Communication-free Detection of Resource Conflicts in Multi-Agent-based Cyber-Physical Systems

Nguyen-Thinh Le*, Lukas Märtin†, Christopher Mumme*, Niels Pinkwart*
*Department of Informatics
Clausthal University of Technology, Germany
Email: {nguyen-thinh.le, christopher.mumme, niels.pinkwart}@tu-clausthal.de
†Institute for Programming and Reactive Systems
TU Braunschweig, Germany
Email: l.maertin@tu-bs.de

Abstract—**Multi-agent approaches can be applied to model behaviour and relations of entities in cyber-physical systems. Here entities frequently compete on insufficient resources (e.g., hardware) at the same time. Hence, resource conflicts between several agents are one of the most important conflict types in such multi-agent systems. These conflicts can significantly slow the operation of a system down, or in the worst case, might lead to a system halt. In this paper, we investigate the challenge of efficiently detecting resource conflicts. For this purpose, we introduce a conflict detection model based on beliefs of BDI agents. One benefit of our approach is that conflicts are detected using local belief state information of agents without communication. For evaluation purposes we apply our conflict detection model to a multi-agent system representing a transportation service with moving robots on a fictitious airport to measure the rate of collisions and completed transportation tasks. The evaluation study showed that the system deploying the conflict detection model can avoid collisions between moving agents and agents execute tasks successfully.**

conflict detection, resource conflicts, multi-agent systems, BDI agents, cyber-physical systems

I. Introduction

In multi-agent systems, conflicts between agents on resources are often unavoidable. This is due to the inherent characteristics of cyber-physical systems that include heterogeneity and autonomy of the entities on the one hand and insufficiency of (hardware) resources available in the system on the other hand. Resource conflicts represent one of typical conflicts in such systems and occur when two or more autonomous agents compete on the same resource at the same time. In this paper we consider conflicts of this type.

In order to ensure reliable and effective operation of systems, potential conflicts between agents in an environment need to be handled. A prior to conflict resolution, conflicts need to be detected and analysed. Thus, an appropriate formalism for modeling and detecting conflicts is necessary.

This paper proposes a conflict model to detect potential resource conflicts between agents and a mechanism of conflict detection based on belief state information of Belief Desire Intention (BDI) agents [9] in a multi-agent system. Our conflict model assumes that agents cannot perceive the whole environment at once, because the view range of each agent is restricted. Hence, each agent has a unique local belief. The model has been developed for agent environments which have a predefined, fixed road grid without specific traffic rules, supervision or global coordination.

The conflict model has been evaluated to test the following hypothesis: A multi-agent system which deploys the resource conflict model will execute given tasks more successful than a system without it.

In Section II, we review briefly the notion *conflict* and existing approaches for conflict modelling. Subsequently, in Section III we propose our conflict model and describe its application to detect conflicts within an environment of a multi-agent system. Then, in Section IV, we present a case study from the NTH Focused Research School for IT Ecosystems[1]: a transportation service at an fictitious airport. In Section V, the implementation of a multi-agent system which deploys the proposed conflict model to the case study is described. The conflict model is evaluated in Section VI. We discuss the limitations and advantages of the model in Section VII. In the last section, we summarize our conclusions and propose future work.

II. Related Work

To classify conflicts in multi-agent systems, several authors distinguish between action, plan and goal conflicts [7], [8], [10]. An action conflict arises on the level of individual actions. For the execution of actions certain resources are needed, which are unavailable in sufficient quantity or with exclusive access. In this process, plans and goals do not stand in conflict. A plan conflict arises if plans of multiple agents contain action-related conflicts. When goals of agents lead to plan conflicts, at least one conflicting goal exists. Hence, the goals of a group of agents are not compatible and only one goal is executable.

[1]http://www.it-ecosystems.org

Furthermore, conflicts in multi-agent systems can be categorized into physical and epistemic conflicts. Physical conflicts occur when agents' interests are affected by insufficient resources, which are not shareable at the same time (e.g., a road element) [11]. Thus, physical conflicts can also be referred to as resource conflicts. Epistemic conflicts, where agents' have different interpretations of a context and their desires, are called knowledge conflicts [7]. These conflicts results from differences in locations, sensory systems, or general skills of the agents (e.g., addictions).

Common approaches for detecting and solving conflicts use communication to deal with sharing knowledge between agents. In this way, identical information of belief bases from several agents are combined to a union of all individual interpretations of the same context. The belief bases are subsequently shared by agent-to-agent communication (n:m) and the determination of a compromise is done by each agent separately. Thangarajah et al. [12] sum up information about resources of multiple agents to detect and avoid resource conflicts. This resource summary process depends significantly on communication structures. The authors in [1] exchange intended goal structures (IGSs) to eliminate goal conflicts between agents. Furthermore, they use pattern matching to detect plan conflicts and conflicting conditions (contradicting pre-/postconditions of oppositional actions). The integration of divergent plans is done by plan integration through merging individual E-PERT diagrams. The work of [5] is related to actual merging of several belief bases with integrity constraints. The exchange of beliefs is also done by direct communication. All of these approaches need communication in the phase of conflict detection. Depending on the number of agents in a system, efforts for communication and determination might increase intensively.

Our approach for conflict detection, does not require communication between agents. Instead, agents individually build up the belief about their environment by perception. Hence, an agent perceives all objects in its environment within its view range and based on this belief information, the agent is able to determine potential resource conflicts if other agents constantly move towards the same resource. Thus, our approach significantly reduces communication effort.

III. Conflict Model

To handle resource conflicts between multiple entities in a multi-agent system, a formal computational conflict model is required. Here, we assume that the environment, in which agents execute their tasks, describes a 2-dimensional grid-based world (x, y coordinates).

Let the environment be described by a triple of three finite non-empty sets: 1) a set of agents α=$\{A_1, \dots, A_n\}$ existing in the environment, 2) a set of environment elements, and 3) a set of resource types β=$\{R_1, \dots, R_n\}$, where elements of an environment are defined formally as follows:

Definition 1 (Env. Element): Given an environment in which agents move, an environment element E is a tuple $\langle X, Y, R, C, \alpha_E, T\rangle$ where

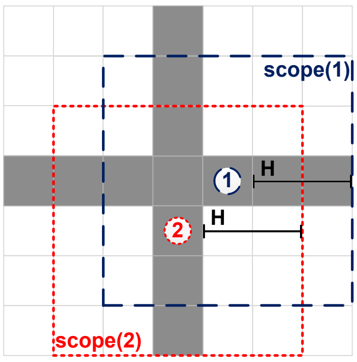


Fig. 1. Scopes of Agents

- X and Y represent the coordinates of the element within the environment,
- $R \in \beta$,
- $C \in \mathbb{N}$ indicates the capacity of this environment element,
- $\alpha_E \subseteq \alpha$ are agents occupying this environment element, and
- T is the time when this environment element is observed.

According to Definition 1, the occupation of an environment element E at the observed time T is $|\alpha_E|$. Each agent is capable to see its peers within its view range, therefore, we define the scope of an agent formally as follows.

Definition 2 (Scope): A scope of an agent A is a set of environment elements which A can see within its visibility H: $scope(A)$=$\{\langle X_i, Y_i, R, \alpha_{E,i}, T\rangle|\ X - H \le X_i \le X + H, Y - H \le Y_i \le Y + H,\ 0 \le i \le (2H+1)^2$, X and Y are the coordinates of the agent $A\}$.

We assume that the scope of agent A is a square whose length is $2H+1$, that is because the view range (horizon) can be stretched to north/south (or west/east) including the agent's position. Thus, the length of $scope(A)$ is $(2H+1)^2$. Fig. 1 shows two agents at a crossing with common horizon $H = 2$.

Each agent has a belief about the last, current, and next possible position of other agents existing within its scope.

Definition 3 (Belief): The belief of an agent A about other agents in its scope is defined by: $belief(A) = \{\langle A_i, last_X, last_Y, current_X, current_Y, next_X, next_Y\rangle | A_i \in \alpha_E, E \in scope(A)\}$, where $last_{X/Y}$, $current_{X/Y}$, and $next_{X/Y}$ are the last, current, and next positions of agent A_i.

The next position of each agent A_i within the scope of A is calculated using information about the current and the last position of that agent. The calculation of the next position is based on the assumption that when an agent is moving, then it is probable that this agent will move straight forward. Let $last_X(A_i)$ and $current_X(A_i)$ be the last and the current X coordinate of A_i, the difference between the last and the current X coordinate of A_i is $diff_X(A_i) = last_X(A_i) - current_X(A_i)$. As a result, the next X coordinate of A_i will be $next_X(A_i) = current_X(A_i) - diff_X(A_i)$. Similarly, the next Y coordinate of A_i can be determined. The next possible position of A_i is composed of the next X and Y coordinates of

A_i. Note that the horizon H of agent A must have a minimum size of 2, because A needs to see both the last and the current position of another agent within its scope.

To enable the calculation of next possible positions of other agents within the scope of agent A, the belief of A about the current and last positions of these agents needs to be updated dynamically. For this purpose, A may use a collection to keep trace of known agents, i.e, that have been once in the scope of A. If an agent in this collection has changed its position, its last and current position will be updated accordingly. Is an agent A_i in the scope of A at the first time, then A_i will be added to the collection of known agents and the last position of A_i will become the current position ($\mathit{diff}_X(A_i) = 0$).

Based on the belief about the next possible position of other agents and the agent's scope, an agent A is able to identify the agents that will release/require an environment element which is also required by A. Such a potential conflict for A is defined formally as follows.

Definition 4 (Potential Conflict): Let A be an agent, its current position is $\langle X, Y\rangle$ and its next action is to require an environment element E at position $\langle next_X, next_Y\rangle$. Let α_E be the set of agents that are occupying E, $\alpha_{release,E}$ and $\alpha_{require,E}$ be the set of agents (excluding A) that will release/require E, respectively, A has a potential conflict, denoted as $conflict(E, scope(A), \alpha_E, \alpha_{release,E}, \alpha_{require,E})$, iff $|\alpha_E| - |\alpha_{release,E}| + |\alpha_{require,E}| + 1 > C$, where C is the capacity of E.

Since A also requires the environment element being observed, the condition for checking potential conflicts must include the value 1 on the left hand-side of the comparative term: $|\alpha_E| - |\alpha_{release,E}| + |\alpha_{require,E}| + 1 > C$.

If the sets of agents which are (1) occupying, (2) releasing, and (3) requiring an environment element being observed can be determined, the potential conflict of consuming this environment element is calculated by applying Definition 4.

IV. Case study: Smart Airport

In order to illustrate the conflict model proposed in this paper, we use an airport departure transportation scenario as a representative for a cyber-physical system of moving robots on a fixed grid. The airport consists of static objects (roads, entrances, check-in counters, plane parking positions, and charging stations) and moving objects (autonomous transportation vehicles (ATVs)). When there is a request of a passenger to be transported, an agency will provide ATVs and manage the passenger's order. A transportation order consists of start and end positions, pickup time, and a latest time for drop-off. The start and end positions represent a route, e.g., from an entrance to a check-in counter. Since ATVs need energy to move, they are equipped with batteries which need to recharge regularly at charging stations. In this scenario, typical resource conflict situations are:

1) At least two ATVs are moving on a road element. One of the ATVs needs the priority to occupy the road element first. The insufficient resource is the road element.
2) Several ATVs are running out of energy and need to be recharged, while the charging station might be occupied. The resource required by ATVs is the charging station.
3) ATVs have to take passengers to unoccupied check-in counters. The resource is a check-in counter. Here, a check-in counter is manned by one or two staff members, and thus has a maximal capacity of two units.

This application scenario of a cyber-physical system can be described by a multi-agent system in which ATVs are implemented as autonomous agents. To enrich agents with intelligent behaviour, we choose a BDI architecture [4]. Agents have a set of goals (*Desires*) which they pursue based on the current world state (*Belief*). Furthermore, agents possess a plan database from which they can select plans (*Intentions*) that lead them to goals. A plan consists of a set of atomic actions which are achieved simultaneously or successively. In this scenario, an action is moving one road element further.

V. Implementation

Besides a simplified hardware realisation with NXT-robots, we implemented the airport scenario using the JRep simulation platform [3] for analysis proposes. JRep is an integration of Repast Symphony and the JADE Framework. Repast provides a tool kit for visual simulations of multi-agent systems. JADE supports developing intelligent behaviours for agents and provides communication protocols according to FIPA-ACL. The capacity for each environment element in this simulation platform can be parametrised. Since Repast supports thread-based simulation, each simulation step underlies a time tick. Here, one tick represents a movement and an agent may need several ticks to perform a task (i.e., to move from a start position to an end position). For each task, a shortest path is calculated based on Dijkstra's algorithm. Repast provides an implementation library of this algorithm. There are cases where no shortest path can be determined. For example, when several agents are moving forward and encountering a queue of agents moving in the opposite direction. Fig. 2 shows this situation. Agents 2 and 3 are moving to the right while other agents are moving in another direction. For agents 2, 3, and 4 no shortest path can be calculated because they are blocked by agents 1 and 5. In this case, agents 2, 3, and 4 have to stay on the same road field until they have a free way to move.

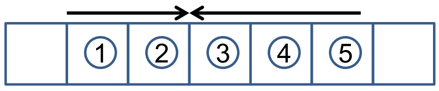


Fig. 2. No shortest path available

Another case is when two agents have two paths whose distance is equal and the two paths have at least one destination in common. Fig. 3 illustrates this problem. Agent1's and Agent2's destination positions are F1 and F2, respectively (see Fig. 3a). Agent1 and Agent2 have a conflict because they have to move in the opposite direction (see Fig 3b). Then, the direction, which leads to a conflict, is removed from the agent's map (marked as crosses) and two new shortest path

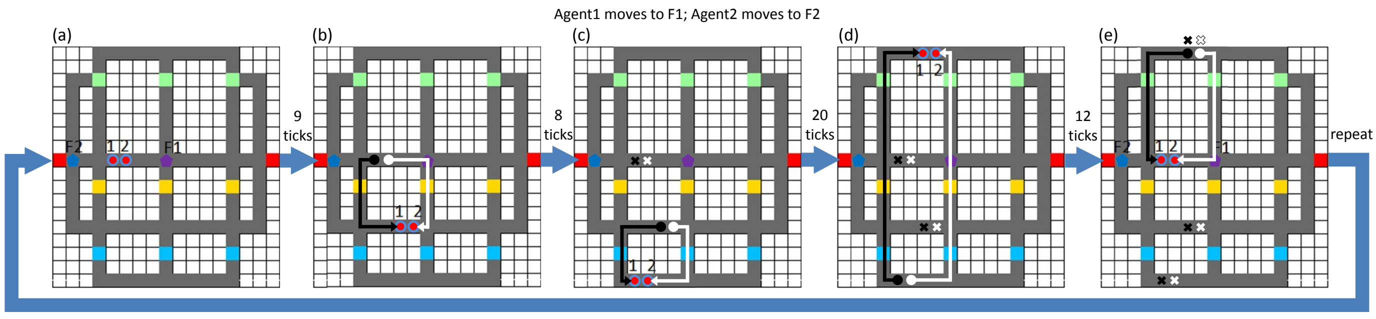
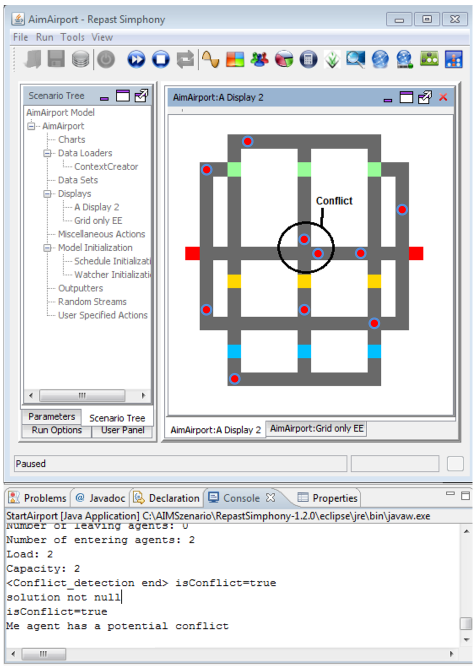


Fig. 3. No shortest path available

are calculated (see Fig. 3c). Again, these two routes lead to a conflict and new two paths have to be generated (see Fig. 3d and Fig. 3e). As result, Agent1 and Agent2 move back to the original positions (see Fig. 3a). Hence, the process of detecting conflict and choosing a path is non-deterministic in this case. To solve this problem, we propose to choose one of the agents randomly and to delay the movement of this agent. While an agent is delayed and renew its map (because on the current map, all conflicting positions have been marked as not moveable), another agent can move along its path.

Our framework uses a scheduler to iterates all agents to perform three steps: scanning an environment to build up belief, detecting conflicts, and moving. The simulation of a conflict situation at a crossing in the grid is shown in Fig. 4.



```
StartAirport [Java Application] C:\AIMSzenario\RepastSimphony-1.2.0\eclipse\jre\bin\javaw.exe
Number of leaving agents: 0
Number of entering agents: 2
Load: 2
Capacity: 2
<Conflict_detection end> isConflict=true
solution not null
isConflict=true
Me agent has a potential conflict
```

Fig. 4. The simulation framework

VI. Evaluation

The goal of the evaluation is to prove that the conflict detection mechanism is beneficial for multi-agent systems. For this purpose, we evaluate the hypothesis mentioned in Section I: A multi-agent system which deploys the proposed conflict model will execute given tasks more successful than a system without it.

A. *Design*

To carry out the evaluation study, we use the simulation framework described in the previous section. Since we intended to provoke resource conflicts on roads, the capacity of road elements is set with the value 1. The simulation framework is used in two versions: one deployed the proposed conflict model and another one did not. In the first version, every time when an agent notices a resource conflict, it looks for a new shortest path leading to the destination position specified for its task. Using this version, all tasks assigned to agents will be executed. However, agents will need a lot of time to execute their tasks because they have to calculate a new route whenever they encounter a potential resource conflict. In the latter version, agents do not have the capability to detect a resource conflict. Thus, they can move to an environment element, i.e., consume a resource, without noticing whether the capacity of that field allows it. In case the capacity of an environment element is overloaded, the system kills every agents occupying that environment element. That is, tasks assigned to these agents can not be executed any more. The evaluation will investigate how many tasks can be executed successfully using the version without the resource conflict detection and how much time agents need to execute all tasks. For this purpose, the two system versions are parametrized with 10, 20, and 30 agents in order to find out how the number of agents impact on the behaviour of the whole system. For each parametrization, the two versions are simulated five times to determine time required to finish tasks in average, because the tasks of agents (i.e., moving from a position to another one in a smart airport) are created randomly for each simulation.

B. *Results*

TABLE I
With conflict detection: Time (in ticks) required to complete agents' tasks

#Agents	Test1	Test2	Test3	Test4	Test5	Avg.(s.d.)
10	37	96	93	39	55	64(28.7)
20	97	95	95	74	81	88.4(10.3)
30	155	146	129	154	213	159.4(31.7)

TABLE II
WITHOUT CONFLICT DETECTION: TIME (IN TICKS) REQUIRED TO COMPLETE AGENTS' TASKS

#Agents	Test1	Test2	Test3	Test4	Test5	Avg.(s.d.)
10	18	23	21	22	18	20.4(2.3)
20	19	25	21	27	24	23.2(3.2)
30	22	18	24	21	24	21.8(2.5)

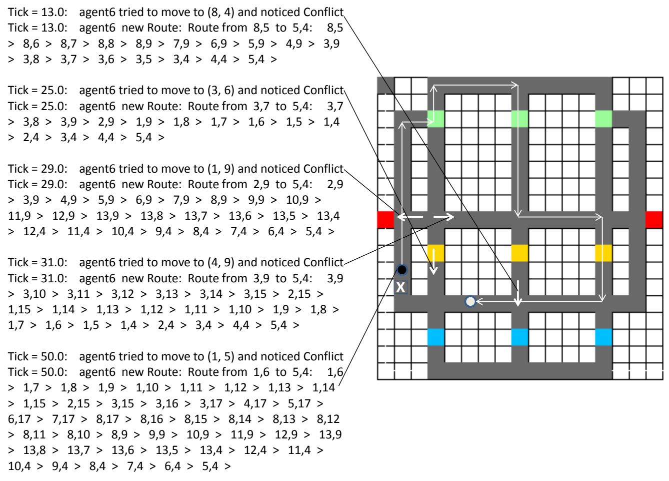
Table I and Table II show statistical results of five simulations using the framework version with and without conflict detection, respectively. From the seventh column of these tables, we can notice that using the version with conflict detection, the time required to execute all tasks increases (from 64 to 159) with the number of simulated agents, while using the version without conflict detection, the time required for executing tasks remains constant (21 ticks in average) between the variation of agents. This can be explained by the fact that with more agents more resources conflicts have been detected, and when a conflict is detected a new route for executing a given task is created. Even on a new route, conflicts still can happen. Using the version without conflict detection, no resource conflicts can be detected, agents are killed when conflicts occur.

From Table III we notice that the percentage of successfully executed tasks decreases (72% to 43%) with an increasing number of agents. In average, about 59% of assigned tasks can be executed successfully using a simulation without conflict detection. This result can confirm our hypothesis that deploying conflict detection more tasks are executed successfully (but more time is required) than without using it. Note, with conflict detection 100% of the tasks have been executed successfully.

TABLE III
WITHOUT CONFLICT DETECTION: SUCCESSFUL TASKS

#Agents	Test1	Test2	Test3	Test4	Test5	Avg.(s.d.)
10	6	8	6	8	8	72%(1.1)
20	11	12	14	10	14	61%(1.8)
30	13	10	8	15	18	43%(4.0)

Exceptionally, Table I shows in the first row that the standard deviation of time required to execute ten tasks of ten agents is high. This is caused by the second and the third simulation runs in which an agent had to create a new long route after noticing a conflict. Fig. 5 illustrates this problem. At Tick 13, Agent6 has noticed a conflict when trying to move from position (8,5) to (8,4), at Tick 25 from position (3,7) to (3,6), at Tick 29 from position (2,9) to (1,9), and at Tick 31 from position (3,9) to (4,9) (these directions are represented as short arrows in the figure). As a result, this agent cannot create a route which leads through these directions any more. At Tick 50, Agent6 is occupying the environment element of position (1,6) and wants to move from that position to (5,4). Agent6 has to choose a long route between position (1,6) and (5,4) (this route is indicated by the white line with arrows in the figure), although the distance between these positions is short.



```
Tick = 13.0:   agent6 tried to move to (8, 4) and noticed Conflict
Tick = 13.0:   agent6  new Route:  Route from  8,5  to  5,4:    8,5
>  8,6 >  8,7 >  8,8 >  8,9 >  7,9 >  6,9 >  5,9 >  4,9 >  3,9
>  3,8 >  3,7 >  3,6 >  3,5 >  3,4 >  4,4 >  5,4 >

Tick = 25.0:   agent6 tried to move to (3, 6) and noticed Conflict
Tick = 25.0:   agent6  new Route:  Route from  3,7  to  5,4:    3,7
>  3,8 >  3,9 >  2,9 >  1,9 >  1,8 >  1,7 >  1,6 >  1,5 >  1,4
>  2,4 >  3,4 >  4,4 >  5,4 >

Tick = 29.0:   agent6 tried to move to (1, 9) and noticed Conflict
Tick = 29.0:   agent6  new Route:  Route from  2,9  to  5,4:    2,9
>  3,9 >  4,9 >  5,9 >  6,9 >  7,9 >  8,9 >  9,9 >  10,9 >
11,9 >  12,9 >  13,9 >  13,8 >  13,7 >  13,6 >  13,5 >  13,4
>  12,4 >  11,4 >  10,4 >  9,4 >  8,4 >  7,4 >  6,4 >  5,4 >

Tick = 31.0:   agent6 tried to move to (4, 9) and noticed Conflict
Tick = 31.0:   agent6  new Route:  Route from  3,9  to  5,4:    3,9
>  3,10 >  3,11 >  3,12 >  3,13 >  3,14 >  3,15 >  2,15 >
1,15 >  1,14 >  1,13 >  1,12 >  1,11 >  1,10 >  1,9 >  1,8 >
1,7 >  1,6 >  1,5 >  1,4 >  2,4 >  3,4 >  4,4 >  5,4 >

Tick = 50.0:   agent6 tried to move to (1, 5) and noticed Conflict
Tick = 50.0:   agent6  new Route:  Route from  1,6  to  5,4:    1,6
>  1,7 >  1,8 >  1,9 >  1,10 >  1,11 >  1,12 >  1,13 >  1,14
>  1,15 >  2,15 >  3,15 >  3,16 >  3,17 >  4,17 >  5,17 >
6,17 >  7,17 >  8,17 >  8,16 >  8,15 >  8,14 >  8,13 >  8,12
>  8,11 >  8,10 >  8,9 >  9,9 >  10,9 >  11,9 >  12,9 >  13,9
>  13,8 >  13,7 >  13,6 >  13,5 >  13,4 >  12,4 >  11,4 >
10,4 >  9,4 >  8,4 >  7,4 >  6,4 >  5,4 >
```

Fig. 5. An agent has to choose a long route due to encountering a prior resource conflicts

VII. DISCUSSION

Although the conflict detection approach proposed in this paper has been shown to be effective, it has several limitations.

In the previous section, we could determine that the time required to complete all tasks in a multi-agent system which deploys the resource conflict model increases with the number of agents. This can be critical if the tasks given to agents are limited in a time frame. For example, if the transportation tasks in the airport scenario are limited within 100 seconds and we assume that each tick represents a second (the tick delay can be parametrised), then not all agents in the simulations with 30 agents would execute their tasks successfully. This phenomenon can be compared with the reality of traffic situation. If there are many vehicles, traffic could become full and hinder the drivers to reach a destination on time. Hence, we have to made a trade off between using the conflict detection model and the number of agents.

The proposed conflict detection model will be problematic if the simulation framework starts with two agents that want to move to the same road element (see Fig. 6). The conflict detection model requires at least two ticks to determine the last and the current position of other peer agents. Thus, at the first tick (when the system starts), the system is not able to detect resource conflicts. As a consequence, two agents from the opposite direction move into the same road element, which results in a collision. In order to work around with this problem, we need an additional mechanism. In every tick, the system checks whether a road element exceeds its capacity.

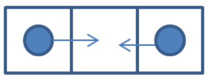


Fig. 6. Two agents move to the same road field at the system's start

The second problematic situation of this conflict model is that the calculation of the next position is based on the assumption that when an agent is moving, then it is probable that this agent will move straight forward. This assumption can result in a position which lies outside the environment as Fig. 7 shows. The dotted square is the field for the next possible position calculated using the conflict model. However, the real path of the agent is to move up. To fix this problem, we need an additional check to determine whether the calculated next position is an element outside of the environment.

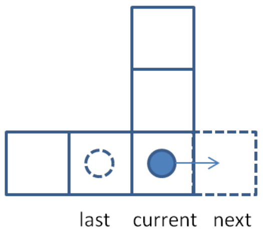


Fig. 7. Next possible position is outside the environment

In case that an agent A detects a potential conflict, but the belief of A is not correct, e.g., A_i moves to another position which is not in accordance with the belief of A, then the worst case is that A has to wait for a time tick until its belief is updated.

At the moment we do not consider the erosion of resources by consumption. When a resource is losing quality over time, the order of its assignment will become more important. Hence, erosion might lead to new conflicts and we have to extend our model for dynamic resource states (more than free or occupied).

Despite of its limitations, the proposed conflict model is advantageous. Most current work on conflicts in multi-agent systems proposed approaches of using communication between agents and focused on resolving conflicts. This normally requires a hard-coded framework of coordinating plans (e.g, [13]), or runtime protocols for coordinating negotiations between agents (e.g., [2]). The approach adopted in this paper is to accumulate the belief of an agent about next possible position of other agents in its scope. This approach has the advantage that no communication between agents is required, belief information of an agent can serve the purpose of detecting potential conflicts. In a large system where many agents exist and interact with each other, computing the communication between agents might become very resource intensive. In general, agents using this conflict model can detect potential resource conflicts with other objects that might be not able to communicate with agents. Therefore conflict detection without deploying communication between agents is advantageous. Exploiting the advantage of this resource conflict model, Le and Pinkwart [6] developed an algorithm that allows autonomous agents to learn problem solving strategies in resource conflict situations through communication with humans.

VIII. Conclusion and Future Work

In this paper, we have presented a model for detecting resource conflicts in a multi-agent system, representing a cyber-physical system. Our approach has the advantage that it makes use of agents' belief, and no communication between agents is required. We have evaluated this conflict model using a scenario of transportation service in an fictitious airport. The evaluation study showed that the conflict model is effective in improving the rate of successful executed agents' tasks.

In future, a comparison between the communication-free conflict detection approach and a communication-based one can be conducted to find out the difference in effectiveness (i.e., successful tasks). The conflict model proposed in this paper will serve as the pre-step for conflict analysis and resolution. In addition, it is a goal of our research to make use of conflicts to build a cyber-physical system that can evolve over time.

Acknowledgment

This work was funded by the NTH School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

References

[1] K. S. Barber, T. H. Liu, and S. Ramaswamy. Conflict Detection during Plan-Integration for Multi-Agent Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 31:616–628, 2000.
[2] J. Chu-Carroll and S. Carberry. Communication for Conflict Resolution in Multi-Agent Collaborative Planning. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 49–56, 1995.
[3] J. Görmer, G. Homoceanu, C. Mumme, M. Huhn, and J.-P. Müller. JRep: Extending Repast Simphony for Jade Agent Behavior Components. In *Proceedings of the 10th IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 149–154, 2011.
[4] M. Huhn, J. Müller, J. Görmer, G. Homoceanu, N. T. Le, L. Märtin, C. Mumme, C. Schulz, N. Pinkwart, and C. Müller-Schloer. Autonomous Agents in Organized Localities Regulated by Institutions. In *Proceedings of the 5th IEEE Digital Ecosystems and Technologies Conference*, pages 51–61, Los Alamitos, CA, 2011. IEEE Computer Society Press.
[5] S. Konieczny and R. P. Pérez. Merging Information Under Constraints: A Logical Framework. *Journal of Logic and Computation*, 12:773–808, 2002.
[6] N.-T. Le and N. Pinkwart. Strategy-based Learning Through Communication With Humans. In *Proceedings of the 6th International KES Conference on Agents and Multi-agent systems - Technologies and Applications*. Springer, 2012. (to appear).
[7] T. H. Liu, A. Goel, C. E. Martin, and K. S. Barber. Classification and Representation of Conflict in Multi-Agent Systems. Technical report, The Laboratory for Intelligent Processes and Systems Electrical and Computer Engineering, University of Texas at Austin, USA, 1998.
[8] J. Müller. *The Design of Intelligent Agents: A Layered Approach*. Springer, 1996.
[9] A. S. Rao and M. P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 312–319, 1995.
[10] A. Sathi and M. Fox. Constraint-directed negotiation of resource reallocations. In *Distributed Artificial Intelligence*, pages 163–193. Morgan Kaufmann, 1989.
[11] C. Tessier, L. Chaudron, and H. Müller, editors. *Conflicting agents: Conflict management in multi-agent systems*. Kluwer Academic Publishers, 2001.
[12] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer. Avoiding Resource Conflicts in Intelligent Agents. In *Proceedings of the 15th European Conference on Artifical Intelligence*, pages 18–22. IOS Press, 2002.
[13] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2nd edition, 2009.