Chapter 7
Multi-Resolution Hierarchical Motion Planner for Multi-Robot Systems on Spatiotemporal Cellular Automata

Fabio M. Marchese

Abstract This chapter presents a new multi-resolution and hierarchical approach to the problem of motion planning of Multi-Robot Systems on discretized spaces. The goal is to operate on large spaces (compared to the size of the robots), where the number of cells quickly becomes untreatable, in particular for n interacting robots problems, without losing precision (resolution). To work around this problem, we have introduced 3 levels of maps: the first is topological, the second a rectangular tessellation covering the free space, and the third a regular (small) cells decomposition. The first two maps are used to reduce the problem and to simplify it with non-accurate planning. Limiting the search space to smaller areas of interest at the last level and considering the interactions between robots, precise parallel motion planning is performed using Spatiotemporal Cellular Automata.

7.1 Introduction

This chapter concerns the coordinated motion of a Multi-Robot System (MRS) over wide spaces. We have considered MRSs composed of heterogeneous mobile robots having generic shapes, sizes and kinematics. In particular, we are interested in the coordination of a team of mobile robots moving over 2D structured environments (2D manifolds). Typical applications include the CMOMMT problem [18] related to surveillance tasks, as well as cooperation issues in soccer teams (e.g., RoboCup) or transportation of parts in manufacturing plants (logistic), rescue robots and equivalent problems. Over the years, various planners have been implemented on continuous spaces. Despite their interesting properties, they are prone to the problem of using high time consuming algorithm. We want to tackle an issue like extended spaces or, conversely, small models with very high-precision representation as an unique

F.M. Marchese (✉)
Università degli Studi di Milano-Bicocca, DISCo, V.le Sarca 336, Milan, Italy
e-mail: marchese@disco.unimib.it

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_7

problem. It is necessary to reduce calculation times throughout two solutions: (i) the adoption of models on discrete spaces (lattices or grids); (ii) the reduction of the complexity of the planning algorithm to reduce the search space. Starting from a 2D metric representation of the environment, in a completely automatic way, maps at different resolutions level are calculated, finally generating a topological space representation (graph of adjacent regions). The aim is to reduce the complexity of the problem on large lattices, in order to make them more manageable. Therefore, we realized a multi-resolution system, starting from a high-resolution map (regular cells decomposition). Then, rectangular regions of different sizes (clusters of cells) are recognized, generating an irregular decomposition or tessellation, but still able to completely cover the Free Space. It is similar to generalized cones by Brooks [4], where rectangles are considered as a degenerate case of cones. From this map, we generated a graph of adjacent rectangles and finally a graph of passing zones, which are used for a topological planning. Conceptually, we have designed a two-level hierarchical planner:

1. High level planner (Gross motion planning). This planner works on a topological map, the only purpose of which is to identify a 'channel' the robot moves through. This planning phase does not take into account the time level, but it works exclusively at the space level (geometric level, hence only trajectories). The channel delimits the area in which to look for a precise spatial trajectory (and later on a spatiotemporal trajectory, i.e. a movement).
2. Low level planner (Fine motion planning). It is a spatiotemporal CA (STCA) planner working on a precise environment map delimited by the above mentioned 'channel'. It coordinates the necessary maneuvers to avoid obstacles (in particular to avoid moving obstacles, i.e. other robots), while taking into account shapes, sizes, orientations, kinematics of robots and temporal constraints (departure time and arrival time).

The rest of the chapter is organized as follows: in Sect. 7.2, the MRS Motion Planning Problem is introduced. Sections 7.3 and 7.4 are related to Fine and Gross Motion Planning. Section 7.5 concerns to an example of MRS problem and in Sect. 7.6 the conclusions are presented.

7.2 MRS Motion Planning Problem

A Multi-Robot System is a set (a team) of robots sharing a common task. To solve the same task, the robots often use the same resources. In the same workspace, several MRS with different tasks can coexist, but they still compete for the same resources. In particular, while moving, they compete for the access to the 'Space' resource. In this work, we address the problem of the coordination of motion for a set of heterogeneous mobile robots in order to avoid collisions with static and dynamic obstacles. Since the late 70s, many approaches have been proposed for the solution of the Path/Motion Planning problem for single and multiple robots. A Configuration Space (C-Space)

solution has been proposed since 1979 [13] where a geometrical description of the environment was given. To address the problem in a dynamical world, some authors proposed the Artificial Potential Fields Methods as in [9]. Jahanbin and Fallside introduced a wave propagation algorithm in the C-Space on discrete maps (Distance Transform [8]). In the 90s, Barraquand et al. used the Numerical Potential Field Technique over the C-Space to build a generalized Voronoi Diagram [1]. In 1994, Zelinsky extended the Distance Transform to the Path Transform [22]. Since 1990, Warren in [21] used a discretized 3D C-Spacetime (2D Workspace + Time) for multiple translating robots' motion planning with simple shapes (only square and circle). In the same decade, the Cellular Automata approach for the single robot path-planning problem was introduced [2, 14, 20]. In [11], the authors applied the concepts of Game Theory and multiobjective optimization to the centralized and decoupled multi-robots motion-planning. A solution in the C-Spacetime has been proposed in [3], where the authors use a decoupled and prioritized path-planning in which they repeatedly reorder the robots' priorities to try to find a solution. In the 2000s, Cellular Automata started to be also used for the coordination of multiple robots [7, 16].

7.3 Fine Motion Planner

The Fine Motion Planner calculates the precise collision-free trajectories and movements that a team of robots must follow to reach their goals. Although it is driven by the Gross Motion Planner, it is the most important phase because it finalizes the method. It relies on a set of discretized metric spaces (see 7.3.1), the most important of which is the cellular space STCA. This planner uses the approach of Artificial Potential Fields, i.e. force fields that drive the robots toward their given goals.

7.3.1 Spaces

In this section, the spaces used by the fine motion planner are described. They are multidimensional Cellular Spaces (up to 5 dimensions) and all of them are projections of the underlaying discretized version of a 2D manifold in the real Space where the robots moves. All these spaces participate, by means of the evolutions of their automata, to the computation of the planner. In particular, the Attraction Space finalized the contribution of the other spaces, generating the final motion of the MRS.

7.3.1.1 Spatiotemporal Cellular Automata

All the spaces used derive from the definition of discretized spacetime. It is a multidimensional cellular space, i.e. a discretized version of the continuous linear

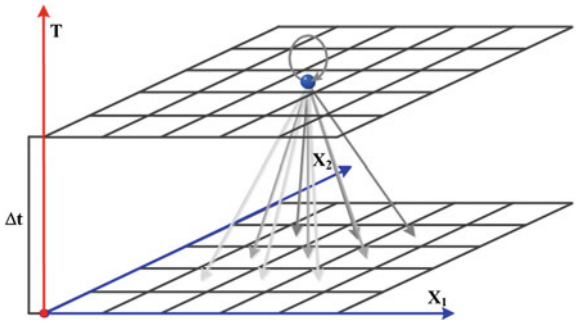

Fig. 7.1 Spatiotemporal cellular automata: dependencies between CA in adjacent layers and itself (radius 1)

(Euclidean) Spacetime, where some spatial dimensions are compounded with the Time's Arrow. Each dimension is discretized in indivisible quanta that represent the lower limit of both spatial and temporal resolution. This discretization induces a partition of multidimensional space in basic cells. Depending on the type of spacetime represented, each cell contains a value/state (even a vector of states) that is calculated on the basis of homologous values in adjacent cells with which it interacts. It is therefore a Cellular Automata, composed of different temporal layers. Each layer contains the representation of the whole workspace at a certain moment. Its peculiarity lies in the mapping of physical Time, not to be confused with Computing Time and Simulation Time. While physical Time is represented as a dimension of Spacetime, and therefore it becomes a static component, the Computation Time is the time necessary to upgrade the cells of STCA, and finally the Simulation Time is the passage from one temporal layer to the next of the STCA to follow the evolution of the system (the robots' movements). Since, as is well known, Time flows in one direction, to satisfy this property and then the physical feasibility, the Cellular Automata in a layer depend exclusively on Cellular Automata of the layer beneath (even more than one layer) at the same location or in the spatial neighborhood of the cell. It depends also on its own state, but it does not depend on automata of the same temporal layer or on those at successive times. Thus the number of total dependencies is reduced and, accordingly, computation time. For example, in a 3D Spacetime, the total number of dependencies from any other cell (neighborhood radius 1) would be 27, but thanks to the direction of Time, they are only 9 + 1 (Fig. 7.1). This condition is necessary to ensure the feasibility of the system, in both cases if it represents a real or a pure mathematical transformation.

7.3.1.2 C-Spacetime Space

When a layer of the STCA represents the configurations of a physical system, it becomes a discretized C-Spacetime. Please note that a configuration is a set of parameters representing the state of a system (in other words, the robot is represented as a point in the space). In our case, the configurations are given by the poses of

each robot. Because robots are considered as rigid bodies, a limited number of parameters represents their states: their positions and orientations. For motions on 2D spatial manifolds, we use (X, Y) for the position + Θ for orientation to define the Configuration Space. The overall C-Spacetime will be 4D: 2D (position) + 1D (orientation) + 1D (time). In the C-Spacetime, obstacles are mapped as not admissible configurations, that is configurations the robot cannot assume at a given instant of time without colliding with them (but which can be assumed in other moments if the obstacles move away).

7.3.1.3 Coordination Space

The Coordination Space is a unique space for all the MRSs, where the interaction robot-robot and robot-obstacle are modeled. It is the space in which the robots are coordinated in their movements, i.e. where to monitor the interactions between various objects (static and dynamic) to prevent their movements that lead them to collide. It is a representation of the real Spacetime in a discretized form, to manage movements on a 2D manifold. It is a 3D space: 2D (position) + 1D (time). In this space all static obstacles (e.g., walls) and dynamic obstacles are mapped, including moving objects (with a priori known movement) and all robots. All their movements are computed by the planner, in a similar way as in the discretized C-Spacetime, but the robots and all the other objects are fully represented, time by time, with their real shapes and extensions. It replaces what was once known as Repulsive Space, where artificial forces where evaluated to keep the robot away from the obstacles. In the Coordination Space, the cells are either occupied or free depending on whether an object at a certain moment covers that position. In order to satisfy Shannon's Theorem, Time is sampled at double frequency, i.e. the time unit used is half of the time unit used in any other spaces to prevent thin objects from passing through each other.

7.3.1.4 Attraction Space

The Attraction Space is a 5D STCA, where the configurations of all robots are represented. It is ultimately a 4D C-Spacetime for each robot, developed along the Robot Dimension, i.e. in each layer the C-Spacetime of a single robot is evaluated. Each automaton has a non-stationary potential value (it evolves along the Computational Time, but it is stationary in its representation in the Spacetime). This potential is referred to a lower potential located in the cell of the goal configuration. As usual, whenever there is a potential field, a force F appears ($-grad\phi$). In this metaphor, this virtual force attracts the robot towards the goal, turning around the obstacles. A potential value represents the integer 'distance' of the current cell from the goal cell along the shortest collision-free path (more simply: it is the minimum cost to reach the goal from the current cell avoiding all obstacles). The potential is updated with respect to the neighbors' values to reach a steady value. The updating rule is quite

simple: the potential value is the minimum value of the potentials of the neighbors plus the cost of the move that carries the robot from the neighbor cell to the current one. Within a temporal layer, the potentials are calculated up to their stability, before the successive layer is updated on the base of the current one (the temporal layers are sequentially updated). At the end, all temporal layers are stabilized and the 4D space contains a 'potential bowl' with a minimum at the goal pose. The CA updating rules ensure that the potential is calculated by turning around obstacles, and with a single minimum, i.e. no local minima are generated (in which the robot would stall). Within the Attraction Space all the movements are represented which may lead to the goal pose from any starting pose, under the form of spacetime trajectories (geometrization of movements). The calculation of the movements (ST trajectories) is made according to the negated gradient of the potential surface. The robot is shrunk to a single point, which moves on (rolls over) a potential 4D hyper-surface towards the goal. In order to take into account the actual size of the robot (which also changes with its orientation), it relies on the Coordination Space to validate the moves set, temporarily deactivating the moves which would lead to a collision (see 7.3.2.2).

7.3.2 Motion and Moves

Planning is the results of the interaction between two aspects: on one hand the search spaces (7.3.1), on the other hand the model of the process. In this section, it is introduced the kinematic model of a generic robot. A new 'dynamical' model of the robot results from combining the robot's kinematic model with its shape, called *Motion Silhouette*.

7.3.2.1 Spatiotemporal Moves

Robot discrete kinematics is described by a set of atomic spacetime moves, also including (non-)holonomic constraints. All objects, including static objects, follow a temporal trajectory (movement) consisting of a sequence of spacetime moves. In a discretized spatiotemporal space Z^4 for a robot moving on a 2D manifold, the definition of spatiotemporal move is the 4-tuple: $(\Delta x, \Delta y, \Delta\theta, \Delta t)$, where Δ is a finite variation, $(\Delta x, \Delta y) \in Z^2$, $\Delta\theta \in S^1$, $\Delta t \in Z$ and with the obvious constraint $\Delta t \geq 0$. This definition has two main interpretations: $(\Delta x, \Delta y, \Delta\theta)$ are finite increments of the spatial coordinates during the finite time interval Δt.

It entails the following space metrics Δs:

$$move \cong \frac{(\Delta x, \Delta y, \Delta\theta)}{\Delta t} \quad \Rightarrow \quad \Delta s^2 = \Delta x^2 + \Delta y^2 + r^2\Delta\theta^2$$

In the same way, $(\Delta x, \Delta y, \Delta\theta, \Delta t)$ are finite increments of the spatiotemporal coordinates, inducing the following metrics:

$$move \cong (\Delta x, \Delta y, \Delta\theta, \Delta t) \quad \Rightarrow \quad \Delta S^2 = \Delta x^2 + \Delta y^2 + r^2\Delta\theta^2 + v_r^2\Delta t^2$$

where ΔS is the 'distance' between two events of the spacetime, r is a dimensional constant, v_r is the 'speed' of spontaneous translation along the Time axis.

Static objects only have a single type of motion parallel to the Time's Arrow. They have only one move (we call it 'existence rule'): (0, 0, 0, +1). This move is part of the kinematics of all robots, without which the object does not have a physical sense (it should move indefinitely, something like a *perpetuum mobile*, never standing in a place).

Thanks to this definition of spatiotemporal move, the movement of a rigid body becomes a geometrical trajectory in the spacetime executed by means of a sequence of finite moves. The speed is computed as usual, but it is a rational value. For example, (+2, 0, 0, +1) is a move with double speed (x direction) or (+1, 0, 0, +2) represents a move at a half speed with respect to normalized units. Using an appropriate set of moves, it is possible to represent every type of kinematics of any robot. Furthermore, there is no theoretical limit to represent any velocity.

7.3.2.2 Motion Silhouette (Moves Validation)

To face with real robots in real environments, it is necessary to represent the shape of a robot. In the Configuration Space, the robot is represented only by a point (a vector of coordinates), but its real extension is not considered. In the late 70s, Lozano-Pérez first introduced the concept of obstacles enlargement to take into account the actual footprint of a robot [13]. In the first method, enlargement was isotropic, by an amount equal to the maximum radius of the robot (Fig. 7.2a). This method clearly led to an excessive enlargement because it transformed the robot shape to a cylinder that surrounded it. The result was a great loss of available free space and the closure of narrow passages. At the beginning of the 80s, Lozano-Pérez et al. [12] proposed a better solution: they also took into account robot orientation (Fig. 7.2b), realizing n obstacles maps, one for each expected orientation and with a different obstacles enlargement (anisotropic enlargement). This was translated in the C-Space as zones with admissible configurations and areas with forbidden configurations.

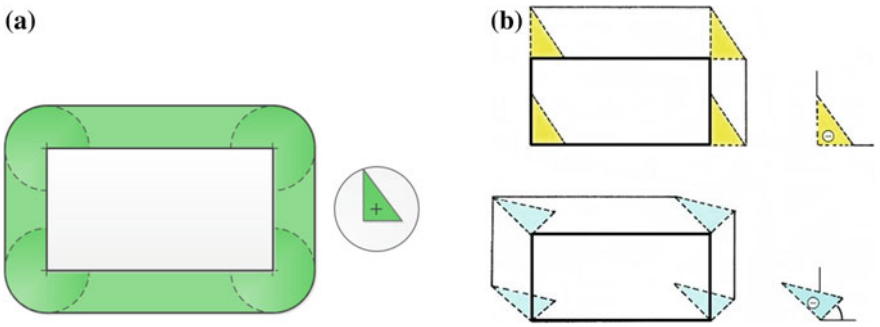

Fig. 7.2 Obstacle enlargement: **a** isotropic [12]; **b** oriented [13]

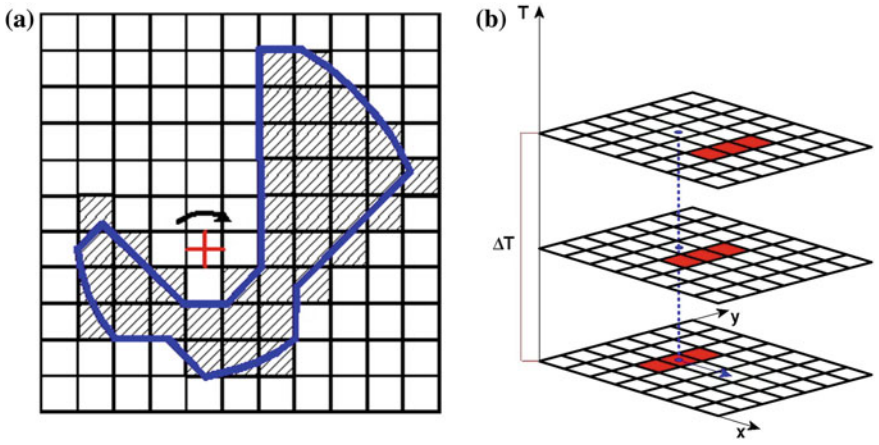

Fig. 7.3 **a** Sweeping silhouette [15]; **b** Motion silhouette [17]

In 2003, the authors proposed a method which takes into account what happen between two different orientations: the Sweeping Silhouettes [15]. Instead of widening the obstacles, the Sweeping Silhouettes of the robot were used on discretized C-Space cells to determine which configuration have to be considered admissible. This method makes it easier and more accurate to determine prohibited configurations, eliminating even the configurations in which the robot does not collide with obstacles just in the start and arrival poses of a single move, but also in all the intermediate positions, avoiding problems of 'tunneling' thin object like in Fig. 7.3a. The next evolution occurred in 2011 with the Motion Silhouette [17]. It coincides with the use of discretized C-Spacetime, strictly necessary when planning in the presence of moving obstacles (in particular other robots). With a double sampling of Time (to satisfy the Shannon's Theorem) a sequence of silhouettes are generated to map an elementary move. A collision test occurs within temporal layers of the Spacetime with respect to the poses of other obstacles. The prediction of a collision in a temporal movement step allows to disable the move starting from that particular robot pose (Fig. 7.3b). The test is performed for each pose and for each instant of time to validate the set of available moves (i.e. kinematics), temporarily disabling those which lead to a collision. Motion Silhouette method is able to represent any robot shape of any size. The only limit is due to the limit of the representation. It is a discretization of a 2D shape, thus the accuracy of the representation depends on the real size of a cell.

7.3.3 Interaction Between Spaces and Moves (Planning)

Conceptually, the motion planning is the result of the interaction between the Spaces and the set of Moves (kinematics) of a robot, i.e. depending on the move selected, spaces are altered accordingly.

7.3.3.1 Prioritized Planning

While the path planning problem for a single robot has a polynomial complexity, in 1979 Reif [19] established that the problem for a team of robots is PSPACE-hard, which implies it is NP-hard. Canny later established that the problem lies in the PSPACE, and therefore the general motion planning problem is PSPACE-complete [5]. Even Warehouseman's problem on a 2D grid is PSPACE-hard [6]. In general, for a N rigid robots problem the number of dimensions of the C-Spacetime would be: $C_{ST} = C^1 \times C^2 \times ... \times C^N \times T$. If robots are moving on a 2D manifold, the C-Space of a single robot is $\Re^2 \times SO(2) \equiv \Re^2 \times S^1$, thus the overall C-Spacetime has $3N + 1$ dimensions. Even for small MRS, the cardinality of the space makes the problem untreatable. Therefore, it is necessary to reduce the number of dimensions, for example adopting the Prioritized Planning technique (a description in [10]). This technique is a case of Decoupled Planning for multiple robots, where the robot to robot interaction is ignored in the first phase of motion design. Then the interactions are taken into account to constrain the available options. The problem arises when no option remains because this approach is not reversible, thus losing the completeness. Nevertheless, Prioritized Planning is very practical and solves most of the situations. In the prioritized approach, a planning order is given, starting to plan with the high-priority robot first. Robots with a lower priority see the higher-priority robots as moving obstacles with designed spatiotemporal trajectories. The planning phases are:

1. Establish a priority order for the robots (ordered set).
2. Plan the motion for the robot with the highest priority not yet planned (single robot motion planning in its Attraction Space).
3. Using the Coordination Space, select one collision-free movement from the set of all movements found (if any).
4. If there is at least one movement, trace the robot in the Coordination Space (mark the configurations along the movement as no longer available: the robot becomes an obstacle for all the other robots with a lower priority).
5. Exit if all the robot has been planned (finding or not a movement).
6. Goto step 2.

If no collision-free movement can be found for a robot, many recovery strategies can be adopted. The simplest one is to eliminate the robot from the space (and from the problem). Another strategy is to consider the robot as a static object standing in the starting pose. A more complex strategy is to let the robot at the starting pose, with the task to stay in the same pose, but reducing its priority. In this case, the robot can move away if another robot (with a higher priority) has to pass in that pose, and then it goes back to the initial pose. It is quite hard to define the complexity of this algorithm. It has been established [14] that, for a single robot, in the worst cases the complexity is $O(N^2)$, where N is the number of cells of the discretized C-Space. Hence, for an algorithm using the Prioritized Planning, its complexity is $O(pN^2)$, where p is the number of robots of the MRS.

7.3.3.2 T-Invariant Motion Planning

Dual kinematics is defined as a set of dual moves. These are obtained by the original kinematics by reversing the sign of all the components of the initial moves:

$move \cong (\Delta x, \Delta y, \Delta\theta, \Delta t) \quad dualmove \cong (-\Delta x, -\Delta y, -\Delta\theta, -\Delta t)$

The dual moves still belong to the same set of the original moves, and the dual kinematics is again a 'normal' kinematics. The original problem was a classical top-down planning (or backward planning: from goal to the start); the dual problem is a bottom-up planning (or forward planning: from start to goal). By exchanging the start with the goal and dualizing the kinematics we can solve the dual problem using the same algorithm. Hence, because they have the same solution, they belong to the same class of motion planning problems. It is possible to plan a movement from the goal to the starting pose, or viceversa from the starting pose to the goal, and we will find the same solution. We call this property as T-invariance of Motion Planning.

7.4 Gross Motion Planning

The approach to multidimensional Cellular Automata suffers from the problem of the appreciable increase of the computational burden, which grows with the number of dimensions of the Spacetime and the number of robots. Let's suppose we model a space with 100×100 cells, with a discretization of the orientation every 3 degrees (120 cells along the orientation axis), with 50 ticks along the time axis (100 cells for double sampling) and finally using 10 robots. The overall Attraction Space easily exceeds the billion of cells needed to manage the MRS (the Coordination Space is limited to only 10^6 cells). Fortunately, the Attraction Spaces of each robot are handled separately, but anyway each one reaches 10^8 cells. It is important to reduce its complexity. To do this, we add a preprocessing phase: the gross motion planning. Its purpose is to identify 'coarse' channels within which the robot will move. Then simply generate a set of possible trajectories (inside the channel) not taking into consideration the travel times. The aim is to discard all regions in the environment through which the robots do not pass and limit the precise motion planning only to the 'touched' areas, thus significantly reducing the number of cells really involved.

7.4.1 Topological Approach

To drastically reduce the computation time, we decided to adopt a static topological approach (not considering the time at this phase). From the 2D grid map of the environment, through appropriate processing, we generated a weighted graph representation of space (topological map). Topological map building occurs in three

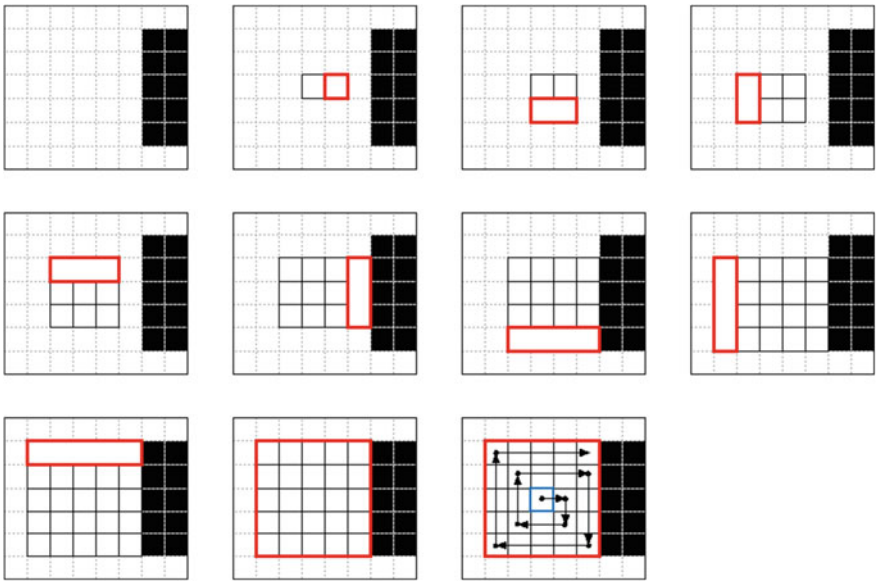

Fig. 7.4 Example of clockwise spiral cells clustering

steps:

1. free space decomposition in rectangular regions (complete coverage) through a mechanism of aggregation of cells;
2. search for border areas between regions;
3. construction of topological weighted graph.

7.4.1.1 Cells Clustering (Rectangular Coverage)

The process of decomposition of the free space in rectangles is performed by clustering adjacent cells in a spiral pattern (Fig. 7.4). It identifies a cell seed not yet part of any rectangle. This cell represents a basic 'rectangle' from which to start an enlargement phase by aggregation. It proceeds with a clockwise spiral by adding a row of cells (a rectangle of one cell thick) to the previous rectangle, getting a new wider rectangle. The result is always a rectangle, i.e. a convex region (the rectangles set is closed under this operation). The procedure continues while it finds new lines of cells that do not overlap with obstacles, or with the map boundary, or with other rectangles already identified, or until it reaches a maximum width.

At this point, it tries with a new cell not yet used and repeats the clustering. The procedure terminates when all cells are part of a rectangle (red in Fig. 7.5a) having the complete coverage of the space.

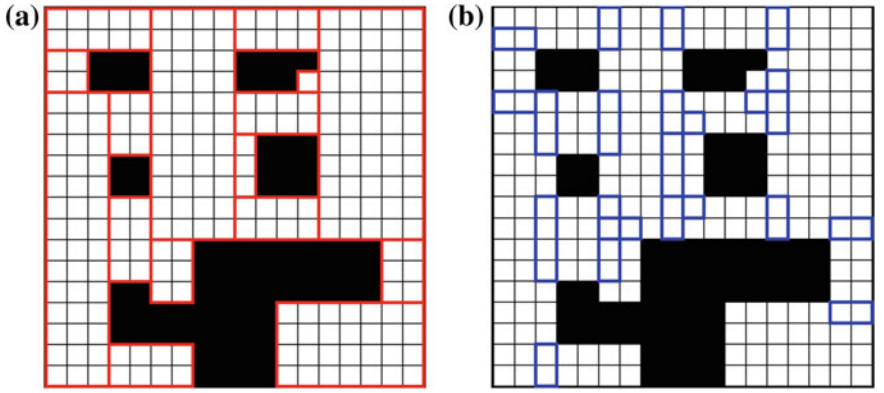

Fig. 7.5 Cells clustering: **a** *Rectangular* tessellation; **b** *Border rectangles* ('doors')

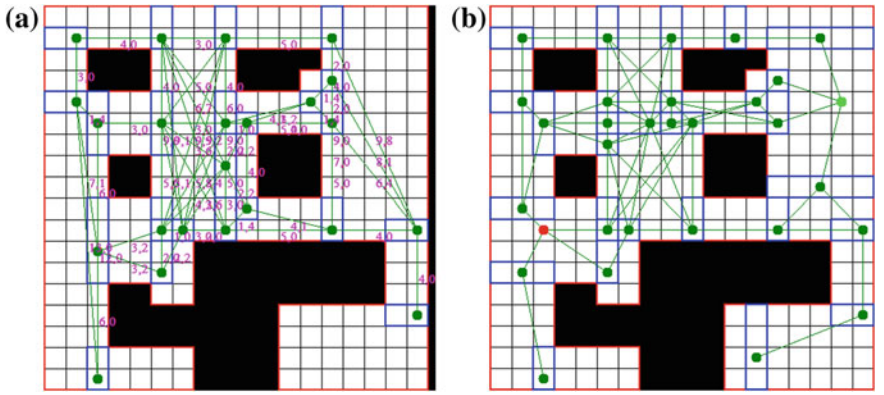

Fig. 7.6 Topological map: **a** weighted graph; **b** insertion of starting and goal nodes

7.4.1.2 Calculation of Boundary Zones

From the set of rectangular regions, border areas between them are determined. The algorithm considers pairs of rectangles to determine if there are groups of border cells between them. These cells will constitute a new set of rectangles one cell thick (blue in Fig. 7.5b). During the movement, a robot that has to come into an area has to go through one of these thin rectangles ('doors' or 'passages'), representing the input door-sill of the rectangles. If they are wider than robot size, they will become the nodes of a graph (Fig. 7.6a), while links will represent a chance to switch between different sills crossing the rectangle in which they are contained. As part of border regions of convex areas, links cannot pass over occupied cells, ensuring the clearance. The links are weighted by values representing the linear distance between the sills, according to the canonical Euclidean metric. This graph will be used for the Gross Motion Planning.

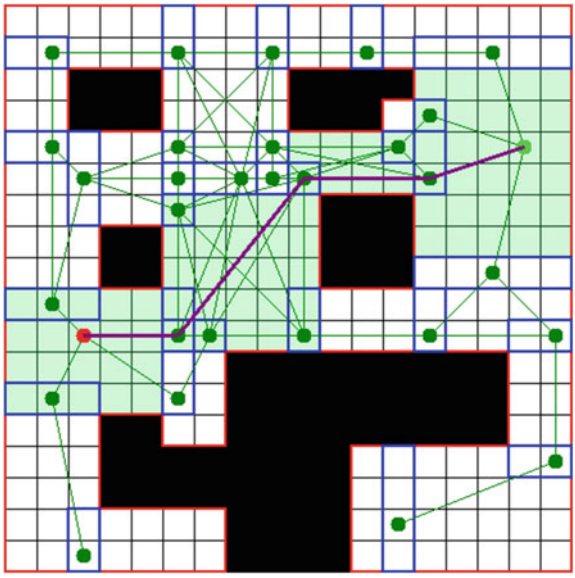

Fig. 7.7 Topological path ('channel' in light green)

7.4.1.3 Topological Path-Planning

Planning starts with the inclusion of two additional nodes that represent the starting point (green) and arrival point (red in Fig. 7.6b). These nodes are connected to all the existing door-sills of the rectangle, altering the topology (the algorithm cuts any other links inside the rectangle because it will never participate in the final trajectory. Likewise for the group of nodes connected to the arrival rectangle).

Topological level planning is done using the classic Djikstra's algorithm on weighted graphs (with exclusively positive weights), identifying the lowest cost path. This cost is only a rough estimate of the true length of the trajectory, and the path obtained is not optimal from many points of view. The purpose of this phase is just to identify the rooms involved by the trajectory in order to find a passing channel (Fig. 7.7), not to compute the path to be followed. With subsequent computation of a precise movement, it can determine the true trajectory inside the channel. Since the fine planner computation is relatively heavier, if we restrict its application solely to space involved by the channel, we can significantly reduce the computation times in some cases even by a factor of 10.

7.4.2 Examples

In this section, we show some examples related to the topological phase (or gross motion planning).

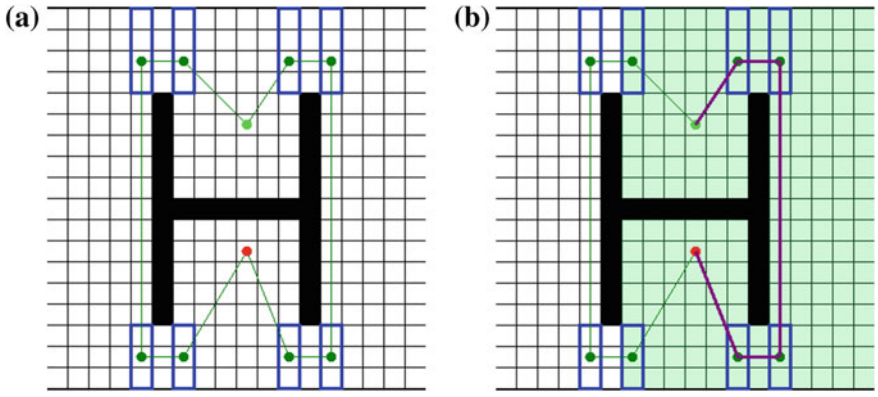

Fig. 7.8 H-shaped obstacle problem: **a** graph; **b** 'channel'

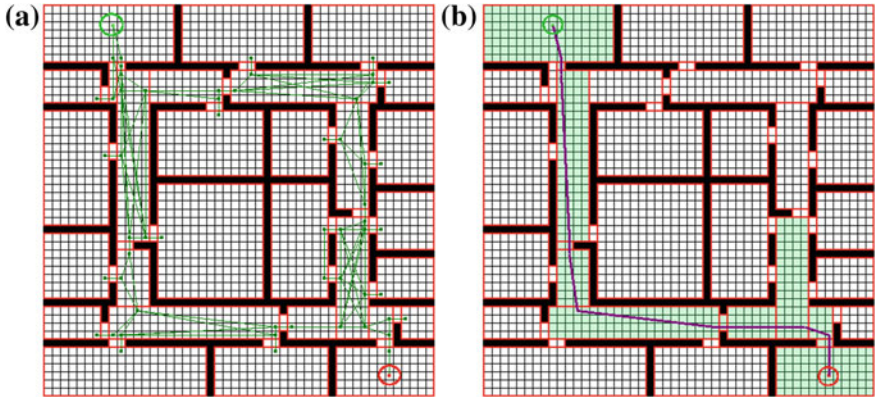

Fig. 7.9 Office-like environment: **a** graph; **b** 'channel'

7.4.2.1 H-Shaped Obstacle

Figure 7.8 shows that using nodes at the 'doors' of the rectangular regions, since they are convex, the planner does not stall in concave obstacles, but it always drives the robot to exit easily from a potentially critical situation.

7.4.2.2 Office-Like Environment

In the situation of Fig. 7.9, an office-like environment is represented. The total number of cells is 2,500 (50 × 50 cells). During the fine motion planning, the total number of cells of the Attractive Spacetime rises to 18 Mcells (18×10^6 cells). With the topological planning phase, the number of cells is reduced to about 3.5 Mcells, i.e. less than 20 %.

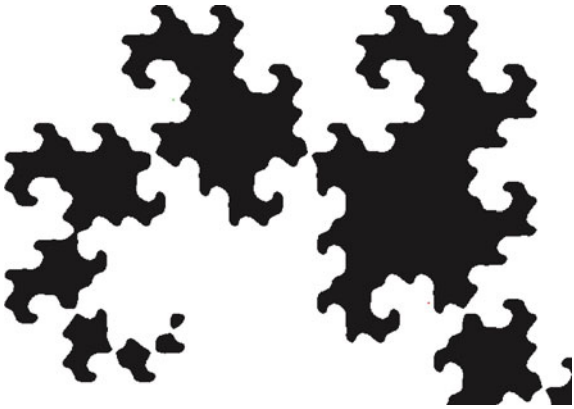

Fig. 7.10 Fractal obstacles example

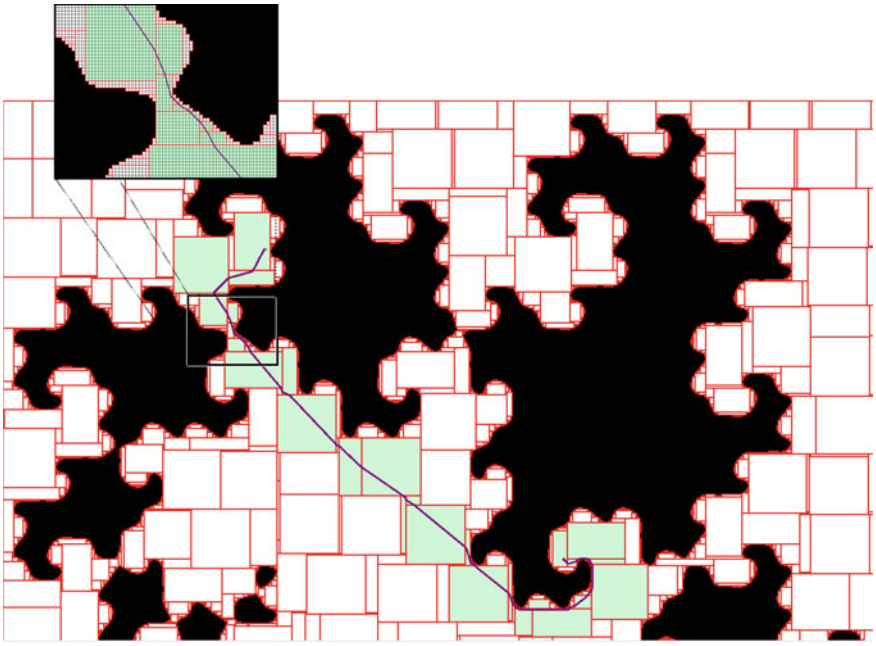

Fig. 7.11 Fractal obstacles channel

7.4.2.3 Fractal Obstacles

The last two examples were set to highly stress the system. In Fig. 7.10 a binarized fractal is represented on a grid of 650 × 450 cells (292,500 cells). In this situation, the Attractive Spacetime for a single robot would be more than 2.1 billions cells!.

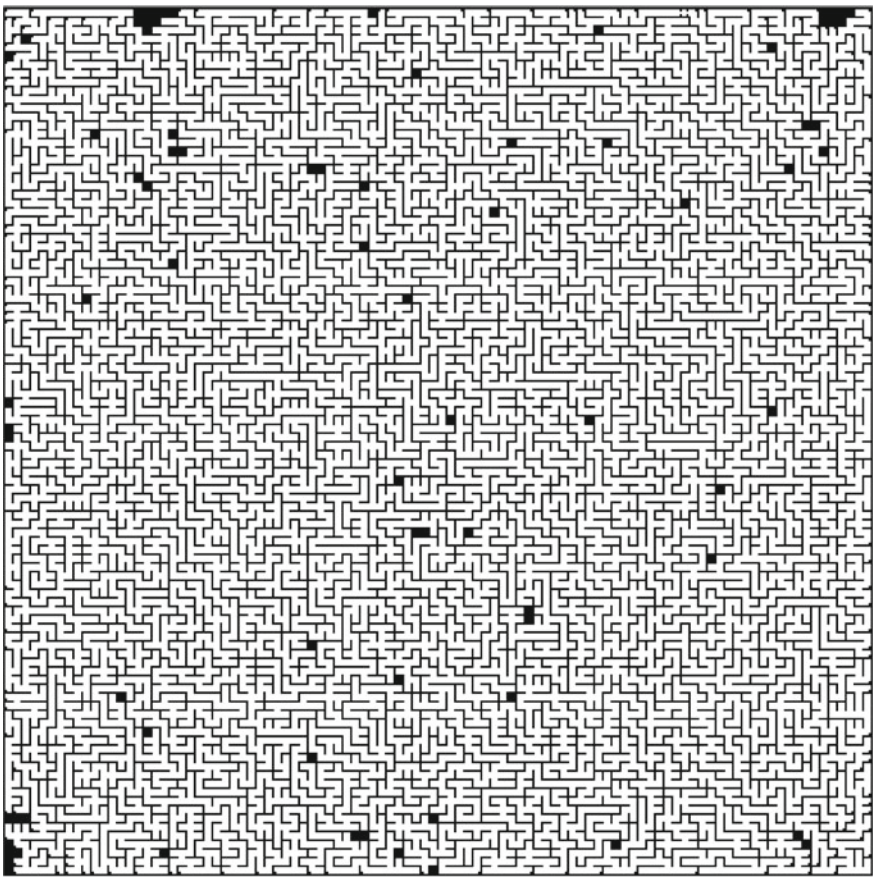

Fig. 7.12 The Maze

Making a topological planning and limiting the rectangles size to 10 % of the length of the environment sides (45 cells), we will get a total of 1,967 rectangles (Fig. 7.11). The trajectory obtained involves only 19,371 cells out of 292,500 (6.6 %), reducing the total amount of cells involved in the next phase up to 139.5 Mcells. Some computation times: even in such a hard situation, the most costly phase (graph building) requires only 0.15 s (on a Intel 32bit 8core 3.6 GHz), while the planning phase just 0.12 s.

7.4.2.4 Huge Maze

This example represents the most complex situation, although it is not realistic for a robot. A maze of 400 × 400 cells (160,000 cells) has been built (Fig. 7.12) using an automatic algorithm, making sure to generate at least 3 cell wide passages. It is

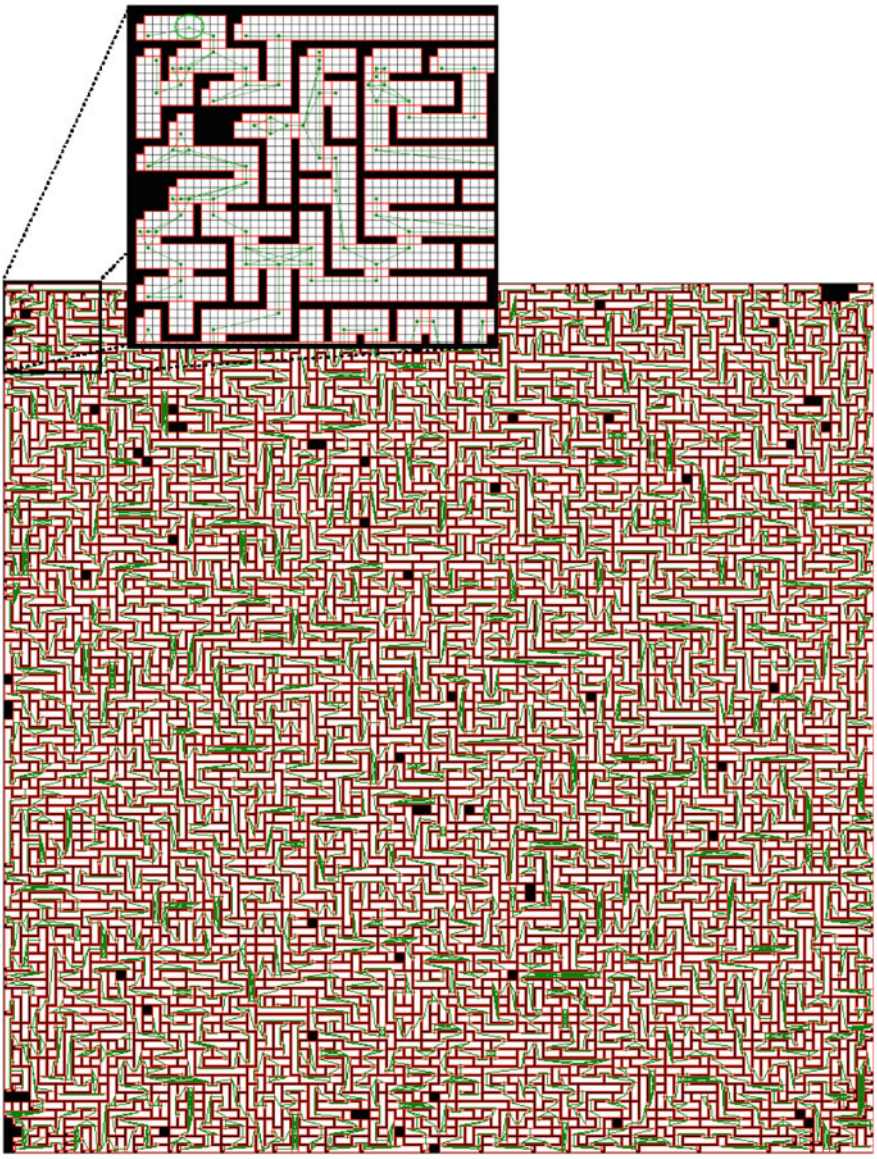

Fig. 7.13 Maze rectangular tessellation

unrealistic for a robot because it would be wide at most one cell to ensure the passage, but it represents a very hard test for this algorithm.

In Fig. 7.13 the set of rectangular regions is shown. The average computation time (on 1,000 repetitions) for this phase is 0.87 s.

Two problems have been tested: in the first the robot starts from the top-left corner and has to reach the bottom-right corner. In the second test a path covering a wider

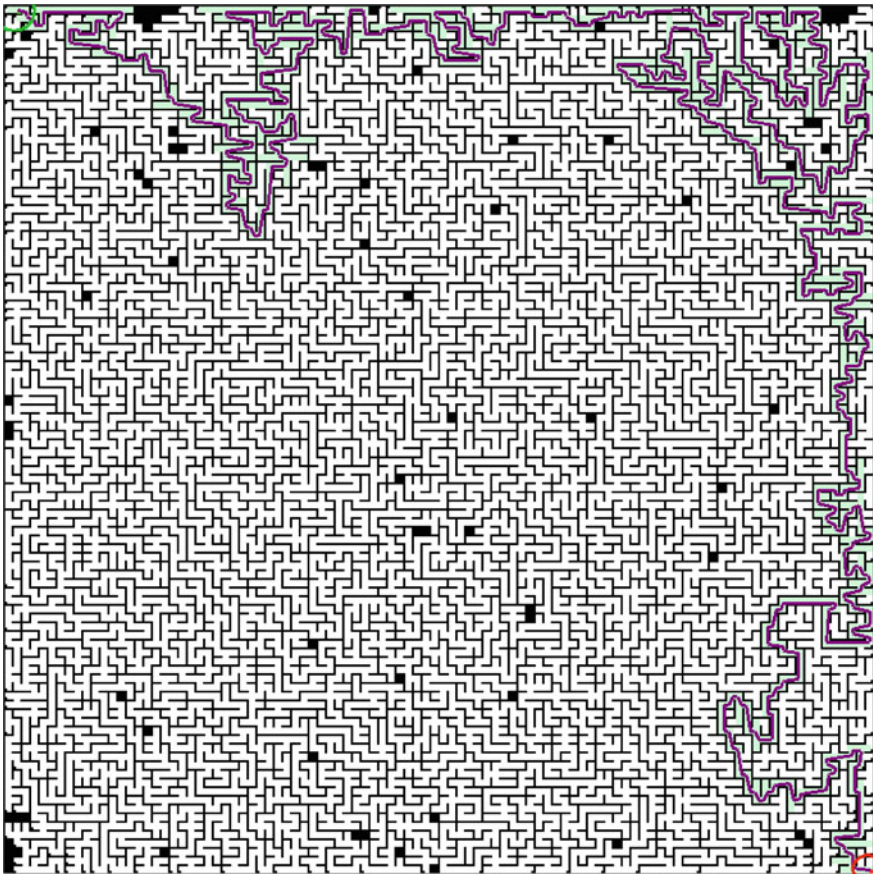

Fig. 7.14 Maze: first problem solution

area is found. The first result is shown in Fig. 7.14. The average time to solve the problem is 0.29 s.

The second problem is shown in Fig. 7.15 and requires 0.39 s. If we use the fine motion planner on this maze, we would have a total number of cells in the Attractive Spacetime of 4.6 Gcells for a single robot. With the topological phase to reduce the involved space, we would need only respectively 7 % of cells for the first issue and 25 % for the second (which deliberately has a broader coverage).

7.5 Multi-Robots Motion Problem

In a situation with multi-robots, the gross motion planning allows to decouple the problem moving a single robot at a time. The adjacency graph (relatively heavier at the computational level) remains unchanged and is calculated off-line once for all the robots: from time to time the goal and start nodes of a robot are added to calculate its

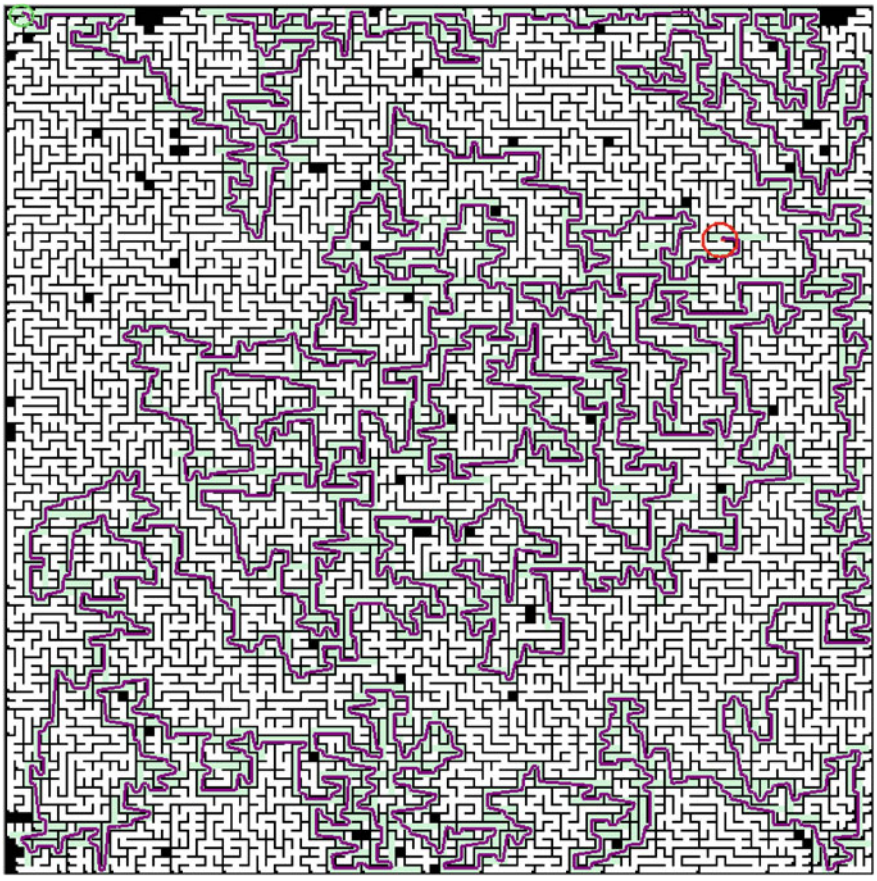

Fig. 7.15 Maze: second problem solution (path with a wider coverage)

trajectory (or to determine that it does not exist). This decoupling enables to locate the coarse paths for all the robots, but also has a second important utility: to determine the areas of interaction (interference) between robots. Identifying which robots pass through the same room, it figures out which robots risk mutual collision and which are excluded. It realizes a decoupling of the problem of multiple robots: on one hand there are robots that move through the environment without any risk of collisions and require just a planner for a single robot (polynomial complexity) applied separately (in parallel and non-interacting). On the other hand, there will be groups of robots that risk reciprocal collisions and that will be processed by a coordinated motion planner for MRS (coordinated within each group, but in parallel between different groups). In all cases, the level of complexity significantly decreases on the basis of the number of robots in addition to the space dimensions. If, as in the example of Fig. 7.16, instead of using a coordinated planner over four robots on the entire space, we use one on only two robots and a single planner on the remaining two, besides

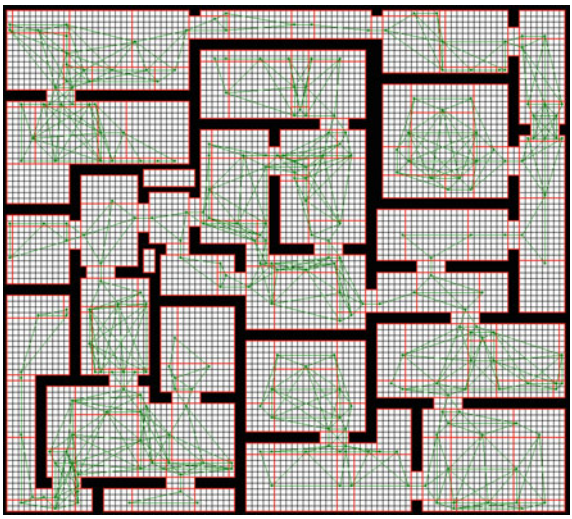

Fig. 7.16 Multi-robots motion problem graph

Table 7.1 Multi-robots motion problem performances (topological phase)

Phase	Mean time (ms)
Graph build	0.639
Green R. planning	0.046
Red R. planning	0.045
Blue R. planning	0.031
Yellow R. planning	0.030

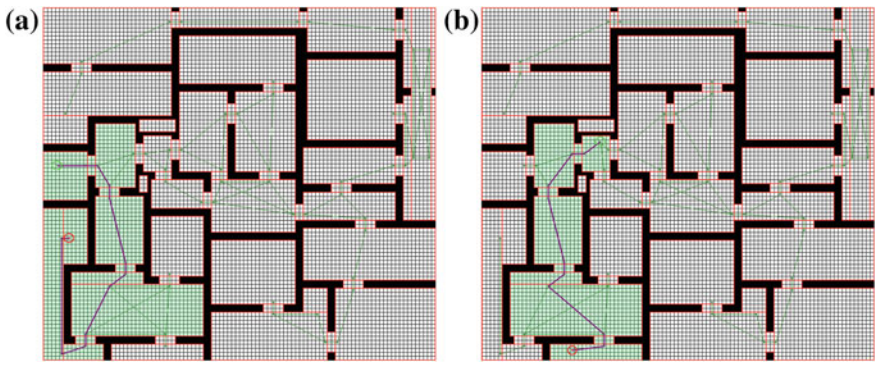

Fig. 7.17 *Green robot* (**a**) and *Red robot* (**b**) topological paths

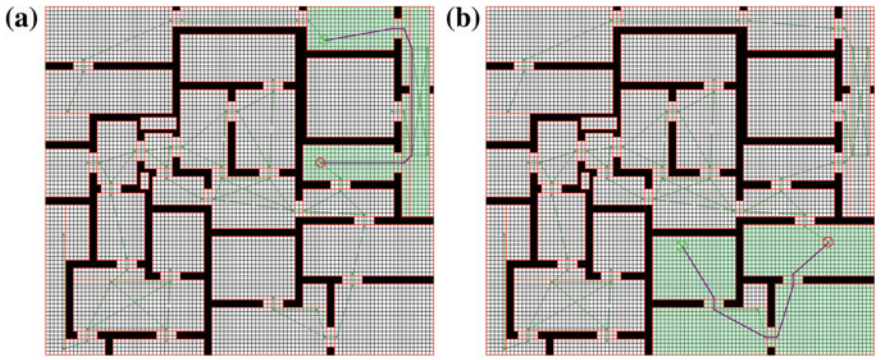

Fig. 7.18 *Blue robot* (**a**) and *Yellow robot* (**b**) topological paths

Table 7.2 Multi-robots motion problem performances (Spatiotemporal planning phase)

Phase	Mean time (s)
Green+Red R. planning	0.83
Blue R. planning	0.50
Yellow R. planning	0.84

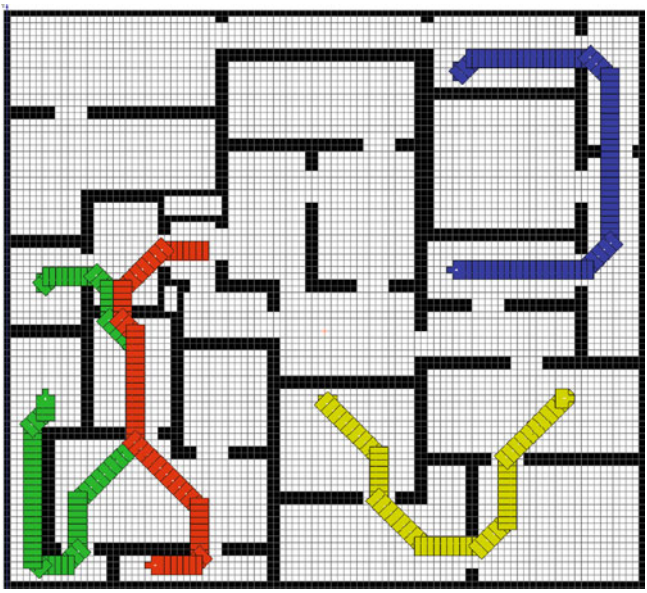

Fig. 7.19 Summary of the precise trajectories of the *Green*, *Red*, *Blue* and *Yellow robots*

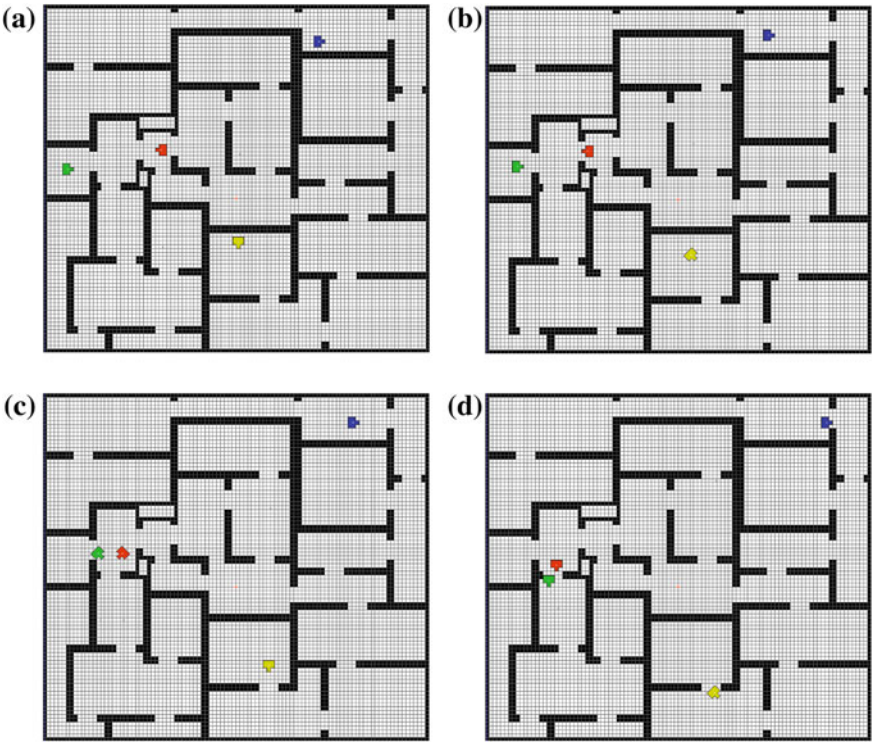

Fig. 7.20 MRS movement (1): **a** T = 0 (robots start moving); **b** T = 5; **c** T = 12 (the interaction between *Red* and *Green robots* starts); **d** T = 20 (the *Green robot* has higher priority and passes first)

reducing the space involved in the planning, the complexity is much lower. For this problem, the average calculation time is less than one millisecond for all phases, including the construction of the graph, as shown in Table 7.1 (Figs. 7.17, 7.18).

The calculation of the graph is made once for any problem data set with any number of robots and can be made off-line. The calculation for each robot takes place in real-time in parallel on different processors, and the worst result is for the Green robot with 0.046 ms, a time absolutely negligible. In the next step, we use the fine motion planner to determine the precise trajectories, maneuvering to handle the interaction between robots in the same rooms. The result is shown in Fig. 7.19, where the trajectories have been summarized. As we can see, the spatiotemporal motion planner finds slightly different but more efficient trajectories, handling the real robots shapes and their orientations; in particular, it manages the interaction between Red and Green robots to avoid collisions.

In Figs. 7.20, 7.21, snapshots at significant timestamps are reported. It is interesting to note the interaction phase between Red and Green robots starting at $T = 12$ to $T = 45$. Actually, the calculation has been assigned to three instances of the same

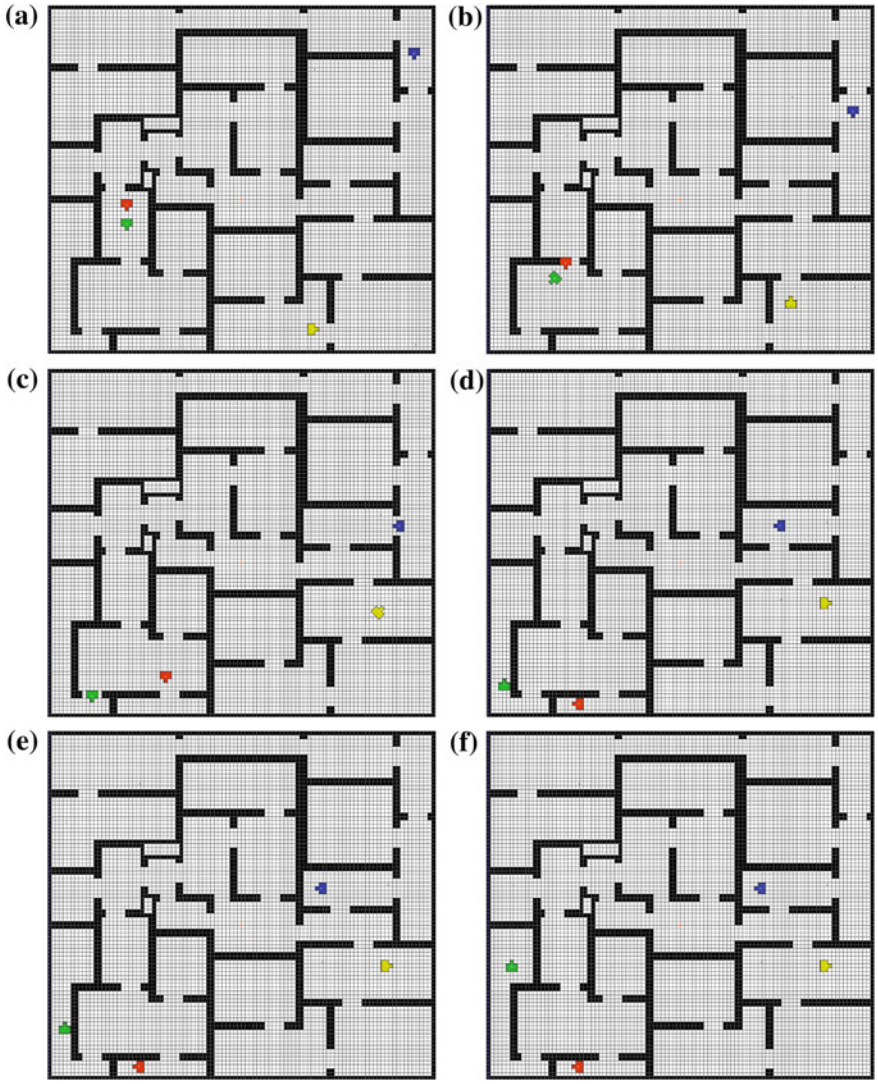

Fig. 7.21 MRS movement (2): **a** T = 30; **b** T = 45 (*Red-Green* interaction ends); **c** T = 60; **d** T = 75 (*Yellow robot* stops) **e** T = 80 (*Red* and *Blue robots* stop); **f** T = 98 (*Green robot* stops)

spatiotemporal motion planner and runs on three different microprocessors. Out of the three instances, the worst computing time results on the Yellow robot (Table 7.2), and it is the time that affects the entire procedure (0.84 s for a total of 15.1 Mcells). Therefore, the total time for all the phases is 0.85 s. On the other hand, assigning the test to only one single instance of the same coordinated motion planner handling all the four robots, the computation time required would be 4.31 s on a total of 64 Mcells

managed. Considering the planning time during the motion (real-time planning), a coordinated motion planner is 5 times slower than the solution proposed (4.31 s vs. 0.85 s).

7.6 Conclusions

A two-phase planner has been used based on a joint metric-topological representation that allows to split the general problem of planning into two separate planning issues (*divide et impera*). Clustering of cells in rectangular homogeneous areas actually introduces a multi-resolution approach: we use a more abstract topological map at a lower resolution and, in the second phase, a regular cells decomposition map at high resolution. The tessellation with variable-sized rectangles allows a complete coverage of the free space and maintains a spatial link with the highest-resolution map. This allows us to move between the two different resolution levels on a spatial basis.

The approach is characterized therefore by:

- Multi-resolution: the detection of clusters of cells leads to two resolution levels of representation, to which a third topological level is added;
- Hierarchical Planning: gross and fine planning brought on two different resolution levels, the gross one for channels identification and the fine one for the precise movements.

The gross part allows to identify channels and significantly reduce the area of interest (number of cells) operated by the spatiotemporal planner. The latter is able to determine the precise trajectories, spreading potentials only within channels. It is also able to solve the interaction problems (in the conflict areas) only between the robots involved, significantly reducing the total amount of computational time.

References

1. Barraquand, J., Langlois, B., Latombe, J.C.: Numerical potential field techniques for robot path planning. IEEE Trans. Syst. Man Cybernet. **22**(2), 224–241 (1992)
2. Behring, C., Bracho, M., Castro, M., Moreno, J.A.: An algorithm for robot path planning with cellular automata. In: Bandini, S., Worsch, T. (eds.) Theoretical and Practical Issues on Cellular Automata, Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry, Karlsruhe, D, Oct 4–6, 2000, pp. 11–19. Springer, Berlin (2000)
3. Bennewitz, M., Burgard, W., Thrun, S.: Optimizing schedules for prioritized path planning of multi-robot systems. In: ICRA, pp. 271–276. IEEE Computer Society (2001)
4. Brooks, R.A.: Solving the find-path problem by good representation of free space. In: Waltz, D.L. (ed.) Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, Aug 18–20, 1982, pp. 381–386. AAAI Press (1982)
5. Canny, J.F.: The Complexity of Robot Motion Planning. MIT Press, Cambridge (1988)

6. Culberson, J.C.: Sokoban is pspace-complete. In: Proceedings of the International Conference on Fun with Algorithms (FUN98), Ontario, Canada. Carleton-Scientific (1998)
7. Ioannidis, K., Sirakoulis, G.C., Andreadis, I.: A cellular automaton collision-free path planner suitable for cooperative robots. In: Panhellenic Conference on Informatics, PCI 2008, August 28–30, 2008, Samos Island, Greece, pp. 256–260. IEEE Computer Society (2008)
8. Jahanbin, M.R., Fallside, F.: Path planning using a wave simulation technique in the configuration space. In: Gero, J.S. (ed.) Artificial Intelligence in Engineering: Robotics and Processes. Computational Mechanics Publications, Southampton (1988)
9. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
10. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge (2006)
11. LaValle, S.M., Hutchinson, S.: Optimal motion planning for multiple robots having independent goals. IEEE Trans. Robot. Automat. **14**(6), 912–925 (1998)
12. Lozano-Pérez, T.: Spatial planning: A configuration space approach. IEEE Trans. Comput. **C-32**(2), 108–120 (1983)
13. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
14. Marchese, F.M.: Cellular automata in robot path planning. In: Proceedings of the First Euromicro Workshop on Advanced Mobile Robot, Kaiserslautern, D, pp. 116–125 (1996)
15. Marchese, F.M.: A path-planner for generic-shaped non-holonomic mobile robots. In: Proceedings of European Conference on Mobile Robots (ECMR 2003), Radziejowice, Poland (2003)
16. Marchese, F.M.: Multiple mobile robots path-planning with mca. In: International Conference on Autonomic and Autonomous Systems (ICAS 2006), California, pp. 56–61. IEEE Computer Society (2006)
17. Marchese, F.M.: Time-invariant motion planner in discretized c-spacetime for mrs. In: Yasuda, T. (ed.) Multi-Robot Systems. Trends and Development, pp. 307–324. InTech, Vienna, A, A (2011)
18. Parker, L.E.: Cooperative motion control for multi-target observation. In: Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97, Sept 7–11, 1997, Grenoble, France, pp. 1591–1597 (1997)
19. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Proceedings of the 20th Annual Symposium on Foundations of Computer Science, SFCS '79, pp. 421–427 (1979)
20. Tzionas, P.G., Thanailakis, A., Tsalides, P.G.: Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. IEEE Trans. Robot. Automat. **13**(2), 237–250 (1997)
21. Warren, C.: Multiple robot path coordination using artificial potential fields. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 500–505 (1990)
22. Zelinsky, A.: Using path transforms to guide the search for findpath in 2d. Int. J. Robot. Res. **13**(4), 315–325 (1994)