide path agent is responsible for finding the possible paths through which the AGVs can be routed. A guide path agent receives information from an AGV agent regarding its present location, location of pick-up, and drop- off workstation. GP agent generates shortest feasible path and K shortest feasible path based on high level reasoning capability and different modules associated with it. The working methodology and generation of shortest path and K shortest path is illustrated in Sect. 5.
   Based on these factors link occupation time is estimated by JTD agent.
2. 4.6.2 Journey time database (JTD) agent 1. The journey time database agent enumerates the link occupation time of each AGVs. Link occupation time is the interval between the entry time and exit time of a link. It also includes the response time that is nothing but the time from the start of the algorithm execution until start of the vehicle movement. In Sect. 5, an illustration has been cited for estimating the link occupation time. These informations are utilized by zone controller agent to plan the trajectory along a candidate path.
3. 4.6.3 Zone controller (ZC) agent 1. A zone controller agent utilizes link occupation table data to determine a collision- and deadlock-free trajectory of a vehicle. Hence, a ZC agent is mainly responsible for trajectory planning of AGVs. However, it is considered that potential collision occur if its link occupation time overlaps any active occupation schedule by other vehicle. In this case, ZC agent searches for temporary staying node and time. In order to avoid collision/ deadlock, ZC agent is responsible for making two types of decisions as discussed in Sect. 5.
4. 4.6.4 Online traffic controller (OTC) agent 1. OTC agent determines the overall motion planning of AGVs. This agent is also termed as decision maker. The OTC agent on the basis of communication takes decision related to the shortest feasible path with other agent and heuristics and rule bases associated with them. Details of rules and heuristics are described in Sect. 5. After deciding the shortest feasible path, OTC agent instructs the AGV agents to initiate its motion and continuously governs its movement. If any problems related to the AGVs control (probable location to be head-on collision of AGVs, breakdown of AGVs) arise, it reports the shop-floor controller to heal up the trouble.
5. 4.6.5 Order agent (OA) 1. As generation of new order to transfer the supplies from one station to another

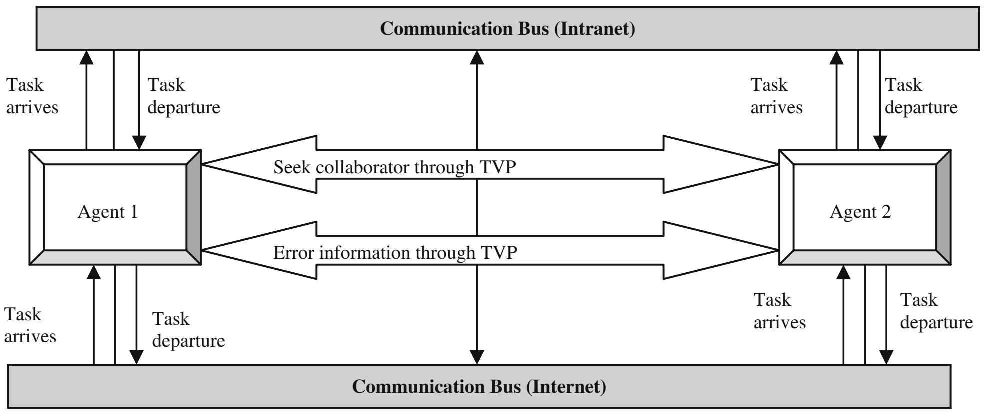

Fig. 3 Agents sending and receiving information/tasks through a communication bus

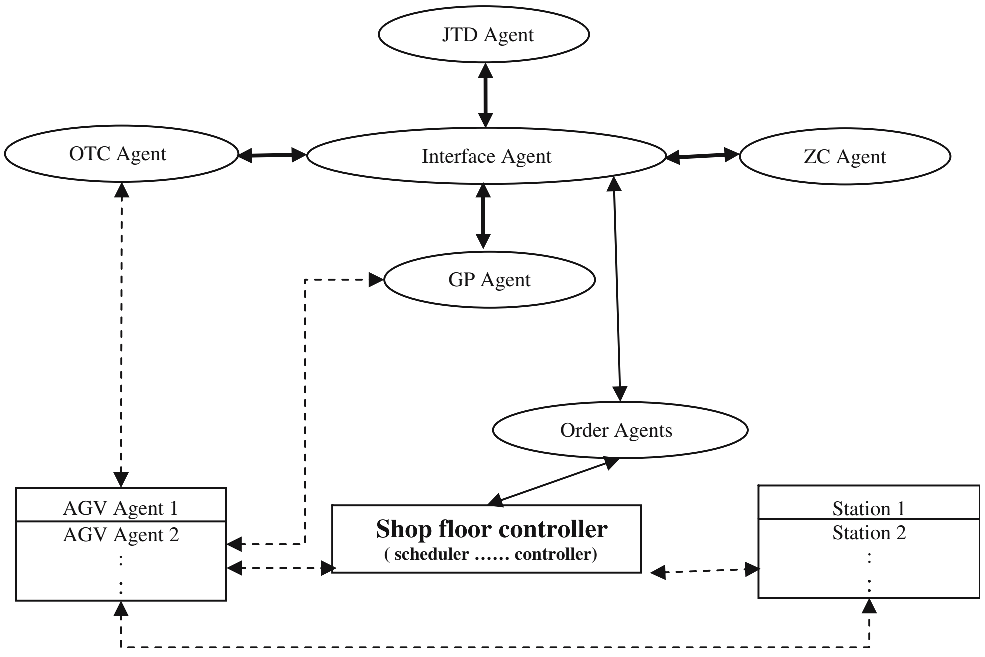

Fig. 4 Agent-based system architecture using proposed approach, (dashed line shows the Internet-based communication, and solid lines show the Intranet based communication)

station or entry of new supplies in the system arise; shop-floor controller detects the requirement of AGVs to transport the supplies from station to station. It instructs the order agent to develop a plan for transportation of the supplies.

6. 4.6.6 AGV agent 1. Each AGV in the system is referred to as an AGV agent, and its routing plan is managed by OTCA. It negotiates with an OTC agent at every incident occurred with AGVs (loading, unloading, information to cross of each node, breakdown, etc.). A detailed structure of AGV agent is outlined in Sect. 4. Figure 2 represents the architecture of AGV agent.

4.6 Agent-based system architecture

Once the agents are available, there are two possible architectures from which to choose: (1) when all the agents handle their own coordination or (2) a group of agents rely on special system programs to achieve coordination. The disadvantage of this first architecture is related to the communication overhead, especially the scalability requirement, which is essential for better communication among agents. As a consequence, the latter federated approach is preferred in which agents do not communicate directly with other agents. Instead, they communicate through a system program called interface agents. This leads to the development of a framework where the agents form a federation in which they surrender their autonomy to their centralized agent. The centralized agent takes the responsibility of fulfilling their needs.

The proposed system is shown in Fig. 5, which consists of seven components: interface agent, GP agent, AGV agent, order agent, JTD agent, ZC agent, and OTC agent.

The interface agent is responsible for the management of interaction and the conflict resolution in the agent community (GP agent, order agent, JTD agent, ZC agent, and OTC agent). It contains the knowledge about capability and function of each connected agent. Therefore, the message can properly pass through the interface agent The interface agent routes the request/response received to the appropriate agent based on its knowledge about the capability of each agent.

5 Agent interaction approaches

An AGV system is based on the guide path constructed of nodes and links having some rules, heuristics and algorithm associated with the agents to control the AGVs. It involves a natural choice that agent should follow to select the rules and heuristics inherent in them. This section presents heuristics and rules associated with agents. The working methodologies of each agent, along with the interaction mechanism among them are also discussed here.

5.1 GP agent

Two algorithms are associated with this agent. The functioning of guide path agent is based on these algorithms. These algorithms are termed as the shortest feasible path (SFP) algorithm [38] and K- shortest feasible path (KSFP) algorithm [39, 40]. These algorithms find the shortest and K shortest feasible path from a start node I to

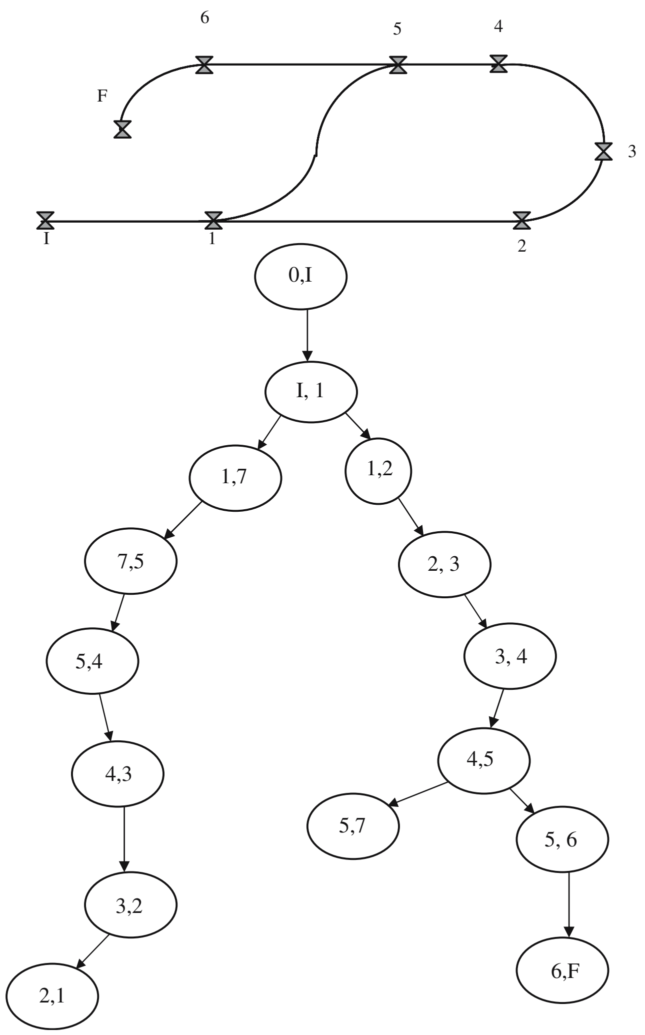

Fig. 5 An example for finding the shortest feasible path via SFP algorithm

goal node F, respectively. Agent-based control of AGVs requires construction of a set of multiple feasible solutions, which are called path table [22]. Hence, path table $PT_{I/F}$ for a start node I to goal node F is a collection of candidate path that contains the shortest feasible path and 2nd, 3rd .... Kth shortest loop less feasible path. Here, K is an input parameter defined by the operator. These algorithms are based on the principle of partitioning the solution space [41]. In this paper, a concept of an induced network model has been visualized to find the shortest feasible path instead of a linkage-constrained network. For a detailed study of linkage-constrained network and induced network model, it is advised to refer [22, 41]. The intension path *P* for a path $p[0_I, p_n] = \{0_I, I_{m...}, p_n\}$ in G_I (N_c, L_c) is defined as the path {I,m,...p,n} on G (N, L) constructed from the base node of each element in *P*. The intension path of any path on G_I (N_c, L_c) is a feasible path on G(N, L) and vice versa. Hence, graph search that performed only on G_I (N_c, L_c) will deal with only feasible path in G (N, L). Some of the notations, which are frequently used in the description of shortest feasible path and K ,shortest feasible Ppath algorithm, are listed in Table 1.

These algorithms are discussed as follows.

5.1.1 Shortest feasible path (SFP) algorithm

SFP Algorithm [42] proceeds to grow like a tree $\Gamma\left(N_C^t, L_C^t\right)$ using the original network G (N,L) until the shortest feasible path from I to F is found. Let a label dist. (p_n) for each $p_n \in N_C^t$ denote the feasible path length from 0_s to p_n. A priority queue £ is also defined, in which the leaf nodes in Γ_I are sorted in their order of Dist (.). SFP algorithm is delineated as follows:

[SFP Algorithm]

Initialization Step: let $0_I \in N_c$ be the only node included in $\Gamma_I \Rightarrow N_c=\{0_I\}$,

Set dist. $(0_I)=0$, Let £=$\{0_I\}$

Step 1 Among the leaf nodes of Γ_I, minimum Dist(.) value is picked up. Choose arbitrarily if tie occurs. Let the leaf node picked up be p_n, now remove p_n from £.
p_n is chosen in such a manner that, $p_n \in £ \subset N_C^t$ and $Dist(p_n) = \min_{u_n \in £}(Dist(u_n))$
p_n is excluded from £, hence, $p_n \in \left(N_C^t - £\right)$

Step 2 If n=F then stop, shortest feasible path is $p[0_I, p_n]$ in Γ_I.

Step 3 Expand p_n, n_m a child node of p_n is generated by selecting m's from *Adj (n)*. All the m's are not selected even if m is feasible from p via n, but only those are considered as the base node for the child node of p_n that met the criteria given below:

Table 1 List of notations used in SFP and KSFP algorithm

Notations		Definition
G (N, L)	:	Linkage-constrained network
G_I (N_c, L_c)	:	Induced network of a given linkage constrained network G (N, L) with start node I
Adj (n)	:	Adjacent of node n
d (p_n, n_m)	:	Distance between p_n and n_m
Π_I	:	Shortest feasible path
Π_i	:	ith Shortest feasible path
S_c	:	Set of candidate path
B[n]	:	{(n, m)} where m∈*Adj*(n)
ψ	:	Set of feasible path from start node I to goal node F
ϕ	:	Null set
P(0_I, p_n)	:	It is defined for each (p,n)∈L, a new element in N_c
$\Gamma\left(N_C^t, L_C^t\right)$	:	Growth of a tree using original network G (N,L)
0_I	:	Start node
Dist(p_n)	:	A label denoting feasible path length from 0_I, to p_n
£	:	Priority queue

```
          For m∈Adj(n), let $n_m \in £ \subset N_C^t$ if:
          m is feasible from p via n, except the case $p_n = 0_I$,
          In $\Gamma_I$, m is not a base node of the ancestor node of $p_n$,
          $n_m$ is not repeated in $\Gamma_I$, i.e., $n_m \notin N_C^t$ yet.
          For such $n_m$, let $(p_n, n_m) \in N_C^t$, and $Dist(n_m) = Dist(p_n) + d(p_n, n_m)$.
Step 4    If £=$\phi$ stop. Otherwise return to step 1 as there is no feasible path from I to F.
```

Here, it is pertinent to mention that the SFP algorithm behaves as greedy algorithm. Figure 5 illustrates the approach for finding shortest feasible path via SFP algorithm. Although it seems strange that from the star node I to goal node F, there is no direct link, yet it is a part of the KSFP algorithm to eliminate one or more links from the original network.

5.1.2 KSFP Algorithm

This algorithm is evolved keeping in mind the principle of partitioning the solution space provided that initial shortest feasible solution exists that can be found from SFP algorithm. Algorithm KSFP is delineated as follows:

```
[KSFP Algorithm]
Initialization step: Using the SFP Algorithm, find the first shortest path, set k=1, $S_c$=ϕ Find $\Pi_1$[I, F].
Step 1  [Generation of feasible deviation path: Shortest feasible deviation path from $\psi_j^i, j = 1 \ldots \Pi_I$ which branches at jth node of the k^th shortest feasible path is found out. Put the paths thus found in $S_c$].
    Loop 1: For jth node of $\Pi_k$ [I, F] ( j=1,.... $\pi_k$),
    {
    Define B*[n] as a subset of B[n]. Let ($n_k$( j),m)∈B* [$n_k$, j], if $n_k$( j, m) is not repeated in $\Pi_1$[I, F],...., $\Pi_k$[I, F] and furthermore m is feasible from $n_k$( j−1) via $n_k$(j) when j≠0.
    Loop 2: For each [$n_k$( j),m]∈B* [$n_k$, j]
    {
    Set d($n_i$(i), . )=∞∀$_I$... ..., j−1.
    Set d($n_k$( j), .)=∞ except for d($n_k$( j), m).
    Find $\Pi_1$($n_k$,( j), g) by applying SFP algorithm.
    Return d(. , .)'s that are set to ∞ in 1) and 2) to the former values
    $\Pi_1$($n_k$,( j), g) does not exist return to 1) for the next ($n_k$, ( j),m)
    Include {I,...$n_k$( j )}+$\Pi_{1}$($n_k$,( j),F) in $S_c$ (if it is not already in $S_c$), where {I,...$n_k$( j ) } is the sub path of $\Pi_I$[I, F].
    } // end of Loop 2
    } // end of Loop 1
Step 2  Estimate $\Pi_{k+1}$[I, F], find the shortest path in $S_c$, and set the path as $\Pi_{k+1}$[I, F],
        Check if $S_c$=φ, stop, since $\Pi_{k+1}$, ... , $\Pi_K$ distinguished from $\Pi_1$, . . ., $\Pi_k$ do not exist.
        Shortest path among $S_c$ is chosen. Let it be $\Pi_{i+1}$[I, F].
        Eliminate $\Pi_{k+1}$[I, F] from $S_c$.
Step 3  Check if k=K stop; otherwise k=k+1 and go to step 1.
```

After applying KSFP algorithm, one constructs a path table. Each table contains K loop less path having same start and goal nodes. The K loop less path in the table is shortenend and stored in data files in order of their journey times. Now, all listed paths are referred to the journey time database agent to determine the link occupation time associated with each link in corresponding path.

5.2 Journey time database (JDT) agent

The journey time database agent generates link occupation time data according to vehicle speed. Link occupation time is the interval between the entry time and exit time of a vehicle on a link. Link occupation time for every link is stored in the form of link occupation table (LOT). Hence, after dispatching of a vehicle, the link occupation schedule of the vehicle is stored in LOT. ZC agent utilizes LOT to determine collision free trajectory of a vehicle. Figure 6 represents the proposed data structure of each link.

Some preference rules are also considered while considering the motion of AGVs. These preference rules are delineated as follows:

– AGV will move through the straight edge rather than two curved situations.
– AGVs will move through the curved edge rather than two straight edges.
– Before departing from initial node, the vehicle will acquire the node position.

```
[Link Occupation Table (LOT) Algorithm]
Step 1  Refer the path table and determine the set of links at which link occupation time is to be calculated. Initial link refers to that link whose initial node contains the idle vehicle.
Step 2  Input the entry time of vehicle at the initial link
Step 3  Equation 1 is used to determine the exit time, or link occupation time of the vehicle.
Step 4  Enclose the entry time and exit time in [ ].
Step 5  Refer to the path table, determine the next link where the vehicle has to proceed further.
Step 6  Repeat step 1 to step 5, until the completion of link occupation data related to all the links of selected path.
Step 7  ∞ is placed at exit time of final link and 0 is placed at link number.
```

∞ Shows the unknown idleness of the vehicle at final nodes loop.

0 Shows unassigned link for the movement of AGVs.

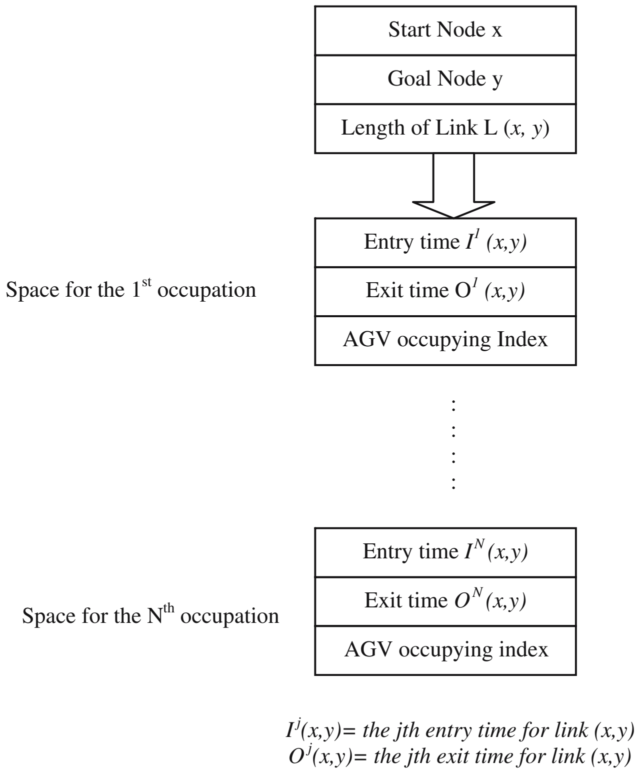

Fig. 6 Data structure for link occupation table

An illustrative example has been expressed for the calculation of link occupation time and is represented in Fig. 7. If an AGV travel from node x to node y, its adjacent node y during a time interval (t_x, t_y), the link (x,y) , and its inverse link (y,x), and all the links and inverse links starting at x or y are considered to be occupied during (t_x, t_y), such a mechanism would protect vehicles from collision at the intersection node.

5.3 Zone controller agent (ZCA)

The zone controller agent is responsible for determining the conflict and interruption free path/route associated with other moving vehicle with in the horizon of journey time schedule. The ZC agent determines the collision free trajectory along the K candidate path in the table that contain K-shortest feasible path from current node to goal node. In order to avoid collision, LOT is exploited by ZCA to ensure that two vehicles do not occupy a link. Relevant rules associated with ZC agent are as follows.

5.3.1 LOT rule

Overlapping of link occupation time is avoided; ZCA checks the link occupation table and plan in such a manner that link occupation times of the vehicle do not overlap with other vehicles.

5.3.2 Intersection node rule

These rules are delineated as follows:

- Neighbor links of physically occupied link have also to be considered as occupied. It is necessary to avoid collision at intersection nodes and also guarantee safety.
- Entry of the two vehicles can be made possible at any node if both the vehicle have to travel on different edges.
- Entry of the vehicle at the same or different time is possible at any node but the link occupation time constraint must be observed in case the vehicles have to travel on the same link.

A potential collision occurs if the link occupation times overlap by any occupation schedule of other vehicle. In this case, ZC agent estimates the temporary staying node and node occupation time. Based on the type of collision, temporary staying node and time are determined. In this work, we have considered four types of collision. Figure 8 illustrates the schematic representation of these four types. Each type of collision determines different temporary staying nodes and times. In this case, hitting of a vehicle from rear by a faster moving vehicle is not allowed as η determines the speed of the vehicle on link.

Figure 9 represents an illustration for trajectory planning of head on collision case. AGV 1 is dispatched along the path Π_1=Π_1\{1,10\}=\{1,2,... 10\}. AGV 2 is to be scheduled and its candidate path is Π_2=Π_2\{11,15\}=\{11, 12, 8, . . . 4, 14, 15\}. Suppose that occupation time of AGV 2 overlaps with occupation time of previously dispatched AGV 1. Then, temporary staying node and time of AGV 2 is determined using trajectory planning algorithm as follows:

5.3.3 Trajectory planning

[TP Algorithm]

Step 1 Refer to LOT find the possible collision on any link in $\prod_2$. Trajectory planning is completed if no potential collision is detected. For the selected case detected link is (6, 5).

Step 2 Get the index of AGV occupying that link. For this case, AGV 1 occupies (6, 5).

Step 3 Take inverse sequence of $\prod_2$ ($inv\prod_2$), for this case $inv\prod_2$=(15,14,4 . . . 8, 12,11)

Step 4 Choose start node (β) of the link that bears potential collision (β=6 for (6, 5)). The temporary staying node is determined as the first element $\xi \in inv\prod_2$ after (β) such that $\xi \notin \prod_1$ *in this case* ξ=12.

Step 5 Pick (ξ) from $\prod_2$, Denote expected entry time by later dispatched AGV, for this case (12,8) is

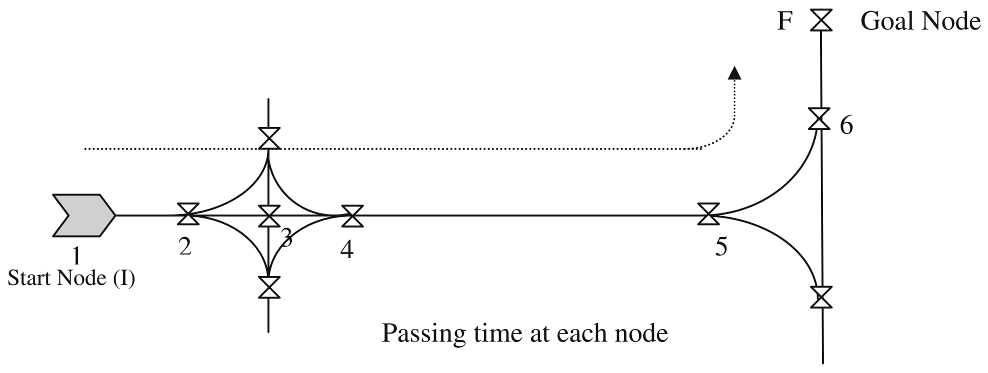

Fig. 7 An example representing calculation of link occupation time

Node 1 = 0: 00: 00 Node 2 = 0: 00: 10 Node 3 = 0: 00 : 18
Node 4 = 0: 00: 26 Node 5= 0:00: 36 Node 6 = 0: 00: 50
Node g = 0: 00: 58

Link (x,y)	Entry time of $I^i(x,y)$	Exit time of $O^{-i}(x,y)$
(1,2), (2,1)	0: 00: 00	0: 00: 18
(2,3), (3,2)	0: 00: 00	0: 00: 26
(3,4), (4,3)	0: 00: 10	0: 00: 26
(2,7), (7, 2)	0: 00: 00	0: 00: 18
(2, 8), (8, 2)	0: 00: 00	0: 00: 18
(3, 7), (7, 3)	0: 00: 10	0: 00: 26
(3, 8), (8,3)	0: 00: 10	0: 00: 26
(4, 7), (7, 4)	0: 00: 18	0: 00: 36
(4, 8), (8, 4)	0: 00: 18	0: 00: 36
(4, 5), (5, 4)	0: 00: 18	0: 00: 50
(5, 6), (6, 5)	0: 00: 26	0: 00: 58
(5, 9), (9, 5)	0: 00: 26	0: 00: 50
(6, 9), (9, 6)	0: 00: 36	0: 00: 58
(6, g), (g, 6)	0: 00: 36	∞

chosen and I*(12,8). Among the exit times O^i (12,8) (i=1,2 . . . N) for link (12,8), select i in such a manner that O^i (12,8) is the smallest exit time that is greater than I*(12,8).

Step 6 Difference in the expected entry time and smaller exit time gives the temporary staying time (ξ_r=O^i (12, 8)−I*(12, 8)). In order to occupy this temporary stays modify the trajectory of AGV 2.

Step 7 Calculating ϕ_i, φ_j for the temporary stay.

Step 8 Detect if there is any potential collision, then repeat step 1–7, else stop.

5.4 Online traffic controller agent

This agent is also known as decision-maker and responsible for overall motion planning of the AGVs. Based on overall motion planning algorithm and selection rule, OTC agent decides the optimal path to be traversed by AGV.

[Overall motion planning algorithm]

Step 1 initialize i=1

Step 2 Communicating with ZC agent, plan the trajectory along the path $\prod_i$

Step 3 Check for $\prod_i$, if there is no potential collision detected, i.e., node staying time=0, the collision free minimum time motion path lies among $\prod_k$ (1≤k≤i). The smallest $\tau(\Pi)$ among $\prod_1$, $\prod_2$, $\prod_3$...$\prod_i$ is selected.

Step 4 If node staying time is greater than zero, and i<I for $\prod_I$, go to step 2 and increment i=+1.

Step 5 If node staying time is greater than zero and i=I; the path $\prod_k$ (1≤k≤i) that has the smallest $\prod_k$ among $\prod_1$, $\prod_2$, $\prod_3$,...$\prod_i$ is selected. Step 2 determines the trajectory planning.

TIE breaking rule If the tie occurs, that is if the two paths show same reachable time at destination, OTC agent prefers the route having less traffic by consulting ZC agent.

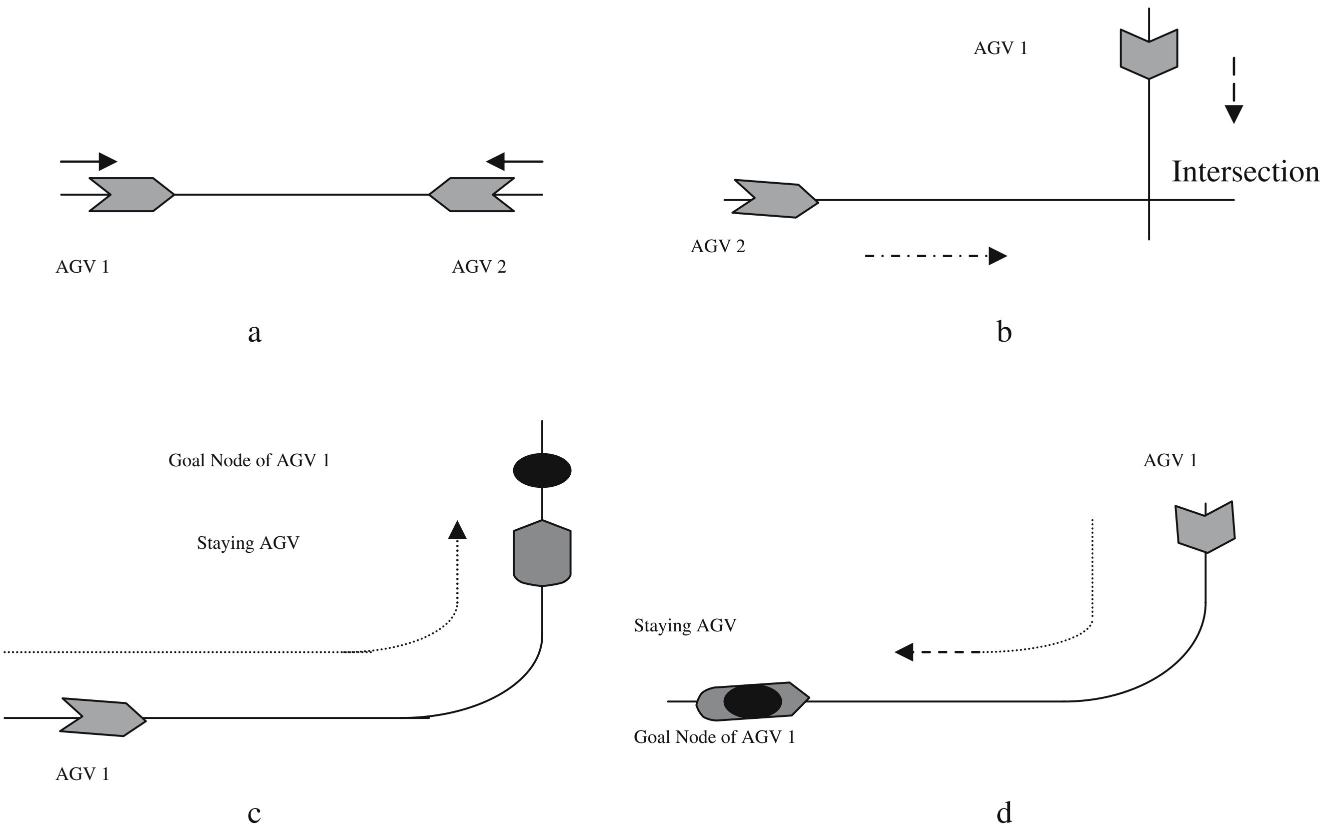

Fig. 8 Representation of four different types of collisions (**a**) Head on collision, (**b**,**c**,**d**) Intersection collision

5.5 Order agent

When order agents receive information about requirement to transfer the supplies from shop-floor controller, OA passes the information to other agent of the system for finding and solving the transportation demand. According to the OTCA instructions, AGVs can load and dispatch the supplies. It is not possible all the time that AGV agent executes the order, in some cases; it may lead to deadlock situation during fulfilling an order. Hence, it is not safe to execute the order. Order agents are used to prevent such situations. Concept of virtual order comes into picture when order is not safe to carry out. In order to execute real order, AGV is given virtual order to take it to a safe point or an alternative load and unload point to start the execution of the real order. Order agent works on the principle of accept an order rule [19] and is delineated as follows.

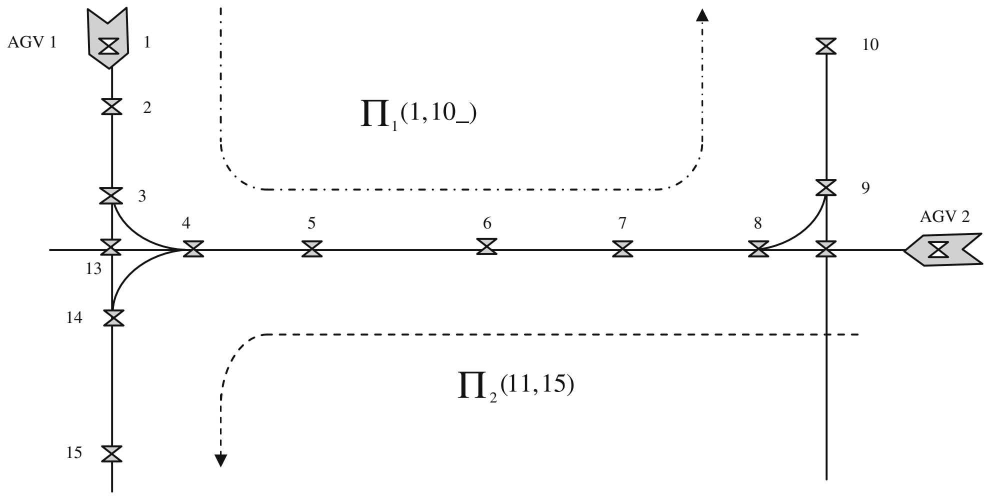

Fig. 9 Determination of temporary staying node and time using trajectory planning algorithm

ACCEPT an ORDER RULE The prime concern of an order agent is to prevent the deadlock situation of an AGV. Order agent checks every order to see whether the order lays down an AGV to its safe point or its goal node without putting AGV into deadlock situation. If the order would result in deadlock situation then a virtual order is given to an AGV until it is safe to carry out the original order, or the AGV would have to remain where it is until it could fulfill a virtual or a real order.

Negotiations among agents play a vital role during the fulfillment of an order. If the blocking situation arises then AGVs negotiate with each other for the movement of the AGV. If this fails then the blocked AGV will try to find another route or dummy destination to go to so that it doesn't put itself or other AGVs in a possibility of deadlock. Negotiation is based on high level contract net protocol.

5.6 AGV agent

Each AGV is associated with an AGV agent. AGV agent manages AGV movement. These AGV agents manage an AGV by initiating enquiries with other agents and by negotiation with other AGV agent. The detailed collaborative structure of proposed AGV agent is discussed in Sect. 4. An AGV agent makes a decision on the basis of message sent by OTC agent. AGV agents communicate with OTC agent at each incident such as AGVs cross the node, receives supplies and unloading the supplies etc. If any AGV becomes breakdown or AGVs come into location to be head-on collision, OTC agent indicates to the shop-floor controller to recover the AGV or/and to reschedule the movement plan remained journey of supplies. Some of the rules associated with AGV agent are delineated as follows:

– Waiting rule: After arriving at its destination, AGV performs the operation of loading and unloading. In order to complete its operation, it has to be allowed to wait on its current point. A virtual order is given to an AGV when it is unable to accept an order. If, it cannot accept the virtual order then it is unable to move anywhere.
– Follow other rules: In order to avoid deadlock, AGVs are allowed to follow OTC agent and hence the situation results in greater efficiency of the system.

The following section describes the implementation aspect of the proposed framework.

6 Implementation aspect of the proposed model

With the development of agent-based technology, recently a number of agent development tools have been reported and some are now commercially available. Java is employed to develop the architecture of present agent-based system. Each agent developed, in this research is based on the underlying framework of Java Agent Template Lite (JATLite), which is a prototype agent environment developed by Stanford's agent-based environment group at http://www.java.stanford.edu/. It is a set of lightweight Java packages that can be used to build multi-agent system. JATLite facilitates the construction of agent, which can send and receives the message using the emerging Stanford communication language KQML [43]. It provides the basic information in which an agent registers with an agent message router using a name and password to connect / disconnect from the internet, send and receives message, transfer files and invoke other programs or actions on the various computers where they are running.

The wide use of agent technology in industry depends upon the availability of development tools and a platform that protect developers from the need to develop basic functionality of each system. Such tools and platform, in turn, presume the existence of standards that reflect the agreement of developers on what basis the functionality should be presented. Some efforts have been devoted to provide standards for agent-based systems, but no accepted standard can be found for developing agent-based manufacturing systems. KQML is intended as a common communication language for agent with KIF [44] as a common content format. KQML as ACL is used here. Some traditional standards have also been used in agent-based system development, such as the common object request broker architecture (COBRA) for inter-agent communication and STEP [45] for providing the semantics of messages in manufacturing application. Currently, two consortia have focused on formalizing standards specifically to support agents. These are the Foundation for Intelligent Physical Agents (FIPA) and the National Industrial Information Infrastructure Protocols (NIIIP). FIPA establishes in 1996 as a worldwide consortium, promotes the development of a specification of generic agent technologies that maximizes the interoperability with in and across an agent-based application. FIPA has already produced version −1 of its set specification called FIPA 98. NIIIP is a consortium of US companies formed to develop open industry software protocols that will make it possible for manufacturers and their supplier to operate effectively in a collective way so that they were part of the same enterprise.

7 Experimental scenario

Experiments were conducted with the intention of analyzing the capability of proposed agent-based framework. Three experimental scenarios with different complexity have been presented to show the robustness of the proposed approach. For the first case, a test problem has been conceived to represent the layout of the AGV system, which maps the scenario described by [15]. In this scenario, the layout consists of 16 links, nodes represent a parking place of an idle vehicle from where a pick up and drop off load can be carried out to a respective work station or any other important areas like battery storage etc. in order to carry out the entire transportation assignment, three AGVs are used. The situation was simulated to find out the feasible, collision free path for third AGV from initial node 10 to goal node 4 via pickup (intermediate node) 9.

The second experiment has been conceived with increased complexity to determine the computational burden of the proposed framework. The simulation run was carried out for a model guidepath with increased number of nodes and links compared to that of previous one. The model guide path tested in the simulation is delineated in Fig. 10.

The specifications related to the guide path are described as follows:

- Number of links=186 (unidirectional)
- Number of links=372 (bidirectional)
- Number of vehicle=3
- Number of nodes=174
- Maximum vehicle speed V_{max}=1.0 m/s (for both directions)
- Acceleration α_{acc}=deceleration α_{dec}=1.0 m/s^2
- Velocity parameter (η)=1(straight link), 0.5 (curved link)

The main task here is to find the computational response time. In this case, motion planning has been decided for AGV 3, after the scheduling of AGV 1 and AGV 2. The path of AGV 1 was from node 3 to node 7, and AGV 2 is scheduled for station 4 to station 6. The task is to find the schedule of AGV 3 from start node to goal node such that conflict-free minimum time motion can be achieved. In this case, computational response time varies as the number of potential collision increase. Experimental condition has been simulated for various start and goal node to find out the worst case scenario of AGV 3.

For the third experiment, a guide path from real world industrial scenario has been modeled. In this scenario, robustness of the proposed approach is shown via increased efficiency of the AGV system. Hence, the important performance measure of this AGV system is the number of order completed. Deadlock occurrence and flowablity of system is also of prime interest. The layout is large in surface area and represents the scenario of a manufacturing plant. From the implementation point of view, some assumptions are made which are described as follows:

1. All AGVs are in the system and travelling to a load and unload station. There is no idle AGV at a workstation or home station.
2. There is always an order for AGVs to collect or drop.
3. Multiple pick-up and delivery are avoided. After completion of a delivery, a new order has been selected from predefined orders. There is no waiting time except the computational response time that the code takes to move from completion of an order to new one.
4. Buffer capacity of load stations is assumed to be infinite.
5. For illustration maximum velocity of AGV is assumed as 1 m/sec, and Velocity parameter (η)=1(straight link), 0.5 (curved link).

Initially, the numbers of AGVs were taken as six. The numbers of AGVs were decreased as the experiment proceeds. The last simulation was carried out for two AGVs. Numbers of AGV were decreased with a view to serve for fewer orders and reduced interference/collisions. The simulation carried out was compared with industrial material handling simulation package.

8 Results and discussions

Simulation run has been carried out for the three selected experimental scenario. The code was developed in Java. The PC used for this simulation is of 256 RAM and 256 MB of virtual RAM with 1.8 GHz and Windows XP environment.

For the first experimental scenario, optimal result is achieved. In this case {(10 9),(9 8), (8 7), (7 4)} is referred to as optimal path. OTC agent chooses this path to complete the associated task with it. This Path is also referred as conflict- and deadlock-free path. Hence, AGV 3 will start from initial node 10 pick up job at intermediate node 9 and dropping off the job to final node 4. Its total journey time was 161 seconds. The computational response time was .01 second. Hence a total of 161.01 second was used by AGV 3 to complete the given task.

Second experimental scenario was the example of a complex guide path. In this case, computational response times for various starts, and a goal node is determined for AGV 3. Table 2 represents the computation time required for motion planning of AGV 3 for various combination of start and goal node. It is evident from the table that for the worst case scenario, computational time of the proposed approach is .22 second. It reveals the fact that these times are well with in the range of the intervals of vehicles

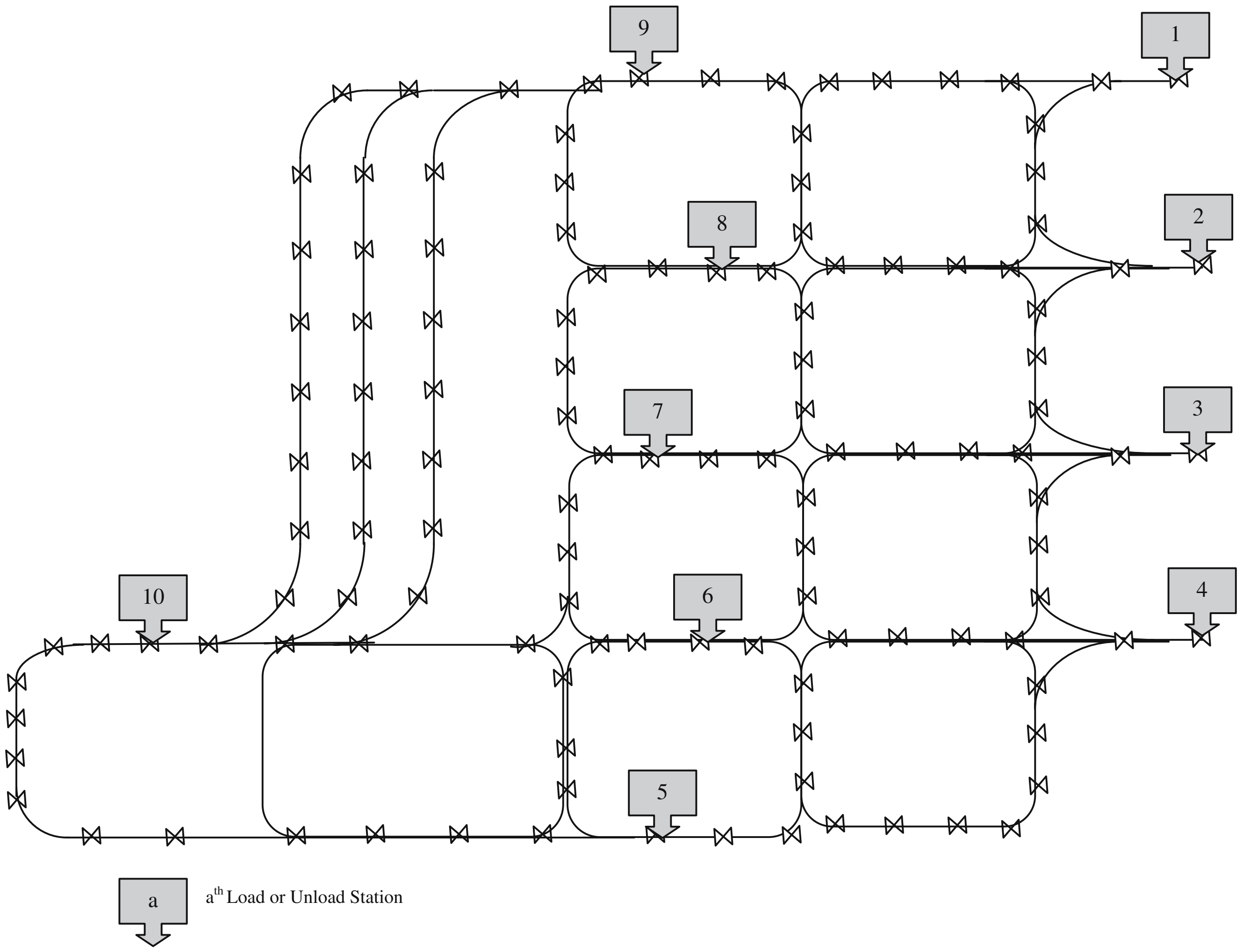

Fig. 10 Model guide path used in experimental scenario 2 to test the computational burden of proposed approach

request in the most practical multiple automated guided vehicle system (MAGVs). Hence, this can be applied to real world application.

For the third scenario, a comparative study is made on the basis of efficiency of the system. For the present purpose, efficiency may be defined as number of deliveries per hour. A real comparison between the control system used in industry and proposed AGV framework is not feasible; hence, a simulation package (AutoMod) is used to simulate the performance of the actual AGV system used in the industry while comparing it with proposed approach. It is found that for the given layout, deadlock and collision occur more frequently using the simulation package.

Table 2 Computational response time for various start and goal node

Start station	Goal station	Computational response time (Sec)
Station 1	Station 10	0.14
Station 2	Station 10	0.12
Station 3	Station 10	0.22
Station 4	Station 10	0.18
Station 10	Station 1	0.12
Station 10	Station 2	0.12
Station 10	Station 3	0.1
Station 10	Station 4	0.14
Station 5	Station 9	0.16
Station 6	Station 1	0.08
Station 5	Station 3	0.11

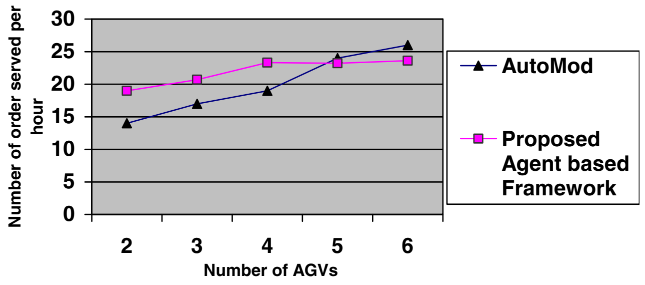

Fig. 11 Comparative study of number of order served/hour for given layout

Table 3 Deadlock formation rate with regard to the number of AGVs used in real system

Number of AGVs	Number of order served/hour	Number of deadlocks predicted	Deadlock formation rate
2	14	8	0.57
3	17	14	0.82
4	19	24	1.26
5	24	32	1.32
6	26	39	1.5

Whereas, simulating with proposed agent-based controller, deadlock- and collision-free minimum time motion is achieved. Figure 11 represents the comparative study of the number of deliveries/hour using these two controllers. It is evident from the Fig. 11 that the number of orders served using proposed controller for less number of AGVs (up to 5) is more. As number of vehicle increases it remains nearly same. This is due to the fact that as time passes deadlocks were removed in practical application. However, working of the proposed approach on such a complex guide path in real world application is considerable success. It is seen from the graph that lower number of AGVs result in higher rate of deliveries and conflict-free path, where as deadlock occurs frequently in the real system. Table 2 shows the deadlock occurrence rate in the real system.

Deadlock occurrence rate is the ratio of number of deadlocks predicted and the number of orders served. This is the systematic measure of conflict resolution.

Tables 3 and 4

9 Conclusion and future scope

The main aim of this research is to resolve some of the complex issues concerning the collision and deadlock avoidance with minimum time motion planning pertaining to operational control of AGVs in an FMS. This paper presents an multi-agent-based framework representing zone controlled AGV environment incorporating various issues like path generation, link occupation time, collision and deadlock avoidance, suggests the waiting node and time estimation, positioning of the idle AGVs, identification of pickup and drop off nodes associated with an optimal and conflict-free path selection. How intelligent agent-based framework can be used to overcome the shortcoming associated with current approach have been discussed. Six types of agents have been proposed; each agent is associated with some rule base and algorithms. SFP and KSFP algorithms are used by GP agent to generate K shortest path. JTD agent generates LOT. The ZC agent takes care of collision and deadlock avoidance. OTC agent acts as a decision-maker and determines the overall motion planning of the system. Order agent takes care of accepting and rejecting an order or to generate virtual order. AGVs are managed by an AGV agent. In order to show the robustness of the proposed intelligent agent-based framework, three experimental scenarios with increased complexity are considered. Simulation result shows that proposed framework provides an optimal path, less computational burden and higher efficiency. Hence a proposed agent-based controller could be a viable alternative to a large and/ or complex and difficult to design AGV system. A rule-based system is advocated to address the continuous

Table 4 Input and output sequence dataflow in MAS

Step	Dataflow	Message type	Performative	Operation
1	OA-IA	Demand for transportation of supplies	Request	Information of transportation requirement goes into the multi-agent system
2	IA-GPA	Request for present situation of AGVs	Request	Request to AGVs to provide the information of its location and status
3	GPA-AGVA	AGVs information	Update_info	Collect the information about present location of all AGVs
4	IA-JTDA	Manipulation of link occupation time of each AGVs	Manipulate	Calculation of link occupation time as shown in Fig. 7.
5	IA-ZCA	Trajectory planning of AGVs	Trajectory_plan	Trajectory planning of all feasible paths
6	IA-OTCA	Decision of shortest feasible path	Shortest_path	Evaluation of shortest and collision- free path
7	OTCA-AGVA	AGV traffic control	Control	Communicate with AGVs at crossing of each and every node
8	AGVA-OTCA	Fault occurrence	Agv_fault	When a breakdown occurs, AGV agent informs the OTCA
9	OTCA-shop-floor controller	Inform to shop-floor controller for fault recovery	Fault_recovery	Report to the shop-floor controller
10	Shop-floor controller instructs the order agent	Transportation information about loading stations to target station		Shop-floor controller instructs the order agent to reschedule the transportation plan for supplies (which could not be reach its destination)

routing of AGV, while negotiating unexpected interruptions on certain edges (like mechanical failure of any movable AGV). Manual operator is adopted on these occasions to counter any failure in its progress.

We propose few future research areas also. It is expected that the control approach proposed here could also be applicable in another context like a transportation network. A developing agent-oriented framework with automated rule generation using an evolutionary neuro-fuzzy system to negotiate the conflict and interruption related to operating control of AGV, increasing the efficiency of the proposed approach for larger number of AGVs in FMS.

References

1. Stecke KE, Solberg JJ (1983) Loading and control policies for a flexible manufacturing system. Int J Prod Res 19(5):481–490
2. Maxwell WL, Muckstatd JA (1982) Design of automated guided vehicle systems. IIE Trans 14:114–124
3. Jennings NR, Wooldridges M (1995) Applying agent technology. Appl Artif Intell 9:357–369
4. Tanchoco JMA, Sinriech D (1992) Osl-optimal single-loop guide paths for AGVS. Int J Prod Res 30:665–681
5. Egbelu PJ, Tanchoco JMA (1986) Potential fore bidirectional guide path for an automated guided vehicles based systems. Int J Prod Res 24:1075–1097
6. Egbelu PJ (1987) The use of non simulation approaches in estimating vehicle requirements in an automated guided vehicles based transport system. Mater Flow 4:17–32
7. Viswanadham N, Narahari Y, Johnson TL (1990) Deadlock prevention deadlock avoidance in flexible manufacturing system using Petrinet models. IEEE Trans Robot Autom 6 (6):713–723
8. Kumar RR, Singh AK, Tiwari MK (2004) A fuzzy based algorithm to solve the machine loading problems of an FMS and its neuro fuzzy petri net model. Int J Adv Manuf Technol 23(2–3):318–341
9. Rajotia S, Shanker K, Batra JL (1998) A semi-dynamic time window constrained routing strategy in an AGV system. Int J Prod Res 36(1):35–50
10. Lee CC, Lin JT (1995) Deadlock prediction and avoidance based on Petri nets for zone-control automated guided vehicle systems. Int J Prod Res 33:3249–3265
11. Reveliotis SA (2000) Conflict resolution in AGV systems. IIE Trans 32:647–659
12. Broadbent AJ, Besant CB, Premi SK, Walker SP (1985) Free ranging AGV systems: promises, problems and pathways, problems and pathways. Proceedings of the 2nd International Conference on Automated Material Handling UK, pp 221–237
13. Tagaboni, F, Tanchoco JMA (1988) A LISP-based controller for free ranging automated guided vehicle systems. Int J Prod Res 26:173–188
14. Chang WK, Tanchoco JMA (1991) Conflict-free shortest time bidirectional AGV routing. Int J Prod Res 29:2377–2391
15. Narshimhan R, Batta R, Karwan M (1998) Routing automated guided vehicles in the presence of interruption. Int J Prod Res 37:653–681
16. Oboth CR, Batta R, Karwan M (1999) Dynamic conflict-free routing of automated guided vehicle. Int J Prod Res 37:2003–2030
17. Choi KH, Kim SC, Yook SH (2000) Multi-agent hybrid shop floor control system. Int J Prod Res 38:4193–4203
18. Lu TP, Yih H (2001) An agent based production control framework for multiple line collaborative manufacturing. Int J Prod Res 39:2155–2176
19. Wallace A (2001) Application of AI to AGV control agent: agent control of AGVs. Int J Prod Res 39:709–726
20. Lim JK, Lim JM, Yoshimoto K, Kim KH, Takahashi T (2002) A construction algorithm for designing guide paths of automated guided vehicle systems. Int J Prod Res 40(15):3981–3994
21. Fanti MP (2002) Event-based controller to avoid deadlock and collision in zone control AGVS. Int J Prod Res 40(6):1453–1478
22. Lee JH, Lee BH, Choi MH (1998) A real time traffic control scheme of multiple AGV systems for collision free minimum time motion: a routing table approach. IEEE Trans Syst Man Cybern, Part A, Syst Humans 28(3):347–358
23. Kim BI, Graves RJ, Heragu SS, Onge AS (2002) Intelligent agent modeling of an industrial warehousing problem. IIE Trans 34:601–612
24. Uzam M (2004) The use of Petrinet reduction approachfor an optimal deadlock prevention policy for FMS. Int J Adv Manuf Technol 23(3–4):204–220
25. Huang J, Palekar US, Kapoor SG (1997) A labeling algorithm for the navigation of automated guided vehicles. Trans ASME J Eng Ind 115:315–321, Aug 1997
26. Miller RK (1987) Automated guided vehicles and automated manufacturing. Dearborn, MI, Soc Manufact Eng
27. Fisher M (1994) Representing and executing agent based systems. In: Proceedings of the ECAI'94 Workshop on agent theories. Architecture and languages. Springer, Berlin Heidelberg New York, pp 307–323
28. Jennings NR, Wooldridge M (1998) Application of intelligent agents. In: Jennings NR, Wooldridge MJ (eds) Agent technology foundation, applications and markets. Springer, Berlin Heidelberg New York, pp 3–28
29. Davidsson P, Astor E, Ekdah LB (1994) A framework for autonomous agent based on the concept of anticipatory systems. Proceedings of cybernetics and systems'94, vol. II. World Scientific, Singapore, pp 1427–1434
30. Maes P (1995) Modeling adaptive autonomous agents. In: Langton CG (ed) Artificial life: an overview. MIT Press, Cambridge, MA, pp 135–162
31. Nwana HS, Ndumu DT (1997) An introduction to agent technology in software agents and soft computing. In: Nwana HS, Azami N (eds) Towards enhancing machine intelligence. Springer, Berlin Heidelberg New York, pp 3–26
32. Huang C-Y, Nof SY (2000) Formation of autonomous agent networks for manufacturing systems. Int J Prod Res 38(3):607–624
33. Wooldridge M, Jennings NR (1995) Intelligent agents: theory and practice. Knowl Eng Rev 10:115–152
34. Jennings NR, Wooldridge M (2000) Agent-oriented software engineering. Handbook of agent technology. AAAI/MIT Press
35. Lou P, Zhou Z, Chan YP, Xi W (2004) Study on multi-agent based. Agile Supply Chain 23(3–4):197–204
36. Singh SP, Tiwari MK (2002) Intelligent agent framework to determine the optimal conflict-free path for an automated guided vehicle system. Int J Prod Res 40(16):4195–4223
37. Mondal S, Tiwari MK (2002) Application of autonomous agent network to support the architecture of a Holonic manufacturing system. Int J Adv Manuf Technol 20:931– 942
38. Mondal S, Tiwari MK (2003) Formulation of mobile agents for integration of supply chain using the KLAIM concept. Int J Prod Res 41(1):97–119

39. Pierce AR (1975) Bibliography on algorithms for shortest path, shortest spanning tree, and related circuit routing problems (1956–1974). Networks 5:129–149
40. Dung DA, Grover WD, MacGregor MH (1994) Comparison of k-shortest paths and maximum flow routing for network facility restoration. IEEE J Sel Areas Commun 12(1):88–99
41. Katoh N, Ibaraki T, Mine H (1982) An efficient algorithm for K shortest simple paths. Networks 12:411–427
42. Topkis DM (1988) A K shortest path algorithm for adaptive routing in communications networks. IEEE Trans Commun 36 (7):855–859
43. Finin T, Fritzon R, McKay D, McEntire R (1993), KQML language and protocol for knowledge and information exchange. Technical Report, University of Maryland, Baltimore
44. Genesereth MR, Fikes RE (1992) Knowledge interchange format. Version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Palo Alto, CA [http://www.cs.umbc.edu/agents/kse/kif]
45. Bjork B, Wix J (1991) An introduction to STEP. Technical Report, VTT Technical Research Center of Finland and WixMe-Lelland Ltd, UK