International Journal of Computational Intelligence Systems, Vol. 5, No. 1 (February, 2012), 39-52

Autonomously Implemented Versatile Path Planning for Mobile Robots Based on Cellular Automata and Ant Colony

Adel Akbarimajd*
Electrical and Computer Engineering Group, Faculty of Engineering,
University of Mohaghegh Ardabili, Ardabil, Iran.
akbarimajd@uma.ac.ir

Akbar Hassanzadeh
Faculty of Electrical and Electronics Engineering,
Shiraz University of Technology, Shiraz, Iran
akbar.hassanzade@gmail.com

Received 16 October 2011
Accepted 18 January 2012

Abstract

A path planning method for mobile robots based on two dimensional cellular automata is proposed. The method can be applied for environments with both concave and convex obstacles. It is also appropriate for multi-robot problems as well as dynamic environments. In order to develop the planning method, environment of the robot is decomposed to a rectangular grid and the automata is defined with four states including Robot cell, Free cell, Goal cell and Obstacle cell. Evolution rules of automata are proposed in order to direct the robot toward its goal. CA based path planner method is afterwards modified by a colony technique to be applicable for concave obstacles. Then a layered architecture is proposed to autonomously implement the planning algorithm. The architecture employs an abstraction approach which makes the complexity manageable. An important feature of the architecture is internal artifacts that have some beliefs about the world. Most actions of the robot are planned and performed with respect to these artifacts.

Keywords: Cellular Automata, Ant Colony, Mobile Robot, Path Planning, Layered Architecture

1. Introduction

Path planning problem of a mobile robot means finding a collision-free path between two specified positions in robot's configuration space[1]. The robot should move from initial configuration to goal configuration while avoiding obstacles. Path planning of mobile robots is a well studied problem if convex obstacles are assumed and exact map of environment is given. Conventional path planning methods can be classified into three major categories including cell decomposition, road map and potential field methods[2]. In cell decomposition methods free space of the environment is divided into some *cells*. By connecting adjacent cells a *connectivity graph* is made and a path is found by searching in the graph. Roadmap methods capture the connectivity of the free space in a network of one-dimensional curves called *roadmap* and then use it as a set of standard paths wherein the path of the robot should be determined. In potential field technique the robot is modeled as a moving object inside an artificial potential field whereas *attractive* potentials are assigned to the robot's goal and *repulsive* potentials are assigned to obstacles. All of these approaches need a complete representation of

* Corresponding Author

configuration space which necessitates large computational efforts. In general, the complexity rises exponentially with the number of degrees of freedom of the robot and the dimensions of the configuration space[3]. Currently, the focus of studies is shifted to development of computationally low-cost path planning techniques.

Cellular Automata (CA) as distributed spatially extended systems were originally developed by Von Neumann[4] and then extended by Burks[5,6]. CA consists of a large number of simple components (*cells*) with local links. It has the ability of performing complex computations with a high degree of efficiency and robustness. CA can also be considered as an alternative for differential equations[7]. For these reasons, CA has been extensively used in technology, computer science, mathematics and natural science. Examples include image processing[8], reconfigurable robots[9], amino acid composition[10] and modeling of phenomena such as urban growth[11], earthquakes[12] and galaxy formation[13]. It has also been used for high speed simulation of scientific models and for computational tasks (see Ref. 14 for examples). According to the advantages of CA in fast and reliable parallel computation and their local computation properties, these architectures can be a good candidate to be employed as a path planning tool in robotics.

Some researchers have used cellular automata as a path planning tool for mobile robots. Shu and Buxton presented a simple path planning algorithm for mobile robots in the presence of convex obstacles[15]. Tzionas et al introduced a collision-free path planning algorithm for a diamond-shaped robot based on retraction of free space onto a Voronoi diagram which is constructed through the time evolution of cellular automata[16]. Behring et al showed that a CA allows efficient computation of an optimal collision-free path from an initial to a goal configuration on a physical space cluttered with obstacles[17]. Marchese presented a reactive path-planning algorithm for a non-holonomic mobile robot on multilayered cellular automata[18]. He afterwards introduced a fast path planner for a multi-robot environment composed of robots with generic shapes and sizes (user defined) and different kinematics[19]. In his work the robot should have primary information about shapes and sizes of the obstacles. As preliminary works of this paper we introduced path planner cellular automata for mobile robots in Ref. 20. In Ref. 21 the method is modified for concave obstacles and in Ref. 22 an autonomous architecture was employed to implement the planning method. In Ref. 23 a path planning method for convex obstacles was developed based on a newly introduced concept "best goal directing cell".

None of above works represents a blind algorithm to deal with both concave and convex obstacles as well as being applicable for single-robot and multi-robot environments. In this paper we will unify our above-mentioned preliminary works to introduce a planning method that does not need primary information of the environment and it is suitable for both convex and concave obstacles. It is also appropriate to be used in the presence of dynamic obstacles and meanwhile it is applicable for multi-robot environments. Moreover this paper exploits an autonomous architecture to implement the proposed algorithm.

The aims of this paper are manifold. The first aim is to develop a CA based path planning algorithm. One major challenge in path planning of mobile robots is dealing with concave obstacles. The robot may be entrapped in the concave region due to local minima. According to the advantages of the *ant colony* algorithms in solving complex problems with large search spaces[24, 25], we can nominate ant colonies to be employed in path planning algorithms of mobile robots more specifically in the environments with concave obstacles. As the second aim, an algorithm inspired by *ant colonies* will be incorporated with CA based path planning algorithm to extend it for concave obstacles. In order to implement the path planning algorithm the mobile robot must be capable of getting sensory data, analyzing information, making decisions and doing proper control commands. As ultimate expectation, all of these tasks should be put into operation in a unified autonomous architecture. Normally the first concern about biologically inspired algorithms is architecture of their implementation. In this paper, as the third aim, we will introduce a layered autonomous architecture to perform path planning algorithm. The architecture employs an abstraction approach which makes the complexity manageable.

2. CAs Based Path Planning Method

2.1. *Cellular Automata*

A cellular automaton is a dynamical system which is discrete in state, space and time. It consists of a regular

lattice of cells in some dimension d[4-7]; each cell can be in one of a finite number of states and may identically interact with adjacent cells to update its state. Updating the states – or evolution of CA – is executed based on a local interaction rule which is the same for all cells[4-7]. Some basic concepts of cellular automata are as the sequel:

- *Cells:* finite-state machines with identical pattern of local connections to other cells.
- *State:* a value which is assigned to a cell in a particular time step. The number of possible states is finite and states may be represented by some integers or by some symbols. The state of cell i at time t is symbolized as S_i^t .
- *CA's rule:* a local rule which takes current states of a cell and the states of its neighboring cells as inputs and returns the next state of the cell. CA's rules can be expressed as a lookup table.
- *Neighborhood:* a cell itself and its adjacent cells in radius r. For one-dimensional arrays definition of neighborhood is simple, however, for two-dimensional arrays there are different definitions for neighborhoods. *Neumann neighborhood* and *Moore neighborhood* are two well-known ones. The Neumann neighborhoods consider all adjacent cells of a middle cell as neighbors whereas the Moore neighborhoods only consider top, bottom, left and right cells as neighbors.

Fig. 1 illustrates a one-dimensional, binary-state, nearest-neighbor (r=1) cellular automaton with array length I=8. Both the lattice and the rule table for updating the lattice are illustrated. The lattice configuration is shown at two time steps.

2.2. *Path Planning Algorithm*

2.2.1. Modeling of configuration space as CA

The aim of this section is to model the robot and its configuration space as CA. The rational behind this modeling is that in CA the state of a cell is updated according its neighborhood; hence the resulting planning method would not need complete representation of workspace.

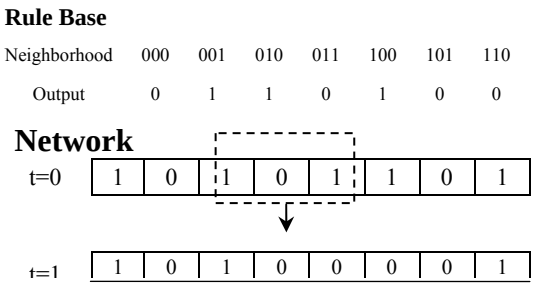


Fig 1. A binary one-dimensional CA

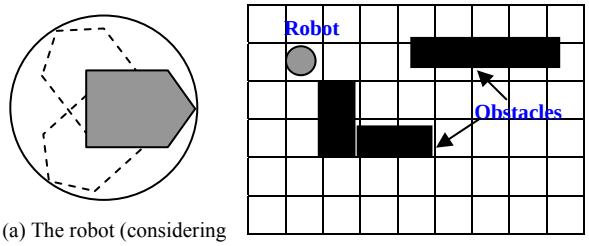
The robot's environment is characterized by the following assumptions:

- The robot is encircled and changed into a circular robot (See Fig. 2.a) and it is assumed to be a free flying object.
- Goal coordinates (x_g ,y_g ,θ_g) are given and the robot can obtain its own coordinates (x_t ,y_t ,θ_t) at each time step (for example by GPS or dead reckoning).
- The robot has only a short sensing depth in order to recognize if the adjacent cells are free not.

In order to model the problem as cellular automata, we decompose the work-space into a $I \times J$ rectangular grid where the robot can be totally located in a cell. Fig. 2.b shows the discrete work-space of a robot. In this discrete work-space three types of cells can be recognized: *Robot cell*, *Obstacle cell* and *Free cell*. Robot cell is a cell wherein the robot is located, *obstacle cells* are cells that fully or totally occupied by obstacles and the other cells are *free cells*. These types of cells are denoted by *R, O* and *F* respectively. One of the free cells that correspond to goal position is called *Goal cell* and is denoted by *G*. Now, we have a lattice of cells with a finite number of possible *states: {R, O, F, G}* that can be interpreted as two dimensional $I \times J$ cellular automata with four states.

2.2.2. Evolution rules for single robot problem

The CA defined in the previous subsection must be evolved in such a way that the robot cell approaches goal cell. Thus, appropriate construction of CA's evolutionary rules may result in path-planner cellular automata. If the obstacles assumed to be convex, we propose rule base described in Fig. 3 to build desired CA. In this rule base there are four rules **R1, R2, R3**



(a) The robot (considering different orientations) can be encircled and converted to a circular robot

(b) Robot cell, obstacle cells and free cells in the work-space

Fig 2. Decomposition of the workspace and constructing cellular autamata

and **R4.** In these rules, at each time step the Robot cell is exchanged with a free cell that is the nearest cell to the goal cell. Thus the robot cell is evolved to the nearest cell to the goal cell. It is noteworthy to mention that only one of these rules is selected at each time step according to relative position of the robot cell and goal

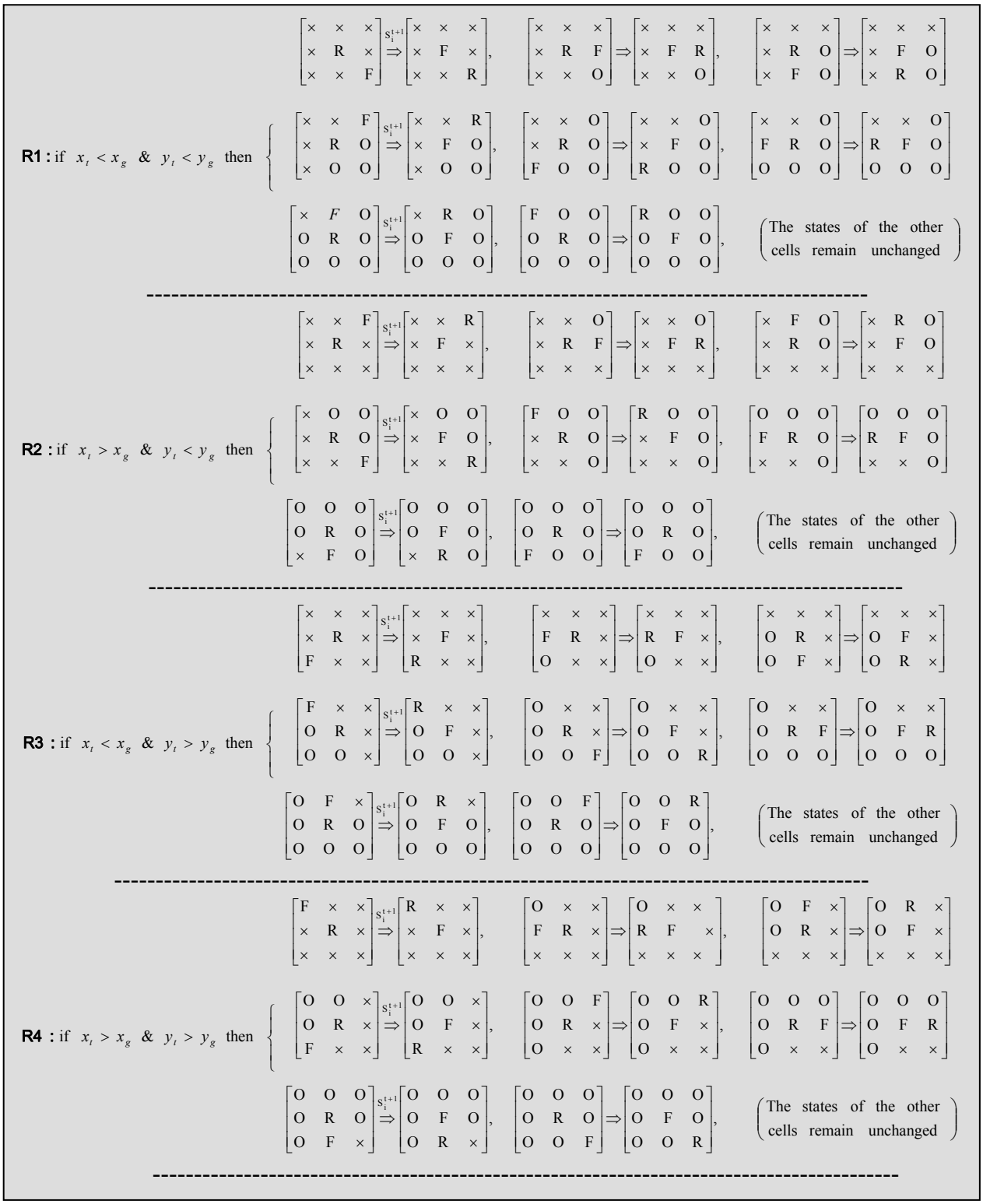
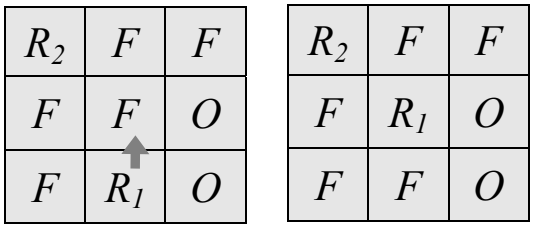


Fig. 3. Rules of the path planner cellular automata. At each time step t only one of the rules is active. In these rules, (x_t,y_t) is position vector of robot at time t and (x_g,y_g) is the position vector of goal cell. Symbol × denotes the cells those states do not important at time step t.



(a) Before applying the rules (b) After applying the rules for R_1

Fig. 4. The robots do not collide with each other in multi-robot problem

cell. In these rules Neumann neighborhood is used.

Lemma 1. *If obstacles are convex, the rules shown in Fig. 3 will direct the robot cell in CA to goal cell in a finite time steps.*

Proof. In each time step the robot cell moves into the free adjacent cell which is the nearest to the goal cell. Since there is finite number of cells in the lattice and there is no concave obstacle, after a finite number of time steps the robot cell would reach into the goal cell. □

2.2.3. Multi-robot problem

In multi-robot problems the number of robot cells and goal cells in CA will be increased. For example in a two-robot problem each cell can be in one of the states: $\{R_1, R_2, F, O, G_1, G_2\}$ where R_1 and R_2 stand for robot cells and G_1 and G_2 stand for goal cells. In multi-robot problem the rule base remains unchanged and at each step time the same rule base **(**including **R1**, **R2**, **R3** and **R4**) is sequentially applied to robot cells.

Lemma 2. *In multi-robot multi-goal problems, the rules in Fig. 3, by sequentially applying for all robots, will generate collision-free paths for robots from their initial positions to their goal positions.*

Proof. For moving each robot toward its own goal the same proof of Lemma 1 still holds. To confirm that the robots never collide with each other, without loss of generality consider two robots in Fig. 4. If the rules are firstly applied to R_1 in Fig. 4.a, it moves into a free cell, for example central cell in the figure. Now in Fig. 4.b. the central cell is not a free cell anymore and R_1 would not move into it. □

To conclude this section a pseudo-code of evolution of path planner automata is illustrated in Fig. 5.

```
while Robot cells R_k have not reached Goal cells G_k do
t=t+1
  for i=1:I
  for j=1:J   // for cell C(i,j)
    if S^t_{i,j} = F or S^t_{i,j} = O or S^t_{i,j} = G then
      S^{t+1}_{i,j} = S^t_{i,j}
    end-if
  if S^t_{i,j} = R_k then
     S^{t+1}_{i,j} = F
     (x_t,y_t)=coordinates of robot cell R_k
     //To update states of cells in the neighborhood of
     robot cell select rule base of Fig. 3 as the following:
     if      x_t ≤ x_g and y_t ≤ y_g then use rule R1
     elseif x_t > x_g and y_t < y_g then use rule R2
     elseif x_t < x_g and y_t > y_g then use rule R3
     elseif x_t > x_g and y_t > y_g then use rule R4
     end-if
  end-if
  end-for
  end-for
end-while
```

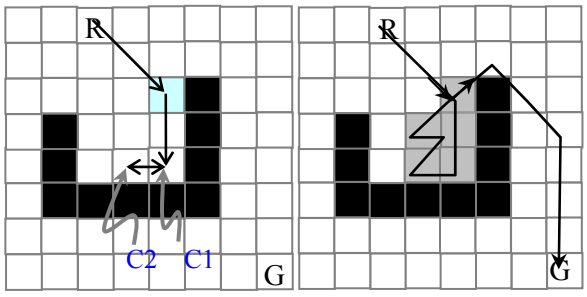
Fig. 5. Pseudo code of evolution of path planner cellular automata applicable for environments with convex obstacles.

3. Modification of Planning Method for Concave Obstacles Using Ant Colony Inspired Mechanism

3.1. *Single-robot problem*

In the planning algorithm introduced in Section 3 all obstacles assumed to be convex. To explain disadvantage of the algorithm in the presence of concave obstacles, consider the situation of Fig. 6 where the robot is initially located in cell R and goal is the button-right cell of the lattice. For all cells of the lattice we have $(x_t<x_g \ \& \ y_t<y_g)$; consequently among the rules of Fig. 3, **R1** should be used. According to rule **R1**, if encounters sub-block $\begin{bmatrix} R & O \\ O & O \end{bmatrix}$ in its neighborhood, the robot cell would move to a cell which is farther to the goal cell. Therefore, applying **R1** to the lattice, the robot cell travels along the path shown in Fig 6.a and moves back and forth between cells C1, C2. In fact the robot is entrapped in the concaveness.

In this section we use a colony mechanism to modify the planning algorithm in order to be applicable for concave obstacles. In our colony a large number of robots (agents), those may be considered as ants, collaborate with each other to find a path in the environment. The key idea is that vanguard agents identify and mark the concave regions for backward agents. A vanguard agent checks states of three nearest



(a) Applying CAs planner in Fig. 3. The robot is entrapped in the concaveness moving back and fro between C1 and C2

(b) If the robot leaves some pheromone on the cells inside the concaveness, it can escape toward its goal.

Fig. 6. Modification of CAs based planning algorithm for concave obstacles using an ant colony inspired mechanism.

cells to the goal in its neighborhood. If all of these three cells are obstacle cells the current cell is assumed to be in the concave region. When the vanguard robot finds itself in the concave region, it drops a certain amount of pheromone on its own location. Subsequently the following robots will behave *Pheromone cells* as *obstacles* and avoid them.

In order to add this idea into CA based planning method, without loss of generality, we consider the environment of Fig. 6. As it is explained before, in this lattice the only active rule base will be **R1**. We change the rules in **R1** in such a way that if the robot (agent) encountered the sub-block $\begin{bmatrix} R & O \\ O & O \end{bmatrix}$ it would drop pheromone on its current cell. This yields to the following rule:

$$\begin{bmatrix} \dots & \dots & \dots \\ F & R & O \\ \dots & O & O \end{bmatrix} \Rightarrow \begin{bmatrix} \dots & \dots & \dots \\ R & P & O \\ \dots & O & O \end{bmatrix} \qquad (1)$$

where state *P* denotes the *Pheromone cell*. At the next time steps, the agent(s) will behave the pheromone like an obstacle cell. Applying this modified rule base gives the path shown in Fig 6.b in which the pheromone cells are depicted in gray. It is obvious from the figure that after five time steps the robot leaves the concave region toward its goal.

It is noteworthy to mention that if the pheromone cells increase unlimitedly they may block the paths toward the goal. To avoid this event we can let the pheromones vapor in a specific rate, i.e. each pheromone cell exists only for limited time steps denoted by *v*.

The CA's rules in Fig. 3 may be modified according to the described colony mechanism to build the rules in Fig. 7.

Remark: There are two differences between the proposed colony and conventional ant colonies:

(1) In the proposed method the agents avoid going into the *pheromone cells* whereas in the conventional ones ants are interested to move in the paths with high density of pheromones.

(2) The agents in the ant colonies continuously deposit pheromone on their paths while in our mechanism pheromone only is dropped when the agents are on the concave regions.

3.2. *Multi-robot problem*

The modified planning method is much more superior while applying to multi-robot systems. If several agents (robots) start to move sequentially with a specific time delay τ, each agent independently performing the rule (1), smoother path is generated. To clarify the concept, consider a two-agent system as illustrated in Fig. 8. In Fig. 8.a the robot R1 has already left the concave region dropping pheromone inside it. Now, R2 behaves the pheromone cells as obstacles and does not move inside the concave region. In this figure the pheromone in cell C1 is dropped by R2[†].

Even if the time delay between two robots is not long enough to let R1 leave the concave region before R2 arrives, the path of R2 is still shorter and smoother than the path of R1 (See Fig. 8.b). Obviously in a three-agent environment, agent R3 will have much smoother path.

In summary, applying the modified algorithm to a single robot in online planning we will get a path like in Fig. 6.b. Applying the algorithm to a sequence of robots (agents) in online planning yields a path like in Fig. 8. The algorithm can also be applied to a single robot in off-line planning where some virtual agents adopted to perform the algorithms and get a smooth path for the robot.

Like in section 2, we have summed up the modified planning algorithm in a pseudo code shown in Fig. 9.

[†] There is no difference between pheromone cells created by R1 or R2. In the figures they are differently shown just for decipherability.

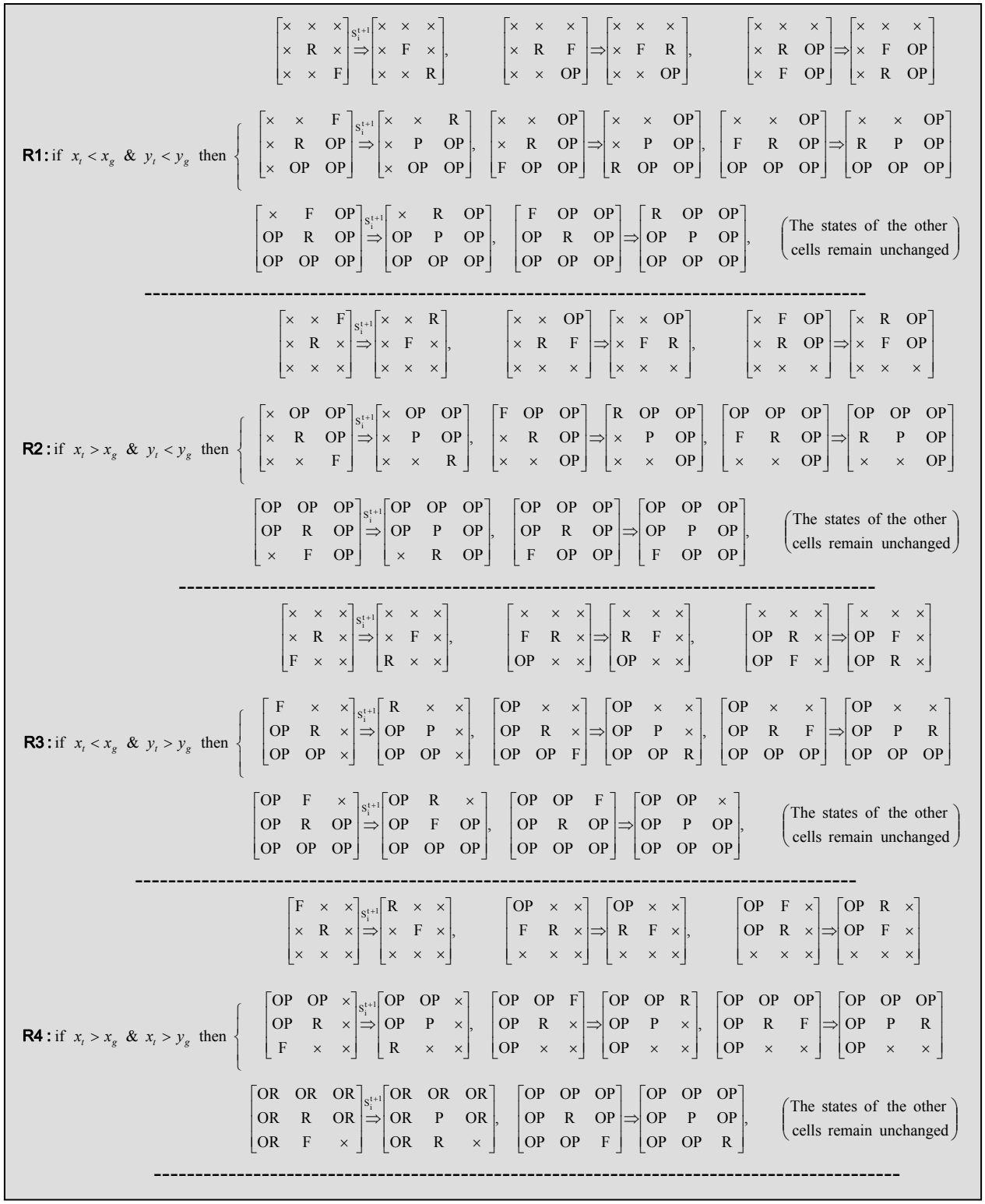
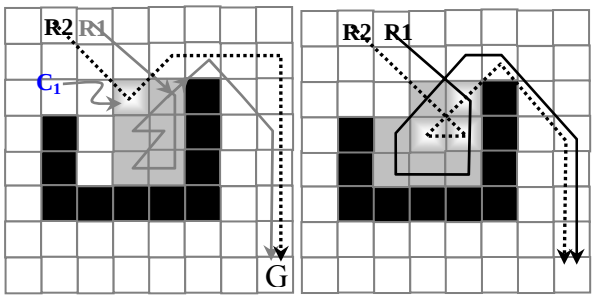


Fig. 7. The rules for the path-planner cellular automata modified by ant colony inspired method. At each time step *t* only one of the rules is active. In these rules, (x_t,y_t) is position vector of robot at time *t* and (x_g, y_g) is the position vector of goal cell. Symbol × denotes the cells those states do not important at time step *t* and OP means: *Obstacle cell* ∨ *Pheromone cell*



(a) R1 has left the concaveness before R2 arrives. (b) R2 starts its motion 3 time steps later than R1

Fig. 8. Applying the modified algorithm for multi-robot systems

```
while Robot cells R_k have not reached Goal cells G_k do
t=t+1
  for i=1:I
  for j=1:J   // for cell C(i,j)
    if S_{i,j}^t = F or S_{i,j}^t = O or S_{i,j}^t = G then
      S_{i,j}^{t+1} = S_{i,j}^t
    end-if
  if  S_{i,j}^t = P and S_{i,j}^{t-v} = P  then  S_{i,j}^{t+1} = F
    else  S_{i,j}^{t+1} = P    // v is vapor time
  end-if
  if S_{i,j}^t = R_k and  t > τ_k  then
      // τ_k is starting delay of robot R_k
      S_{i,j}^{t+1} = F
      (x_i,y_i)=coordinates of robot cell R_k
      //To update states of cells in the neighborhood of
      C(i,j), select rule base of Fig. 7 as the following:
      if      x_t ≤ x_g and y_t ≤ y_g then use rule R1
      elseif x_t > x_g and y_t < y_g then use rule R2
      elseif x_t < x_g and y_t > y_g then use rule R3
      elseif x_t > x_g and y_t > y_g then use rule R4
      end-if
  end-if
  end-for
  end-for
end-while
```

Fig. 9. Pseudo code of evolution of path planner cellular automata modified to be applied for environments with convex and concave obstacles.

4. A Layered Architecture to Implement the Path Planning Algorithm

4.1. *Platform selection*

To implement the planning algorithm, the robot needs an instrument to obtain its own coordinates at each time step. This instrument can be, for example, a dead reckoning tool. The robot also requires some sensors to recognize the state of adjacent cells. Sonar and IR proximity sensors are usual options for obstacle avoidance and low-resolution surface extraction. Moreover, they are potentially the best sensing modality for mobile robots. Therefore by mounting a rotary sonar sensor (or equivalently 8 fixed sensors) on the robot, a good sensing of the neighborhood would be available. On the other hand the robot must be able to read encoders in order to facilitate control of motors and update dead reckoning information in a feedback loop. Velocity control and position control are both desired in mobile robots. Power control of motors is also significant. In order to execute the path planning algorithm, a mobile robot must be capable of getting aforementioned sensory information and do appropriate control tasks. Here, it is preferred that the robot is autonomous i.e. capable of gathering information and carrying out tasks without programmatic involvement[26]. Autonomy involves complex sensing and planning operations on the part of the robot, including coordination of motor controls and planning.

There are variety of platforms developed for autonomously design and control of mobile robot. RAP, RWI, TeamBots, Xavier, BERRA and Saphira are the most well-known ones[27]. One can find a comparative study of these platforms in Ref. 27.

To implement our path planning algorithm, we employed Saphira (firstly introduced in Ref. 28) because of the following advantages:

- Saphira employs a layered abstraction approach which makes the complexity manageable[29].
- Saphira provides a geometric representation of the robot's environment which is very useful in path planning task.
- In control system, Saphira is based on some beliefs about the world those are represented in artifact. This feature is suitable to implement rules of the proposed path planning method.
- A high level scripting language (Colbert) is available in Saphira that highly improves productivity[27].
- Saphira lets the operator check some of the internal states and variables which is a very convenient characteristic[27].
- Saphira is a platform that brings up bindings for different languages and supports for multi-agent systems[27].

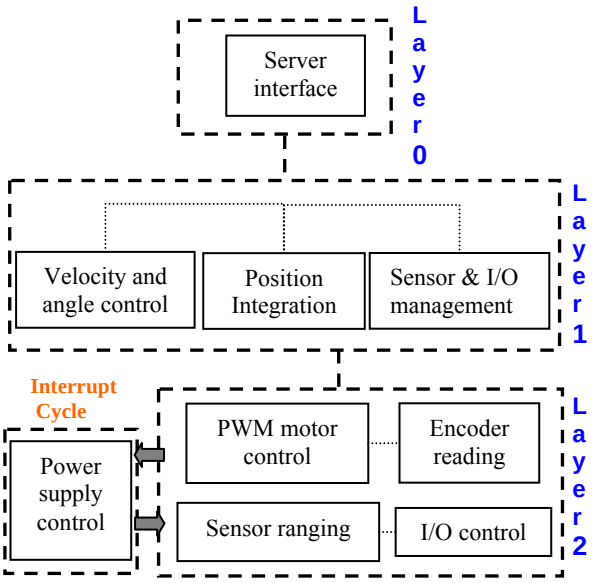


Fig. 10. Server operating system. Low-level tasks are motor control, sensor control and position integration. High-level tasks are encoder reading, PWM control and I/O control.

In the rest of this section we will describe the components of the Saphira and the architecture required for realization of the proposed algorithm.

4.2. *Robot server*

Central part of the architecture is robot server. The robot server includes two levels as shown in Fig. 10. In the lower level, the server performs some regular tasks like management of motor and sensor operations and updating of the robot's internal dead-reckoning position and power control.

In the higher level, the server implements a set of basic services including set point generation for velocity and direction, power control on a fast interrupt cycle supplied to the motors, inspection of encoder information and supervision of the sensors.

In Saphira, the program core is coded in the C language. A particular high-level language *Colbert* is also used in Saphira. It was firstly developed by Konolidge[30].

4.3. *Saphira architecture*

Saphira architecture is an autonomous multilayer sensing and control system appropriate for robotics applications. The architecture is an integrated mechanism for robot perception and action and runs a reactive behavior system with behavior sequencer.

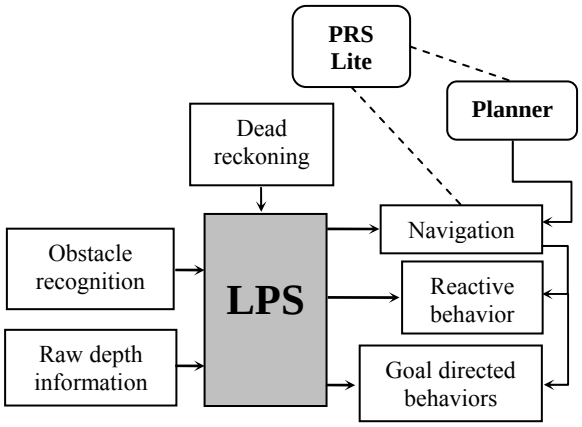
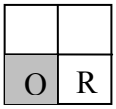


Fig. 11. *Saphira* system architecture. Perceptual bocks are on the left and action blocks are on the right. Control is coordinated by PRS-lite.

Some routines for sonar sensor interpretation, map building and navigation were integrated. At the central part it has a Local Perceptual Space (LPS) which provides a geometric representation of the robot's environment (see Fig. 11). Because of different representations for different tasks, LPS contains different levels of interpretation of sensor information as well as a priori information from sources such as dead reckoning tool. LPS gives the robot knowledge of its immediate environment and it is very significant in the activities like path-planning algorithms those require combination of sensor information and planning of local movement. LPS gives the *Saphira* architecture its representative rationality.

Another important feature in *Saphira* architecture is internal artifacts. *Saphira* has some beliefs about the world and these beliefs are represented in artifacts and most actions are planned and performed with respect to these artifacts. Our planning algorithm includes two groups of actions, high level actions and low level ones. The high level actions are goal-directed actions and include selecting the appropriate rule from *CA* rule-base. The low level actions are obstacle avoidance actions including the task of updating the states of the cells. Obstacle avoidance, as a low level action, sometimes necessitates that the robot perform a diagonal motion by a combination of two non-diagonal ones. For example in the following block the robot moves up then left in order to avoid obstacle *O*.



Selecting the appropriate rule is a reactive behavior and utilizes surface information and artifacts. At the task level complex behaviors are performed. Time critical behaviors such as updating of cell states are carried out by a very simple processing of the sensory information those are available quickly. At the control level, the *Saphira* architecture is behavior-based: the control problem is decomposed into small units of control task called basic behaviors.

In our system if the robot's current cell is the cell C*(i,j)*, obstacle-avoidance behavior might have the following variables indicating blocked directions in robot's path.

C (*i*-1, *j*-1) *is blocked*
C (*i, j*-1) *is blocked*
C (*i-1, j*) *is blocked*
C (*i*-1, *j*+1) *is blocked*
C (*i*+1, *j*-1) *is blocked*
C (*i, j*+1) *is blocked*
C (*i*+1, *j*) *is blocked*
C (*i*+1, *j*+1) *is blocked*

Obstacle avoidance routines must be written according to goal-directed requirements. For example, if the goal cell is on the right side in the bottom of the robot cell, the rule **R1** of the automata must be selected and we may have the following rules:

If ~ **(C**(*i*+1, *j*+1) *is blocked***)**
➔ **Exchange (***robot cell***, C**(*i*+1, *j*+1)**)**

If (C(*i+1, j+1*) *is blocked***)** & **~(C**(*i+1, j*) *is blocked***)**
➔ **Exchange (***robot cell***, C** (*i*+1, *j*)**)**

If (C (*i*+1, *j*+1) *is blocked***)** & **(C** (*i*+1, *j*) *is blocked***)** & ~ **(C** (*i, j*+1) *is blocked***)**
➔ **Exchange (***robot cell***, C** (*i, j*+1)**)**

If (C (*i*+1, *j*+1) *is blocked***)** & **(C** (*i*+1, *j*) *is blocked***)** & **(C** (*i, j*+1) *is blocked***)** & ~ **(C** (*i*+1, *j*-1) *is blocked***)**
➔ **Exchange (***robot cell***, C** (*i*+1, *j*-1)**)**

If (C (*i*+1, *j*+1) *is blocked***)** & **(C** (*i*+1, *j*) *is blocked***)** & **(C** (*i, j*+1) *is blocked***)** & **(C** (*i*+1, *j*-1) *is blocked***)** & ~**(C (***i*-1, *j*+1**)** *is blocked***)**
➔ **Exchange (***robot cell***, C** (*i*-1, *j*+1)**)**

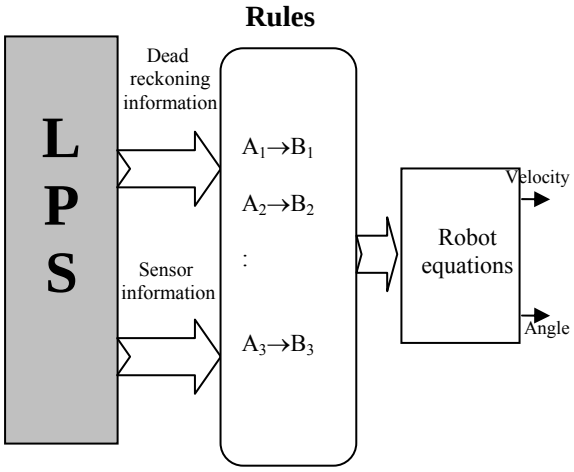


Fig. 12. Behavioral Rules provide true angle and velocity set points for the robot

Recall that exchanging the robot cell and a free cell means that the real robot moves to the free cell. Therefore the rules provide true angle and velocity set points for the robot as it is shown in Fig. 12. Obstacle avoidance tasks, those are reactive behaviors, often can take their input directly from sensors, probably after some transformation or filtering. Goal-directed behaviors can often benefit from using artifacts. Normally, artifacts in the LPS are reorganized based on the robot's dead-reckoning mechanism, which is reliable only over short distances.

4.4. *Control and Decision-making: PRS-Lite*

The control architecture of Saphira is composed of some routines that infer sensory information relative to the model of geometric world and some action routines that map the states to the actions. Localization routines link information of the local sensors of the robot into world's map and the *Colbert* sequences.

For the physical actions of the system, a low-level control is applied using artifacts. In the higher level, there is a requirement to associate behaviors with specific goals those the robot should complete. This management requires determination of the exact time to enable or disable behaviors as parts of a task as well as coordinating them with other activities in the system. PRS-Lite[31], a reactive controller based on the Procedural Reasoning System (PRS-CL), performs this role in *Saphira*[26].

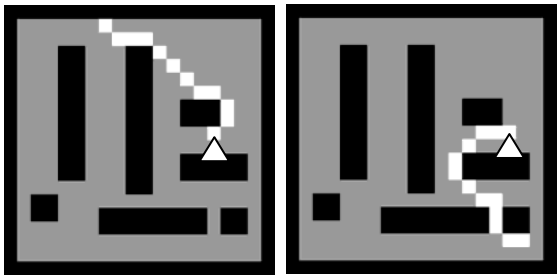


Fig. 13. Single robot moves in environment with convex obstacles. The path is planned for two different initial-goal sets using proposed CAs based algorithm. (Δ shows goal points)

4.5. *Communication*

In order to facilitate communication between the server and the robot, Saphira supports a packet-based communications protocol. Since the servo control of motors and sensors is done locally, the server can communicate with a robot via a low-bandwidth channel.

In the implemented system, the server sends an information package, containing information about the position, velocity and sonar sensor readings, to the robot every 100ms to update its set point variables and sensor schedules. A radio modem is used as data channel and to have a reliable communication, a checksum is done on packets to determine if the packet is damaged.

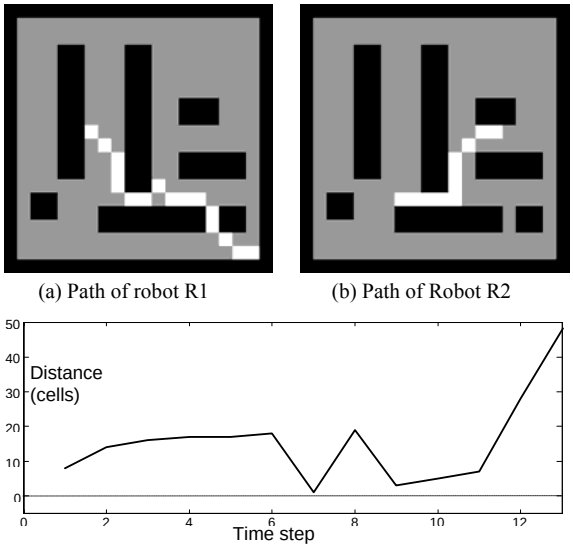
5. Simulation and Experimental Results

5.1. *Simulations of planning algorithm*

In the first step, we considered an environment with some convex obstacles. The CAs planning algorithm is applied for two sets of initial-goal points in a single robot environment. The planned paths (generated by simulations in MATLAB) are shown in Fig. 13.

Secondly in a two-robot environment the robots were initialized in two points of the work space. It is assumed that the robots have different goals. The generated paths by the proposed algorithm are illustrated in Fig. 14.a and Fig. 14.b. To confirm that the robots are not in the same cell at the same time, distance between robot cells in different time steps are shown in Fig 14.c. This distance never becomes zero which implies that the robots are not in the same cell at the same time.

In the third step, a maze with concave regions is selected. The proposed method is applied to a sequence



(a) Path of robot R1 (b) Path of Robot R2

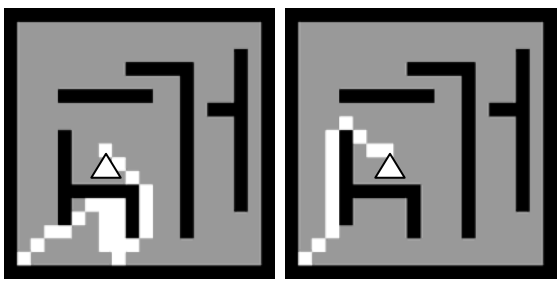
(c) Distance between robots R1 and R2 in terms of time steps. The robots are not in the same cell at the same time.

Fig. 14. Two robot problem

of robots considering τ=10 time step delays between agents (all starting from the same point). The vapor rate is set in such a way that pheromone disappears v=12 time steps after dropping. Fig. 15 shows the generated path for the first and forth robot. It is noteworthy to mention that in Fig. 15.b. the concaveness has already filled with pheromones. The proposed technique could successfully solve the path-planning problem.

5.2. *Implementation and experiments*

A differential wheel mobile robot was selected to perform the path-planner cellular automata. A rotary sonar sensor is implemented on the robot to detect states of adjacent cells. The *Saphira* architecture was implemented to make the robot autonomous. Servomotors (in wheels) are independently controlled



(a) Path of the first robot (b) Path of the forth

Fig. 15 Modified planning algorithm is applied to a sequence of robots in an environment with convex and concave obstacles. (Δ shows goal points)

by a PI controller.

The planning algorithm is embedded by the proposed architecture and the robot was tested in three different environments.

The first test corresponds to a single robot problem in an environment with convex obstacles. Fig. 16 shows four snapshots of the experiment. Generated path by the algorithm is also depicted in the snapshots.

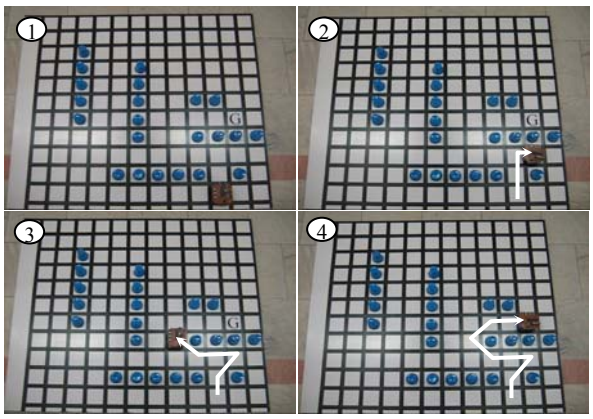


Fig.16. Four snapshots of a single robot test in an environment with convex obstacles. The generated path by the proposed algorithm is shown by white lines.

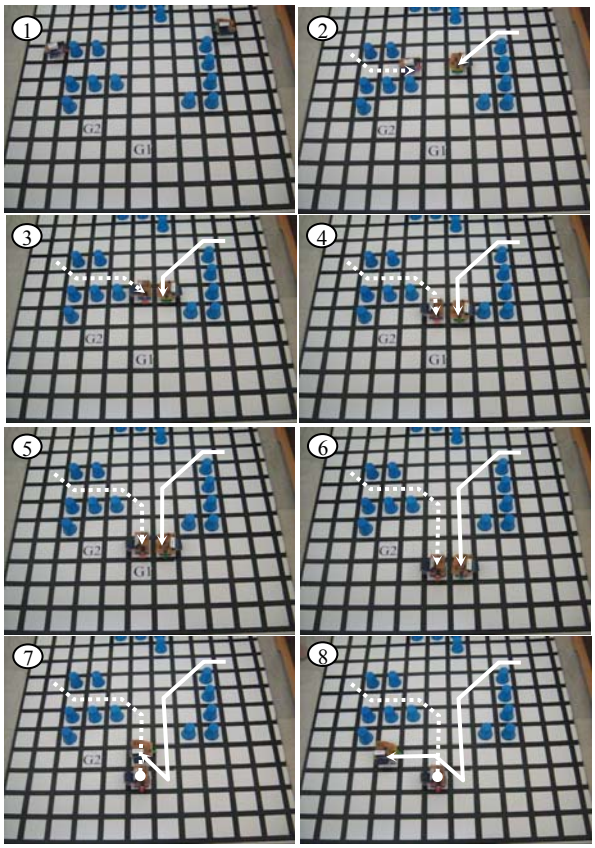


Fig.17. Eight snapshots of a two-robot test in an environment with convex obstacles. The generated path for R1 and R2 by the proposed algorithm is shown by dashed and solid white lines respectively. R2 adjusts its path in order to avoid collision with the robot R1

As second experiment, a two-robot problem with convex obstacles is investigated. The robots have different initial-goal sets. The experiment is designed in such a way that the paths of the robots are intersecting. It can be observed that the robot R2 adjusts its path in order to avoid collision with the robot R1. Eight snapshots of this experiment are shown in Fig. 17.

In the third test an environment with concave obstacles is considered. A single robot is assumed and its initial-goal cells is selected in such a way that the robot has to go inside the concave region. Fig. 18 shows eight snapshots of the corresponding experiment. It is clear from the figure that the robot has escaped from the concaveness and moved toward its goal.

6. Summary and Conclusions

Due to the advantages of CA in fast and reliable parallel computation and its local computation properties it was

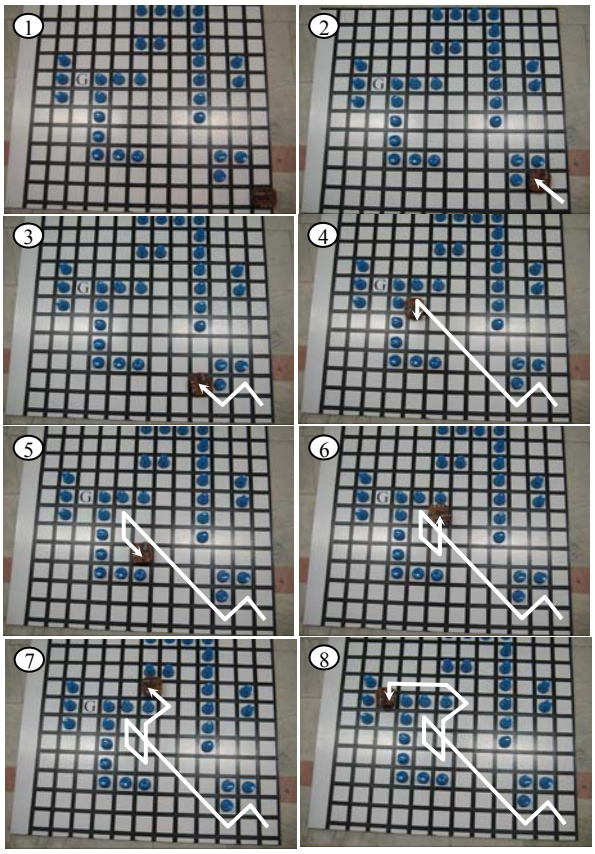


Fig.18. Eight snapshots of a single robot test in an environment with concave obstacles. The robot has escaped from concave region. The generated path by the proposed algorithm is shown by white lines.

selected as a tool to develop a path planning method for mobile robots. In the proposed method, the robot environment is considered as two dimensional automata with four states: Robot cell, Free cell, Obstacle cell and Goal cell. The robot (the agent) was assumed to have only a short sensing of its environment in order to detect adjacent obstacle cells. Evolutionary rules of the automata were developed in such a way that the robot progressively moves toward the goal. This yielded to an on-line planning method that could be easily extended for multi-robot environments. To this end, evolutionary rules should be sequentially applied to robots.

The algorithm was also modified using an ant colony inspired mechanism to be pertinent for concave obstacles. In this extension the agent drops some amount of pheromone on the cells inside the concaveness and marks them as Pheromone cells. Pheromone cells are behaved as obstacles in the succeeding time steps.

Finally a layered architecture called *Saphira* was used to implement the planning algorithm in an autonomous manner. *Saphira* is a multilayer behavioral sensing and control system to manage tasks of an agent. Eventually we came to an autonomously implemented mechanism for path planning of mobile robots in single and multi robot problems which is applicable for environments with both concave and convex obstacles.

References

1. J.Xiao, and L.Zhang, Adaptive Evolutionary Planner/Navigator for Mobile Robots, *IEEE Trans. Evolutionary Computation*, **1***(*1), (1997), pp. 18-28
2. J. C. Latombe, *Robot Motion Planning*, (Kluwer Academic Publishers, U.K. , 1996)
3. J. H. Reif., Complexity of the mover's problem and generalizations, *In Proc. IEEE Symposium on Foundations of Computer Science*, (Washington, DC, USA, 1979), pp. 421-427
4. J von Neumann, *Theory of Self-Reproducing Automata* edited and completed by A. W. Burks, (Urbana, IL: University of Illinois Press, 1966).
5. A. W. Burks, *Von Neumann's self-reproducing automata*, In Burks (1970a).
6. A W. Burks, *Essays on Cellular Automata*. (Urbana, IL: University of Illinois Press, 1970)
7. G. Doolen, *Lattice Gas Methods for Partial Diferential Equations*, (Reading, MA: Addison-Wesley, 1996)
8. Paul L. Rosin, *Training Cellular Automata for Image Pocessing,* (Lecture Notes in Computer Science, Springer Berlin, 2005)
9. S Murata, H Kurokawa, Self-reconfigurable robots, *IEEE Robotics and Automation Magazine*, **14**(1) (2007), pp. 71-78
10. X. Xiao, S. Shao, Y. Ding, Z. Huang and K.-C. Chou, Using cellular automata images and pseudo amino acid composition to predict protein sub cellular location, *Journal of Amino Acids*, **30**(1) (2006) pp. 49-54
11. J. Han, Y. Hayashi, X. Cao, H. Imura, Application of an integrated system dynamics and cellular automata model for urban growth assessment: A case study of Shanghai, *Landscape and Urban Planning,* **91**(3) (2009), pp. 133-141
12. I.G. Georgoudas, G.C. Sirakoulis, I. Andreadis, Modelling earthquake activity features using cellular automata, *Mathematical and Computer,* **46**(1-2) (2007), pp. 124-137, 2007
13. J. Maddox, The Universe as a fractal structure*. Nature,* **329**(17) (1987) pp. 195
14. T. Gramss, S. Bornholdt, M. Gross, M. Mitchell, and T. *Pellizzari, Computation in Cellular Automata: A Selected Review*, (Nonstandard Computation Wiley, 1998*)*, pp. 95-140
15. C. Shu and H. Buxton, Parallel path planning on the distributed array processor, *Parallel Computing* **21**(10) (1995)*,* pp. 1749-1767
16. P.G. Tzionas, A. Thanailakis, P.G. Tsalides, Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata*, IEEE Tran. on Robotics and Automation,* **13**(2) (1997), pp. 237-250
17. C. Behring, M. Bracho, M. Castro, J. A. Moreno, and Emergente, An algorithm for robot path planning with cellular automata*, In Proc. of the Fourth Int. Conf. on Cellular Automata for Research and Industry*, (Karlsruhe Germany, 2000), pp. 215-235
18. F. M. Marchese, A directional diffusion algorithm on cellular automata for robot path-planning, *Future Generation Computer Systems,* **18**(7) (2002), pp. 983–994
19. F. M. Marchese, Multiple Mobile Robots Path-Planning with MCA, *In the Proc. of Autonomic and Autonomous Systems, ICAS '06,* (Silicon Valley, CA, 2006), pp. 56-65
20. A. Akbarimajd, C. Lucas, On-Line Path-planning Technique for Mobile Robots Using Cellular Automata, *In the Proc. of Int. IEEE/APS Conference on Mechatronics and robotics*, (Aachen, Germany, 2004), pp. 478 - 482
21. A. Akbarimajd, C. Lucas, A Colony Included Cellular Automata for Revolving the Concave Obstacles in Path-planning of Mobile Robots, *In the Proc. of Int. Conf. on Advances in Intelligent Systems-Theory and Applications,* (Luxembourg, 2004), pp. 124-129
22. A. Akbarimajd, C. Lucas, A New Architecture to Execute CAs-Based Path-Planning Algorithm in Mobile Robots, *In the Proc. of IEEE Conference on Mechatronics*, (Budapest, Hungary, 2006), pp. 478-482
23. A. Akbarimajd, A. Hassanzadeh, "A Novel Cellular Automata Based Real Time Path Planning Method for

Mobile Robots", *Int. Journal of Engineering Research and Applications*, **1**(4) (2011), pp.1262-1267

24. E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*., (NY: Oxford University Press, New York, 1999)
25. M. Dorigo, G. Di Caro and L. M. Gambardella, Ant algorithms for discrete optimization, *Artificial Life,* **5**(2) (1999) pp. 137-172
26. K. Konolige, K. Myers, E. Ruspini, The Saphira Architecture: A Design for Autonomy, *Journal of Experimental and Theoretical Artificial Intelligence* ***9*** (1997), pp. 215-235
27. A. Oreback, H. I. Christensen, Evaluation of Architectures for Mobile Robotics, *Autonomous Robots,* **14**(1) (2003), pp. 33–49
28. A. Saffiotti, K. Konolige, and E. H. Ruspini, A Multivalued Logic Approach to Integrating Planning and Control, *Artificial Intelligence* **76**(1-2), 1995.
29. J. Connell, SSS: A Hybrid Architecture Applied to Robot Navigation, *In the Proc. of the IEEE Conference on Robotics and Automation*, (Nice, France, 1992), pp. 2719 - 2724
30. K. Konolidge, COLBERT: A language for reactive control in Saphira. *In the Proc. of German Conference on Artificial Intelligence*, (Freiburg, Germany, 1997), pp. 31-52
31. N. J. Nilsson, Teleo-Reactive Programs for Agent Control, *Journal of Artificial Intelligence Research Vol. 1*, 1994