Artificial Intelligence based Solutions for Cooperative Mobile Robots

D. Panescu, M. Kloetzer, A. Burlacu, C. Pascal

Technical University of Iasi, Dept. of Automatic Control and Applied Informatics
(e-mails: {dorup, kmarius, aburlacu, cpascal}@ac.tuiasi.ro)

Abstract: This paper presents the way some Artificial Intelligence techniques can contribute to obtaining an efficient solution for a cooperative robotic problem. Based on certain abstraction and sensing procedures the problem specification, the robots' environment and their motion capabilities can be transposed to a finite state representation. The distributed nature of the considered application involving two robots and the reduced computation resources of individual robots conducted us to attaching robot deliberative components in an agent based implementation using a specialized software platform, namely JACK. The obtained multiagent system creates a framework for an approach that allows the interleaving of planning and execution. The agents apply an A* type bidirectional search to find the movement plan. The developed coordination protocol permits correct path generation even when the environment is changing while the robots are moving. The computational complexity of the proposed approach is low, and the system operation is supported by simulation experiments.

Keywords: mobile robots, multiagent system, path planning, A* algorithm, JACK agents.

1. INTRODUCTION; TARGETED PROBLEM

Collaborative Robotics has become a growing interdisciplinary research area addressing problems like: task allocation, cooperative planning and execution, cooperative perception, multi-robot mapping and localization, formal models of multi-robot plans, multi-robot learning, self-configuration, networked robotics (Singh and Thayer, 2001, EURON SIG), some of them being tackled by the approach proposed in this paper. The above issues can be linked with the methods of Artificial Intelligence (AI), the most frequently with the multiagent systems (MASs). The agents' features – reactivity, autonomous and proactive operation – are important for Robotics (Murphy, 2000). In the case of a multi-robot system the need of coordination becomes obvious and the connection with an MAS turns out to be the practical solution (Liu and Wu, 2001, Hsu and Liu, 2005).

This paper takes into account the case of a system with two mobile robots. The task to be solved regards the robots' movement in a partially known environment so that they should meet as soon as possible. Such a scenario may appear both in an industrial environment (two mobile robots that must transfer a part or a tool) and in other types of situations (for example, in exploring or rescue robots' activities) (Dias *et al.*, 2004, Fiero *et al.*, 2002, Uny Cao *et al.*, 1997). The paper presents an approach that obtains an optimal solution to the robots' meeting problem even in dynamic environments. Although the scalability of our solution to more than two robots is a challenging matter, the proposed method has the main scientific contribution of constructing a computationally feasible solution that links a rendezvous robotic problem with a bidirectional search algorithm. This is a significant approach as the subject of more robots' meeting is still raising open issues on the time optimality and dependency on communication protocols (Meghjani and Dudek, 2011). Our solution considers the robots as components of an MAS and provides specific coordination and planning algorithms that should be run by each individual robot.

The problem specifications are as follows. The map of the environment is known and the initial positions of the two robots, too. The plane area where the robots can move is abstracted to a set of locations interconnected by several ways. The robots have to find the optimal route to traverse so that they meet in the shortest time, which conducts to determining the shortest path. The difficult aspect is that though the map is known in advance, some changes can appear, namely it may happen that some new obstacles are added and block certain ways. Thus the robots start with a planned path and then they have to adapt it when one or more ways are obstructed by objects that were not considered in the initial map. Our hypothesis is that such obstacles will be detected by the robots' sensorial systems (e.g., by using ultrasonic sensors in an approach as the one explained in (Nagy, 2009)) from the beginning of a way, namely from the moment when a robot is in an intersection, so that it will be able to find an alternative path.

As further presented in this paper, the above specified robotic problem can benefit from a set of AI methods and techniques. Besides the already mentioned agent based solution for the needed deliberative components, methods and tools from AI for environment detection and abstraction, robots' control and communication, as well as for an efficient implementation of the robotic system architecture can be involved, too. Thus, the paper is organized as follows. First, we present abstraction techniques that can be used for obtaining a finite-state representation of the problem; these

can be coupled with artificial vision instruments for the robotic environment detection. Then, the algorithmic steps of the devised robots' coordination protocol are explained. Some details regarding the searching mechanism that we use are provided, followed by a description on the agents' design and implementation in JACK programming environment. Some comments on the performed simulation experiments and few conclusions end this paper.

2. ROBOT ABSTRACTION

This section illustrates some approaches that can be employed for abstracting the motion capabilities of a mobile robot evolving in a continuous environment to a finite state representation. We consider an environment where a set of regions is predefined. These regions can be obstacles, or they can be areas of interest, such as places where the robots must meet, or locations of different parts that have to be picked by the robots.

As a primary stage, an abstraction procedure is needed that should allow both the robots' environment modelling and the obtaining of the necessary data for the planning and control mechanisms. It will involve the partitioning of the environment into a set of adjacent cells having the same shape. Such partitions (also called cell decompositions) can be created by using tools from computational geometry (Berg *et al.*, 2008), the name of the partition being given by the shape of its cells (Choset *et al.*, 2005, La Valle, 2006). Once a partition is created, each cell corresponds to a node of a graph in the abstract representation. The edges between nodes correspond to adjacency relations between cells and to control capabilities of robot for moving from one cell to an adjacent one. Then, for designing these control laws, one can use results from (Habets and van Schuppen, 2004), where feedback control laws driving all trajectories of an affine system from a polytopal or simpliceal region through a desired facet were designed. Thus, the motion and control capabilities of robots are abstracted into a finite graph, where a node (place) corresponds to a region where the robot can be located, and an edge (way) corresponds to a feedback control law driving the robot from one place to another. For a better understanding, Fig. 1 presents a triangular partition and the corresponding graph of an environment containing three obstacles.

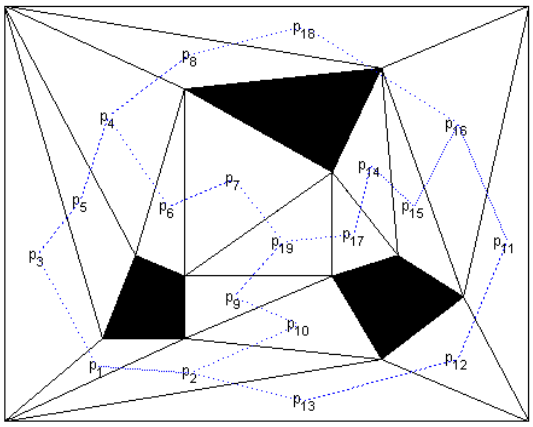

Fig. 1. A triangular partition of a given environment. The obtained abstraction has 19 nodes (labelled by p1,…,p19), and the edges are represented by dotted lines

We mention that such abstraction techniques were successfully used when motion specifications for robots are given as linear temporal and logic (LTL) statements about attainment or avoidance of some regions from the environment (Kloetzer and Belta, 2010).

The abstraction procedure can be connected with an artificial vision technique in order to obtain measurements from the workspace. Artificial vision is the most powerful sensor of robots regarding the quantity and quality of obtained information, seeking to emulate the performance of the human eye. For the proposed approach, environment analysis will be conducted using images acquired with a fixed camera (eye in the sky (Siegwart, 2008)). A visual feature detection method is employed for object description and for construction of the Voronoi diagrams (Berg *et al.*, 2008). These are adaptable geometric structures that have numerous applications in physics, astronomy, robotics and social geography (Berg *et al.*, 2008). A Voronoi diagram is a complete roadmap method that tends to maximize the distance between robot and obstacles in the map. For each point in the free space, the distance to the nearest obstacle is computed. The Voronoi diagram is defined by the edges formed by these sharp ridge points. When the configuration space obstacles are polygons, the Voronoi diagram consists of straight segments that define the Voronoi cells.

Image processing algorithms can generate different types of visual features. For objects' description, image moments, contour or point features may be used. In visual analysis applications point features are low level descriptors precisely locatable and persistent. These basic properties define the usefulness of point features for object description. In the considered solution, the Harris corner detector is used, as an algorithm based on an underlying assumption that the point features are associated with the maximum of the local autocorrelation function. The algorithm has proved popular due to its high reliability in finding L junctions, and the good temporal stability makes it an attractive corner detector for tracking (Gonzales and Woods, 2008). Initially, for a more effective detection of visual features it is necessary to improve the image through a variety of pre-processing mechanisms: contrast adjustment, conversion to grayscale, filtering, conversion to binary images. The Harris operator is used to detect the point features that can characterize the objects within the working scene. Using point features extracted via the Harris algorithm (red representation in Fig. 2), the Voronoi diagrams are constructed and overlaid onto the image, as shown in Fig. 2.

A set of equidistant points are obtained (blue representation), points that will represent the base information for the map construction. The remaining vertices of the Voronoi cells represent the nodes of a graph. Edges between two nodes in the graph have the cost distance equal to the distance (expressed in pixels) between the two considered vertices in the image plane. A fundamental constraint is that an edge can be added to the graph if it can be completed by the robot (the

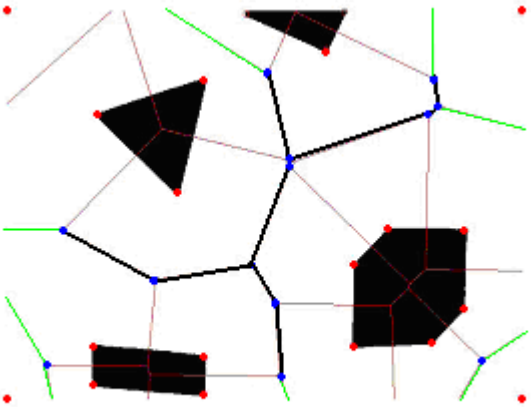

Fig. 2. Voronoi diagram resulted from visual point features

robot's diameter transformed into pixels doesn't intersect any object).

In Fig. 2 the edges are divided in categories: those that are defined by two nodes (black representation) and those corresponding to one node (green representation). More details on artificial vision utilization for robots' environment detection and abstraction can be found in (Burlacu *et al.*, 2010).

The techniques presented in this section enable us to solve a robotic problem on the finite representation where places are connected by ways that can be followed by robots, similar to the approach considered in (Panescu *et al.*, 2010). The plans obtained on such a representation correspond to a sequence of feedback control strategies in the initial environment, and thus the solution provided in the remainder of the paper can be adapted to real robotic scenarios.

3. THE COORDINATION PROTOCOL

Because the on-board computing resources of a mobile robot are often limited, the proposed solution is to have an external computer to run the agents dedicated to the two robots. This means each robot has as its high level decision system an agent. The two agents can communicate one with the other and with the robot under control. If the task to be solved becomes more complicated and further reasoning abilities are needed, then a scheme with the two agents running on two interconnected computers may be also taken into account, as shown in Fig. 3.

The robots' operation comprises two main parts: planning and execution. These must be interleaved in a specific manner, so that the imposed performance is obtained. The planning part is solved by the MAS, applying a distributed search; then, each agent knows the solution and can send to the corresponding mobile robot commands regarding the execution of necessary movements; they come back to planning when a change in their environment is detected. At a first decomposition level the agents' activity is conducted according to the following scheme:

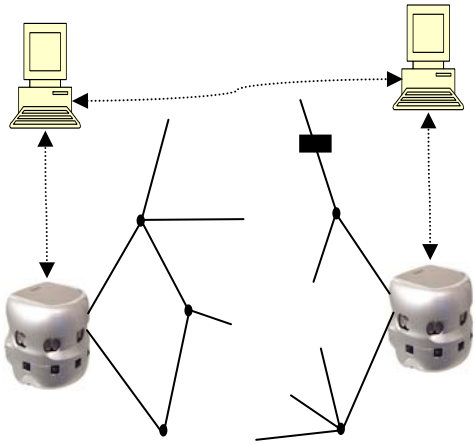

Fig. 3. A multi-agent architecture to solve the navigation for two mobile robots

Phase 1. Receive the initial robots position.

Phase 2. Plan a whole path to obtain the robots' meeting.

Phase 3. Launch the execution of the planned path and then monitor the conducted robot movement.

Phase 4. Take a decision in accordance with the information acquired from robots: if the goal position is to be reached in the next step, an approaching command is sent to each robot, then the mission is ended; if the sensorial information regards a blocked way, then go to Phase 1.

The above cycle is explained as follows. Based on the initial information on the robots' positions and the environment map an appropriate searching algorithm is carried out to find the solution for the robots' movement. The proposed approach is for using the A* heuristic search in a distributed manner, because it offers certain advantages. The method provides both completeness and optimality (Robotin *et al.*, 2010, Russell and Norvig, 2003, Yokoo and Ishida, 2001).

After the Phase 2 the two agents will know the entire optimal succession of ways connecting their initial positions (of course, one may exclude the case when no solution is possible). In the above description Phases 3 and 4 appear in sequence, but in fact they will be interleaved. This means an agent sends towards its robot a specific command depending on the sensorial information acquired from robot. Namely, when the planned way is obstructed the mobile robot sensorial system sends this information to its agent and the entire cycle is re-started through the agents' coordination mechanism. Meanwhile, the two agents can detect the moment when the two robots are supposed to meet and thus they will inform the robots to apply a specific approaching procedure, so that they should come nearby without collision. To better clarify how the Phases 3 and 4 operate, further details are provided, explaining the agent coordination protocol and the way the decision about the next action is taken. Let us note P_i the current position of one robot and respectively P_j the present position for the other robot, while P_{i+1} and P_{j+1} are the next positions of the two robots. The developed protocol contains the following steps:

Step 1. If the agent received a message regarding a blocked way then it updates the map of the environment.

Step 2. If the agent received a message regarding the next position of the partner robot, then it updates P_j and P_{j+1}.

Step 3. If the agent received a message for re-planning then it sends the command to stop the controlled robot at the end of the current way; when this happens, P_i becomes P_{i+1} and the agent continues with the **Step 5**.

Step 4. If $P_i = P_j$ then the agent sends to the robot the command to apply the approaching procedure and the cycle is ended.

Step 5. The agent sends a message to the robot to check the way from P_i to P_{i+1} together with the other ways connected to P_i and waits for the robot's answer. For every way connected to P_i that was detected as blocked, the agent updates the environment map and sends the respective information to the partner agent.

Step 6. If the way P_i to P_{i+1} is blocked, the agent sends to the partner agent the message asking a re-planning phase and then it goes to the Phase 1.

Step 7. If the agent received a message from the other agent for re-planning then it goes to Phase 1.

Step 8. If the way from P_i to P_{i+1} is not blocked then the agent tests for $P_{i+1} = P_{j+1}$. If this condition holds

then:

- it sends to the robot the command to go to P_{i+1}, to apply the approaching procedure and the cycle is ended;

- it sends P_{i+1} to the partner agent, as the next position.

else:

- it sends to the robot the command to go to P_{i+1};

- it sends P_{i+1} to the partner agent, as the next position.

Step 9. The agent waits for the information from its robot regarding the reach of the commanded position. When this happens, P_i becomes P_{i+1} and it goes to the **Step 1**.

These steps regard the phases 3 and 4 in the above presented scheme and the connection with the phases 1 and 2, when needed. It is to remark that an agent based implementation overcomes a purely sequential operation. Namely, it is considered a BDI (Belief-Desire-Intention) agent architecture, as it can be obtained by making use of the Jack agent development environment (Wooldridge, 2001, Evertsz *et. al.*, 2004). In an agent based implementation the execution is an event driven one (Padgham and Winikoff, 2004). This means an agent is waiting for events, mainly these meaning messages received from its environment. As already told, in our case an agent can receive messages from the other agent and from the controlled robot. These messages can asynchronously arrive and they will be kept in a queue. So, even if a message is not received during the waiting state within the Step 9, it will be kept and accordingly used. In fact, the Steps 1, 2 and 3 are carried out as soon as the corresponding messages are received and the agent ends the current operation. Thus, one can say that the agents can decide in almost real-time on the necessity of a re-planning stage, a present or next approaching procedure (the only possible delay regards the time spent by the agent to finalize the ongoing activity). Regarding the Steps 4 and 8, they are devoted to allow a smooth robots' meeting without the need to a priori establish this moment. Thus, the robots can have different speeds and the coordination procedure allows a correct robots' approaching.

It is important to understand how the re-planning phase is launched and operates. It is started when a robot detects the next way in its path as being obstructed. Until that moment the corresponding agent had already sent to its partner both the present robot location (P_i) and the way that is blocked ($P_i - P_{i+1}$). According to the Step 2, the agent also received the present position of the other robot (P_j). Thus all the information for applying a new A* based search is available: the initial positions of the two robots and the updated map of the environment. As soon as a new entire path is found, this is sent for execution to the two agents, and they come back to the Phase 3. It is to remark that when the re-planning phase is asked, the agents have already exchanged information on all the environment changes produced since the last updating.

Even during an execution without any obstacle appearance the agents send messages to each other, in order to be informed about the next positions to be reached, according to the Step 8. Thus, there is no need for other feedback so that the robot should have a meeting without collision. One can easily show that this holds in the proposed procedure for all the paths containing at least three segments. The only restricting supposition is that once a robot starts moving on a path it arrives at its end, because the way was checked to be free. The case of an obstacle being introduced on a way when a robot is already moving on it is not treated, but a corresponding feedback can be added to the developed mechanism.

4. THE ALGORITHM TO FIND THE OPTIMAL PATH

As already mentioned, a heuristic search was used to get the plan of the robots' movement, namely the A* algorithm. As the considered application contains two agents an obvious construction would be to apply a distributed approach, like the one offered by the bidirectional search (Russell and Norvig, 2003, Yokoo and Ishida, 2001). The problem is that when applying A* in a bidirectional search, the performance of the method is highly dependent on the heuristic function depression. Namely, for a path planning problem (a case for which the difference between the values of the heuristic function for the successors of a node can be high) when trying to put into practice an efficient bidirectional search, the performance in the combined problem space is worse than when using the initial problem space (Yokoo and Ishida, 2001). That is why the proposed approach is to avoid the drawbacks of making the search in the combined problem

space (this has n^2 states when the initial space has only n states), but to further benefit by the use of a distributed bidirectional algorithm. Thus, the two agents know the initial and the goal states (according to the initial positions of the two robots) and apply A* with the corresponding initial data: the initial state for an agent is the goal state for the other one. The goal in each of the two searches is fixed and thus the initial problem space is kept, while the ending condition regards the moment when one agent finds a position that is already within the solution of the other agent. Some details of the constructed search procedure are further presented.

The devised bidirectional A* search relies on the typical A* routine for finding successor nodes and it is implemented by each of the two agents according to the following protocol:

Agent 1:

(start node = initial position of Agent 1;

goal node = initial position of Agent 2)

1. Use the typical A* strategy to choose a successor node (denoted by P_1)
2. Update the current path (best path to P_1)
3. Receive from Agent 2 its current path (denoted by $Path_2$)
4. IF P_1 belongs to $Path_2$

- construct the whole path by combining the current path to P_1 with $Path_2$

- send the whole path to Agent 2

- exit the search algorithm and begin movement

ELSE

- go to 1

END IF

Agent 2:

(start node = initial position of Agent 2;

goal node = initial position of Agent 1)

1. Use the typical A* strategy to choose a successor node (denoted by P_2)
2. Update the current path (best path to P_2) and send it to Agent 1
3. IF Agent 1 sends the whole path

- exit the search algorithm and begin movement

ELSE

- go to 1

END IF

Such a searching approach benefits from the complexity reduction determined by the bidirectional search (Russell and Norvig, 2003). In our case this is coupled with the fact that the two agents can work simultaneously and exchange messages to detect the moment when the solution was reached, which additional reduces the period for the solution reaching; of course in this case there is a time spent with the communication phase.

5. ON THE AGENTS' DESIGN AND IMPLEMENTATION

The two agents are carried out in the programming environment named JACK and Fig. 4 represents their planning and execution design diagram (Padgham and M. Winikoff, 2004, JACK intelligent agents, 2005). The common agent structure is represented with continuous lines, while the entities specific for the Agent 1 are marked with discontinuous line, and the elements that appear only in the case of Agent 2 are represented with dotted line. As already mentioned, an agent has an event driven operation. The reception of a message represents an external event (*ExternalMsg*) that is treated by a corresponding plan, found in accordance with the message contents and the BDI mechanism. There will be internal events too, the ones that an agent is using to drive its activities, like the *Planning* and *Execution BDI Events* in Fig. 4.

Three types of messages are used by agents and these determine the launching of corresponding plans:

- messages regarding the need of a re-planning phase, containing as parameters the present robot position and the blocked way (the corresponding plan is labelled *Re_PlanningMsg* in Fig. 4).

- messages concerning the present robot position, which are used when the partner agent has asked re-planning (these activate the plan called *PresentPositionMsg* in Fig. 4).

- the third type of messages regards the path planning phase and this is different for the agent that is supposed to assemble the plan of the path – Agent 1, and respectively the other agent, called Agent 2. The Agent 1 sends a message after a plan was found for the entire path and the message contains this path (the plan *PathMsg* handles the message); Agent 2 sends messages containing the positions that it has already planned; such messages are handled by the plan *NextPositionMsg*.

At the initial time, the planning process is started after each agent has received the other agent's robot position. Through the *PresentPositionMsg* plan the *Planning BDI Event* is posted as an internal message. This activates the *FindNextPosition* Plan that materializes the corresponding searching algorithm, as described in the previous section.

The agent's knowledge is kept in three entities, named *BeliefSets* for JACK agents. Thus, the *AgentPath* beliefset stores the information on the path for the commanded robot. In the planning phase this knowledge base is filled in with the positions found through the A* algorithm, while during the execution it keeps the track of the robot's movements. The *OtherAgentPath* beliefset contains the current position and the path planned by the other agent for its robot. The third

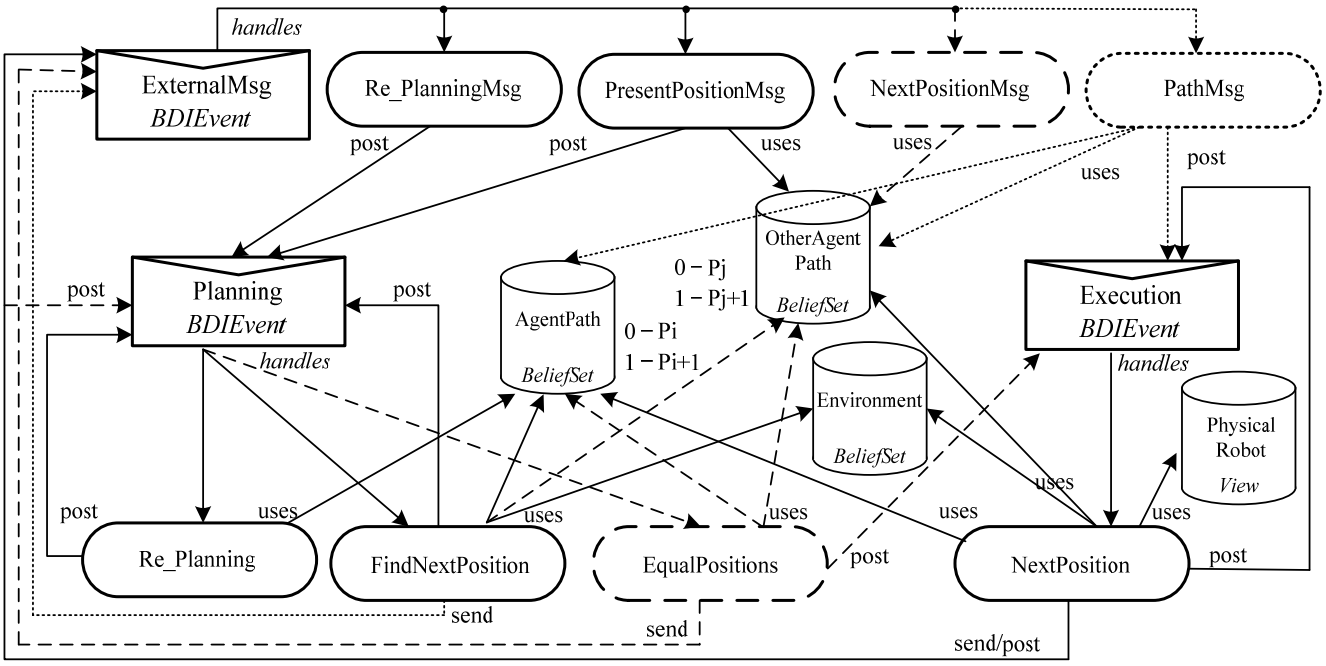

Fig. 4. The agents' planning and execution diagram

knowledge base, the *Environment* belief set holds the information about the map of the robots' environment. All these belief sets are used by the *FindNextPosition* plan to determine the next robot planned position. This plan contains an additional action for the Agent 2, namely the one for sending a message regarding the next planned position towards the Agent 1 (this appears with dotted line in Fig. 4). The last action of the *FindNextPosition* plan posts a new *PlanningBDIEvent* message, thus resulting a loop (each iteration corresponds with an application of the algorithm presented in the previous section). This loop is ended when Agent 1 determines a coincidence between two positions within the belief sets *AgentPath* and *OtherAgentPath*. At that moment the *EqualPositions* plan is activated; this can compose the whole path to be traversed by the two robots. After that, the Agent 1 up-dates its belief sets and sends the entire path to the Agent 2 (as an external message). The *Re_Planning* plan is used after an external message regarding the need of a re-planning phase is received and it has as consequence the ceasing of the execution phase when the corresponding robot reaches the next position. After that both agents enter the planning cycle.

The execution phase is started by different events for the two agents. Agent 1 starts its execution phase by launching the *ExecutionBDIEvent* from the *EqualPositions* plan. This event is handled by the *NextPosition* plan, which uses the agent's belief sets to determine each movement This plan includes the tests within the Steps 2 and 5 of the algorithm presented in Section 3. The agent's plan supposes a link with the physical robot by the means of a JACK view entity, which materializes the communication with the robot controller. As with the *FindNextPosition* plan, the *NextPosition* plan, which controls the execution phase, establishes a loop (see the algorithm in Section 3). This is ended by one of the following three situations. One case regards the moment when the test for the two robots' positions shows they are supposed to meet. The second ending condition appears when the test regarding an obstructed way has a positive result; in this case the agent sends an external message to inform the partner agent and to restart its planning phase. The third situation is when according to the agent's *AgentPath* beliefset its next robot position is not available, as a consequence of receiving an external re-planning message from the other agent. The execution phase is similar for the Agent 2, except for its starting condition that is determined by the *PathMsg* plan.

The structure of Fig. 4 conducted the development of JACK agents and these can run on the same computer, or they can be deployed on distinct networked computers.

6. EXPERIMENTAL RESULTS

This paper regards an on-going research. The plan is to couple the two agents with two Khepera type mobile robots. Until now only some simulation experiments have been conducted to prove the efficiency and adequacy of the proposed scheme, the performance of the agent based path searching mechanism and of the agents' coordination protocol.

A sequence of situations obtained within an illustrative scenario is presented in Figs. 5, 6 and 7.

One can see in Fig. 5 the initial robots' positions (marked with red and blue colours) and the map of their environment. The experiment considers the case when the nodes in the searching graph will have one or two successors. The weight of edges is proportional with the length of ways. Thus, in our case the weight of horizontal edges is 1, while the oblique edges have the cost of $\sqrt{2}$, except for the edges that

correspond to the ways between positions 5 and 8, 24 and 32 which have the weight of $\sqrt{5}$. The optimal path for this map consists in the following succession of positions: 1 – 2 – 4 – 11 – 17 – 21 – 27 – 33 – 36 – 38. In fact, the Agent 1 (the one for the red robot) detected the path from position 1 to 21, and the Agent 2 the path from 38 to 11, and then the node corresponding to the position labelled 21 is detected by the Agent 1 in both agents' paths and the searching processes were ended, the result being the optimal solution. The image in Fig. 6 shows that in our experiment five ways were successively detected as being obstructed.

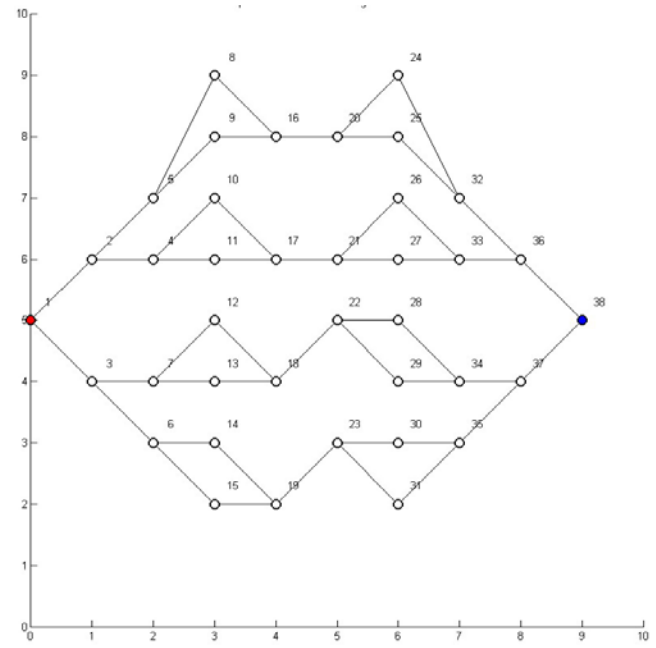

Fig. 5. The environment map and initial positions of robots

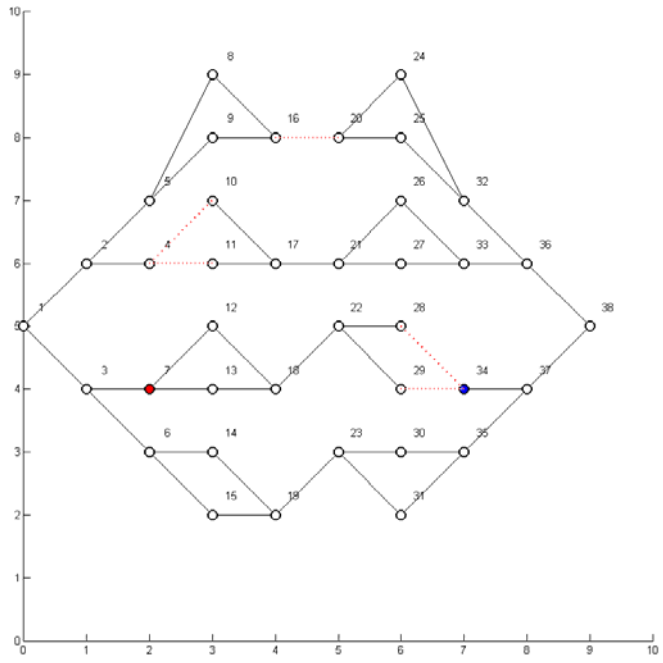

Fig. 6. The environment with several obstructed ways

This is a difficult case, as the two agents have had to re-plan their path three times. Thus, first the robots begin with the path having the minimal cost and when the red robot is in the position 4, both possibilities to continue are being obstructed (see Fig. 6). The optimal path in these new conditions is the one through the positions: 2 – 5 – 9 – 16 – 20 for the first robot (as discovered by the Agent 1), and 36 – 32 – 25 – 20 for the second one. A further blocked way appears (16 – 20) and the agents carry out another planning phase. The reconfigured path is: 9 – 5 – 2 – 1 – 3 – 7 – 13 – 18 – 22 for the red robot, and 25 – 32 – 36 – 38 – 37 – 34 – 28 – 22 for

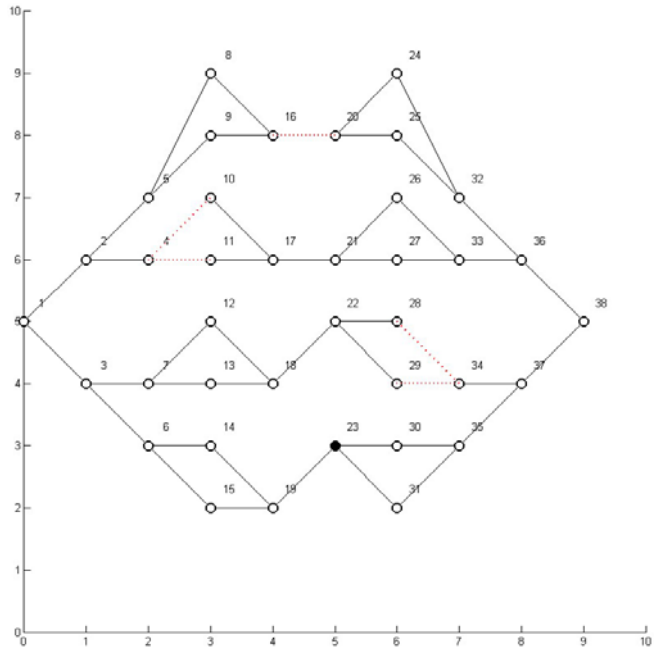

Fig. 7. The robots in the meeting position

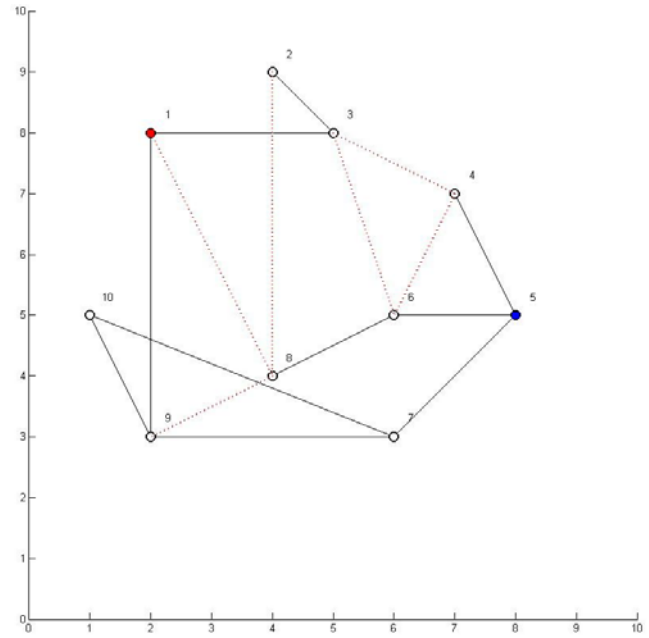

Fig. 8. A case with a large number of blocked ways

the blue one. The two paths are traversed until the first robot is in the position 7 and the second robot is in the position 34. By that time, the blue robot detects the two blocked ways and asks for the re-planning phase. The agents are able to find a new solution, this being: 7 – 3 – 6 – 14 – 19 – 23 for the first robot, and 34 – 37 – 35 – 30 – 23 for the second one, which is again the optimal one for the current environment. By making use of this plan the robots successfully meet in position 23, as shown in Fig. 7 obtained in our simulation.

Even the distributed solution has visible advantages in the common cases, it may appear that when a large number of blocked ways occurs the multi-robot solution is not better than a single robot one. Such a case is presented in Fig. 8. The positions 1 and 5 are the robots' initial places. The scenario involves several movement attempts. First, the optimal routes are 1 – 3 – 4 for the red robot and 5 – 4 for the blue one. The robots start the movement on these paths and when reaching the positions 3 and respectively 4, the ways 3 – 4, 3 – 6 and 4 – 6 become obstructed. The new re-planned path is 3 – 2 – 8 – 6 for the first robot, and 4 – 5 – 6 for the second robot. After carrying out a single movement, when the robots are in the positions 2 and 5, the way between 2 and 8 becomes obstructed. A new reconfigured path is 2 – 3 – 1 – 8, coupled with 5 – 6 – 8.

After two movement steps of the two robots, the first of them detects the blocked way between the positions 1 and 8, while the second one observes the obstructed way between the positions 8 and 9. After a further re-planning, the path 1 – 9 – 7 – 5 is used by the first robot, while the second robot traverses the route 8 – 6 – 5. Thus, the meeting position is the initial place of the blue robot. Though in this scenario the distributed solution seems to provide no advantage (it appears that the first robot had to traverse the whole path), it is to remark that the second robot influences a faster reaching of the solution through its sensorial acquisition. So, this robot is the one that detects the obstructed way between the positions 8 and 9, and this information being provided to the agent of the red robot allows it to reduce the number of re-planning phases.

7. CONCLUSIONS

The proposed architecture is an additional example on how the AI techniques can determine an improved solution for Robotics. The way planning and execution are interleaved and the robots are coordinated by the means of an MAS allow the system to face a dynamic environment. Even when several changes appear during the robots' movement, their decisional system is able to reconfigure the path in an optimal way. It is to mention that the sensorial system considered for the mobile robots can be a simple one, as only the presence of the obstacles has to be detected.

The agent based solution has distinct advantages regarding the way a dynamic environment can be handled. An environment modification is producing an event that can be appropriately treated by the provided plans. The BDI agents' reasoning mechanism determines a robust planning scheme because more plans can be devised for the same event; these will be chosen by a suitable filtering scheme, and used one after the other when failures appear. Thus the proposed application can be enhanced, so that the planning phase should be conducted by more decision criteria: if there is no dangerous area find the shortest path, when such an area exists find a path that avoids the robots' approaching the prohibited places.

This paper focused on developing an efficient planning and coordination protocol on a finite-state representation of the problem. The connection between this solution and a real robotic scenario was maintained by various AI tools for environment detection and abstraction and for robot control. Although the involved bidirectional search algorithm implies a computationally feasible method, it prevents the direct scaling of the current scenario to more than two robots.

As future work, we plan to extend the approach to more robots and to use Khepera type robots for carrying out real experiments of the proposed techniques. It is to further study if the bidirectional search can be extended for the case of three robots that are supposed to meet by repeating the procedure for pairs of robots, or the whole algorithm has to be redesigned.

ACKNOWLEDGEMENTS

The second author acknowledges the support of the CNCS-UEFISCDI grant PN-II-RU PD code 333/2010. The third author acknowledges the support of the project PERFORM-ERA "Postdoctoral Performance for Integration in the European Research Area" (ID-57649), financed by the European Social Fund and the Romanian Government. The forth author acknowledges the support of BRAIN "Doctoral scholarships as an investment in intelligence" project, financed by the European Social Found and Romanian Government.

REFERENCES

de Berg, M., Cheong O., van Kreveld, M., and Overmars M. (2008). *Computational Geometry: Algorithms and Applications*, Springer-Verlag.

Burlacu A., Kloetzer M. and Panescu D. (2010). Some AI Based Approaches on Mobile Robots Motion Planning, *Solid State Phenomena*, vols. 166-167, pp. 101 - 108.

Choset H., Lynch K.M., Hutchinson S., Kantor G., Burgard W., Kavraki L.E. and Thrun S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Boston, USA.

Dias, M.B., Zinck, M., Zlot, R.M. and Stentz, A. (2004). Robust Multirobot Coordination in Dynamic Environments, IEEE International Conference on Robotics and Automation, April, pp. 3435 - 3442.

EURON SIG (Special Interest Group on Cooperative Robotics of European Robotics Research Network), http://www.aass.oru.se/Agora/EuronCoop/.

Evertsz R. *et. al*. (2004). *Implementing Industrial Multiagent Systems using JackTM*, PROMAS 2003, Australia, Springer, pp.18-48.

Fiero R. *et al.* (2002). A Framework and Architecture for Multi-Robot Coordination, *International Journal of Robotics Research*, Oct.-Nov., pp. 977 – 995.

Gonzalez R. and Woods R. (2008). *Digital Image Processing Third Edition*, Pearson Pretince Hall.

Habets L., and van Schuppen J. (2004). A Control Problem for Affine Dynamical Systems on a Full-dimensional Polytope, *Automatica*, 40: 21–35.

Hsu H.C.-H. and Liu A. (2005). Multiagent-Based Multi-team Formation Control for Mobile Robots, *Journal of Intelligent and Robotic Systems*, 42, pp. 337–360.

JACK™ Intelligent Agents, *Agent Manual*, Agent Oriented Software, Carlton South, Victoria, Australia, 2005

Kloetzer M. and Belta C. (2010). Automatic Deployment of Distributed Teams of Robots from Temporal Logic Motion Specifications, *IEEE Transactions on Robotics*, 26(1): 48–61.

La Valle S.M. (2006). *Planning algorithms*, Cambridge University Press, Cambridge, UK.

Liu J. and Wu J. (2001). *Multi-Agent Robotic Systems*, CRC Press, Boca Raton, pp. 4 – 17.

Meghjani M. and Dudek G. (2011). Combining Multi-Robot Exploration and Rendezvous, 2011 Canadian Conference on Computer and Robot Vision, IEEE Computer Society, pp. 80 – 85.

Murphy R. (2000). *Introduction to AI Robotics*, The MIT Press, pp. 257 – 305.

Nagy I. (2009). Behaviour Study of a Multi-Agent Mobile Robot System during Potential Field Building, *Acta Polytechnica Hungarica*, Vol. 6, No. 4, pp. 111 – 136.

Padgham L. and Winikoff M. (2004). *Developing Intelligent Agent Systems. A Practical Guide*, John Wiley & Sons, Chichester, pp. 8-31.

Panescu D., Kloetzer M., Burlacu A., Pascal C. (2010). A multiagent based solution for mobile robots path planning, The 14th International Conference on System Theory and Control, Sinaia, pp. 225 - 230.

Robotin R., Lazea Gh., Dobra P. (2010). Mobile Robots Path Planning With Heuristic Search, *Journal of Control Engineering and Applied Informatics (CEAI)*, Vol.12, No .4, pp. 18-23.

Russell S. and Norvig P. (2003). *Artificial Intelligence. A Modern Approach*, Prentice Hall, Upper Saddle, pp. 96–106.

Siegwart R. and Nourbakhsh I. (2008). *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge, USA.

Singh S. and Thayer S. (2001). *ARMS: Autonomous Robots for Military Systems. A Survey of Collaborative Robotics Core Technologies and Their Military Applications*, Technical Report, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Uny Cao Y., Fukunaga A., Kahng A. (1997) Cooperative Mobile Robotics: Antecedents and Directions, *Autonomous Robots*, 4, pp. 1–23.

Wooldridge M. (2001). Intelligent agents. In: *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Edit., The MIT Press, Cambridge, pp. 54-61.

Yokoo M. and Ishida T. (2001). Search Algorithms for Agents. In: *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Edit., The MIT Press, Cambridge, pp. 179-191.