Motion Planning Method Using Cell Decomposition With Petri Nets

Dr. Hala Bahjat	Dr.Yossra Hussain Ali	Dr.Alia Karim Abdul Hassan
University of Technology	University of Technology	University of Technology
Bagdad/Iraq	Bagdad/Iraq	Bagdad/Iraq
Hala_bahjat@yahoo.com	Yossra_h_a@yahoo.com	hassanalia2000@yahoo.com

Abstract: In this work a proposed method for robot motion planning in two dimension static environments is presented. This method use cell decomposition principles and Petri Nets as unified model for all the motion planning subtasks. This method is called Petri Net based Motion Planning Method (PN-MPM) for single and multiple mobile robots and has the ability to find a collision free path.

PN-MPM is a complete and sound method. Complete since it guarantees to yield a collision free path if one exits. And sound since it guarantees that all its solutions are collision free. PN-MPM can plan a path for the robot in the workspace cluttered with rectangle shapes obstacles or different shape obstacles, as well as generates collision free path since the path has no contact with obstacles.

الخلاصة

في هذا البحث تم وضع طريقة لتخطيط الحركة للروبوت في محيط ثنائي الأبعاد ثابت. هذه الطريقة قائمة على استخدام تجزئة الخلايا مع شبكات بتري كنموذج موحد لكل مهام تخطيط الحركة. هذه الطريقة سميت طريقة شبكات بتري لتخطيط الحركة لروبوت واحد او اكثر. وتتمكن من أيجاد مسار خالي من التصادم.

طريقة شبكات بتري لتخطيط الحركة هي طريقة كاملة وقوية. كاملة لأنها تضمن ان تنتج مسار خالي من التصادم إن وجد. وقوية لأنها تضمن أن تكون كل الحلول التي تنتجها خالية من التصادم. طريقة شبكات بتري لتخطيط الحركة تستطيع تخطيط الحركة لروبوت في محيط تنتشر فيه العوائق المستطيلة الشكل او عوائق مختلفة الأشكال. وتولد طرق خالية من التصادم لأن ليس لها تماس مع أي عائق.

1. Introduction

The motion-planning problem is the problem of finding a collision-free motion for one or more moving objects between the start and the goal configurations. A moving object can be a rigid object, or a jointed object such as an industrial manipulator [1]. To be able to plan a motion, a robot must have some knowledge about the environment in which it is moving. For example, a mobile robot moving around factory must know where obstacles are located. Some of this information where walls and machines are located can be provided by a floor plan. For other information the robot will have to rely on its sensors. Using information about environment, the robot has to move to its goal position without colliding with any of the obstacles [2]. Motion planning may decompose into three subtasks are [3]:

Subtask1: mapping and modeling the environment; this concerns the representation of free space; the workspace through which a robot is to move amongst a number of obstacles.

Subtask2**:** path planning; this constitutes the core of the planner, it concerns the computation (i.e. searching) within predetermined criteria, of near optimal or even an optimal paths for a robot to navigate throughout its environment.

Subtask3**:** path following and collision avoidance; path following in the case of single robot motion and for multiple robot motion path following and coordination.

The problem to be addressed here is an instance of the famous finding path or mover's in robotics. Everything is to be assumed static except the position of the navigator who is small compared to the area being traversed.

Most of the researches are focused on the workspace representation since "finding collision free path" is more important than finding optimal path or optimizing the path generation. Little efforts was dedicated to search strategy, thus A*, heuristic, breadth-first, and other

strategies were found to be conveniently adopted. Javier, Dario, & Jose, in 1998 proposed a method for single mobile robot route planning. In this method topological modeling of environments is used and route-planning algorithm for single mobile robot based on Fuzzy Petri Net has been presented [4]. In 1989 Lui, etal, presented a method that uses a decoupled approach to avoid collisions in multi robot system. Then it used approximate cell decomposition approach to find collision free path if one exists for each robot and uses PN to resolve conflicts among two robots [5].

The expressive and representation power of Petri Nets (PN) renders them ideally suited to the modeling of many complex event systems and to reveal important information about the structure and dynamic behavior of the modeled system [6]. In this work Petri Nets are used as a modeling tool for the free space of the environment (subtask1) and use the reachability analysis method of PN for the searching collision free path with predetermined performance (subtask2), while the notion of time of PN is used for coordination of the path following for multiple robot motion paths planning (subtask3).

2. Motion Planning

The process of generating a sequence of actions that has to be performed in order for a robot to move through its environment autonomously and without collisions is called motion planning [7].

The space in which the motion problem "lives" will be defined as the workspace (world), for which there are two possible choices [8]: A two-dimensional (2D) workspace in which $W = R^2$ (R denotes the set of real numbers). And three-dimensional (3D) workspace, in which $W = R^3$. Generally the workspace contains two entities [8]:

1- Obstacles: Portions of the workspace that are "permanently" occupied, for example, as in the walls of a building.
2- Robots: Geometric bodies that behave according to a motion strategy.

Basic motion planning problem (*single robot motion planning problem*) assumes that the robot is the only moving object in the workspace around stationary obstacles. This problem can be solved by merely constructing a geometric path.

An alternative way of classifying motion-planning methods is to say whether they are *on-line* or *off –line.* On-line planning is performed in real time, i.e., at the same time the robot is moving, and is exceptionally useful when the environment is not known. Off-line planning is performed before any robot motion and is not useful unless the workspace is known. Almost all motion planning methods can be characterized along the following [9]:

***Complete*:** A method is said to be complete if it guaranteed to find a collision-free path if one exists; otherwise return failure.
***Sound*:** if it guarantees that all its solutions are correct (i.e., collision free).

In the case where several robots move independently in the same workspace among stationary obstacles the resulting problem is called the *multiple robot path-planning problems* [9]. In order to organize the various facets of motion planning in a coherent framework, the basic concepts to motion planning will be exposed in detail.

2.1 Configuration Space

In [10], the author suggest that instead of handling a complex geometrical representation of a robot in the Euclidean representation of the workspace, the robot could be treated as a point in its Configuration space (C-space). The underlying concept is to represent the real-world robot as a point in an appropriate space, and to map obstacles into this same space. Then, the space

contains a concise representation of the robot's geometrical constraints on motion, and a motion planner needs only to consider the path of the single point, which represents the robot [9]. Figure (1) shows the configuration q of a rotation invariant robot.

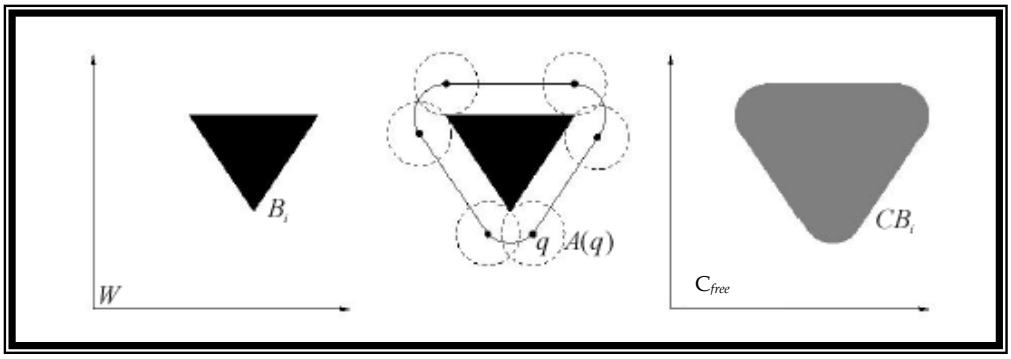

Figure (1) Configuration space for rotation invariant robot

A Specifies the exact position and orientation of A relative to a fixed reference frame. Therefore, the C-space of A is the set of all possible configurations of A. Obstacles are mapped into C-space by determining which configurations of the robot produce collisions with an obstacle; these configurations are deemed forbidden. Let $A(q)$ denote the location of $A's$ particles when A is in configuration q. A C-space obstacle (or "C-obstacle") associated with a physical obstacle B is defined as

$$CB = \{q \in C \mid A(q) \cap B = \phi\} \text{---------(1)}$$

The complement of the C-obstacles is termed the "free space":

$$C_{free} = C \setminus CB \text{---------(2)}$$

Motion plans are constructed in C_{free}.

2.2 Methods for Motion Planning

There exist a large number of methods for solving motion planning problem for single and multiple mobile robots. Despite many external deference's, the methods are based on few different general approaches. These approaches will be described in the next subsections.

2.2.1 Single Robot Motion Planning

To date, motion planning approaches for single robot can be classified into three categories [9]: Skeleton (Roadmaps), Cell decomposition and Potential field.

In the **skeleton** approach, the free space is represented by a network of one-dimensional (1-D) paths called a Roadmap. There are many different roadmap methods, but one thing they all have in common is that they try to convert the free space of the workspace into a graph representation (a roadmap). A collision-free path can now be constructed (if one exists) by connecting the start position and destination to the roadmap.

In the **cell-decomposition** methods the free space of the workspace is decomposed into a set of cells. The cells must be simple so that a path easily can be planned through each cell (a suitable cell is typically a convex polygon). A *channel* of free cells (i.e., a sequence of contiguous cells) is subsequently constructed starting with the cell, which contains the current position of the robot, and ending with the cell that contains its destination. Finally, a path can

be planned from the start position through the channel to the destination. Cell decomposition is further divided *into exact and approximate cell decompositions*.

In the **potential-field** approach, a scalar potential function that has high values near obstacles and the global minimum at the goal is constructed. In this case the robot moves in the direction of the negative gradient of the potential.

Most methods for motion planning can be derived from these approaches or hybrids of these approaches.

The motion planning approach called cell decomposition method is quite attractive. It allows generation of collision-free. Moreover it is practical and it takes global knowledge into consideration. Cell decomposition methods have the following main steps [9]:

1) Represent the free space as collection of cells.
2) Generate the connectivity graph representing the adjacency relation between cells.
3) Search the connectivity graph for a sequence of adjacent cells connecting the initial to the goal cell.
4) transform the sequence of cells (if one has been produced) into a path. Each cell represents a connected region of free space.

2.2.2 Multi Robot Motion Planning

All motion planning methods for single robot motion planning are applicable to multiple robot motion planning but with modification. According to the way the multiple robots are treated the multi-robot motion planning approaches are often categorized *as centralized* and *decoupled* [11].

Centralized approaches treat the separate robots as one composite system, and typically perform the planning in a composite configuration space, formed by combining the configuration spaces of the individual robots.

Decoupled approaches first generate paths for the separate robots more or less independently, and then consider the interactions between the robots (with respect to the generated paths). Decoupled is more less computation complexity than the centralized approaches [9].

3. Petri Nets Models

Carl Adam Petri in the beginning of the 1960s presented the foundation of PN. Petri wanted to define a general-purpose graphical and mathematical model describing relations between conditions and events. PNs offer a versatile modeling framework for complex distributed concurrent systems and have been used in a wide range of applications, such as computer systems, communication networks, real time systems, automated manufacturing systems consisting of computer-controlled machine and automated guided vehicles, and power systems [4].

PNs are mathematical descriptions of systems. They consist of four basic elements: transitions, places, arcs and tokens. Transitions and places are described by sets. Arcs connect transitions and places, and are described by backwards and forwards incidence functions, often termed arc weights, which relate to the movement of tokens between places. Tokens are associated with places and the initial marking describes the initial number of tokens on each place. Markings, the number of tokens on each place, represent subsequent system states. A marking can be represented by 1-dimensional integer vector whose components correspond to the places of the net, *tokens* exist into the circle representing the place, then it is said that

place *p* holds *K* tokens at the marking *M*. A special marking denoted by M_0, will be called *initial marking* [12].

Formal Definition of PN [9]:
An unmarked PN is a 4-tuple N={P, T, I, O} where:
· $P = \{P_1, P_2, \ldots, P_n\}$ *is a finite, nonempty set of places.*
· $T = \{t_1, t_2, \ldots, t_m\}$ *is a finite, nonempty set of transitions.*
· $P \cap T = \varnothing$, *i.e., the sets P and T are disjoint.*
· $I: P \times T \rightarrow \{0,1\}$ *is the input incidence function.*
· $O: P \times T \rightarrow \{0,1\}$ *is the output incidence function.*
A marked PN is a pair PN=(N,M_0) in which N is an unmarked PN and M_0 is the initial marking. $I(P_i, T_j)$ is the weight of the arc connecting place P_i with transition T_j .This weight is **1** if the arc exists and **0** if not. $O(T_j, P_i)$ is the weight of the arc connecting transition T_j with place P_i. This weight is **1** if the arc exists and **0** if not.

Transition, Enabling and Firing [9]:
M is a marking of PN and $t \in T$, then t is said to be *M-enabled* (denoted as *M[t>* if $\forall p \in I(t) \cap P: M(p) > 0$ i.e. , all the input places of transition t have at least one token). Transition t can be fired from *M* if t is *M-enabled*, firing t from *M* results in a new marking *M'* (denoted as *M[t>M'*), for $\forall p \in P$, we have

$$M'(p) = \begin{cases} M(p) - 1 & p \in I(t) \wedge p \notin O(t) \\ M(p) + 1 & p \in O(t) \wedge p \notin I(t) \\ M(p) & otherwise \end{cases} \text{--------(3)}$$

Firing Sequence [9]:
A sequence $\sigma = t_1, t_2, \ldots, t_n$ of transitions is a firing sequence for a given marking M_1 in a PN, if and only if there exists a sequence of markings $M_2, M_3, \ldots, M_{n+1}$ such that:

$$\forall_i \in \{1,2,\ldots,n\}, \quad M_i \xrightarrow{t_i} M_{i+1}, \quad where \quad M_i \in R(M_0) \text{--------(4)}$$

In general, it is denoted as $M_1 \xrightarrow{\sigma} M_{i+1}$

3.1 Properties of PNs
Properties of PNs that are useful in modeling and analysis are mainly divided into two groups as *modeling properties* and *behavioral properties.* Modeling properties reflect the features of systems (such as concurrency, conflict, nondeterministic, synchronization) that PNs model. Behavioral properties are directly related with the net model, and are marking-dependent (depend on the initial marking). Below are some of these [9]:

Boundedness: A place p_i is bounded for an initial marking M_0 if there is a nonnegative integer *k* such that, for all markings reachable from M_0, the number of tokens in p_i is not greater than *K* (p_i *is* said to be k-bounded).
Liveness: A PN is said to be *liv*e, if for any marking $M \in R(M_0)$, it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. A live PN therefore guarantees *deadlock-free* operation.
Safe: A PN is safe for an initial marking M_0 if, for all reachable markings, each place contains at most one token, i.e., the net is 1-bounded.

3.2 Reachability Tree [14]

It involves essentially the enumeration of all reachable marking such that the reachability tree of the PN model under a designated initial marking is obtained. In this tree the nodes correspond to the reachable markings and the arcs correspond to the firing of transitions. Another, equivalent name for the reachability tree is the *marking tree.* The algorithm takes the initial marking M_0 as input and produces the reachability tree as output. This algorithm is given below:

Reachability Tree Construction

Input: Initial marking M_0
Output: Reachability tree
Step1: Let the initial marking be the root node and tag it "new" and M as current marking.
Step2: While new markings exist Do

2.1 If M is identical to another node in the tree which is not "new", then tag M to be "old" and stop processing M .
2.2 If no transition is enabled in M then tag M to be "terminal"
2.3 For every transition t is enabled in M Do
2.3.1 obtain the marking M', which results from firing t in M.
2.3.2 introduce M' as a node, draw an arc from M to M' labeled t, and tag it "new".

3.3 Timed PN [9]

Using timed PN (TPN) the system with time dependent behavior can be described. TPN introduces the concept of time. Time can either be introduced into a PN by specifying sojourn times of tokens on places or associating firing delays with enabled transitions. For the former, tokens generated on places are unavailable for a set time. In the latter case, enabled transitions wait a firing delay before firing. Using TPN, systems with time dependent behavior can be described.

4. The Proposed Motion Planning Method (PN-MPM)

In this section, the proposed method for motion-planning problem is considered. PN is used as unified models for modeling the workspace, searching for shortest collision free path and path coordination. The below subsection show the used of PN to develop the method.

4.1 PN Definition

A definition of PN used in motion planning method will be presented. First it is assumed that the PN is safe, namely, the number of tokens in one place is guaranteed to be either zero (free position) or one (single robot). B is used to denote the set $\{0,1\}$, R the set of real number. Let P represents a set of places, T set of all transitions. The marking M is denoted by M_n where $M_n : P \to B$, is the number of tokens in a place. Mt is defined as $M_t : P \to R$, is the time when the token is put at the place while $M_d : P \to R$ is the time when the token becomes available in the place. $\boldsymbol{M}$ denotes the set of all markings. The input incidence function is $I : P \times T \to B$ and the output incidence function is $O : T \times P \to B$. Then the TPN is defined as PN= (PN, M_0), where PN= (P, T, I, O), and $M_0 \in \boldsymbol{M}$ is an initial marking. Let $M \in \boldsymbol{M}$ and $t \in T$. A transition t is said to be enabled at time τ if

$$M_n(p)\phi(\tau - M_t - M_d) \geq I(t) \qquad (\forall p \in P) \text{ --------(5)}$$

$$\text{Where } \phi(.) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

4.2 Free Space Modeling

In this work, for simplicity reasons monochrome bitmaps are used in which obstacles are denoted by black color with associated value 0 and the free space is represented by the white color with associated value 1. In this way, one pixel occupies one bit in the computer memory. The reason for choosing raster image as data structure is related to the fact that most sensors (image sensors and ultrasonic and laser range finding sensors) yield such data.

For the navigation system, the set of places P is formed by positions and orientations of the robot and the set of transitions T. Each transition determines the movement of the robot between two places. The net marking depends on the place where the robot is situated. As mentioned in the previous above a two-dimension matrix *(row×col)* represents the workspace, where each number in this matrix reflects its position in the image. In this work each free position (pixel with white color) is represented as single place and each legal movement from this place to any eight free neighbor places represents a single transition.

4.3 Example1: Free Space Modeling

The free space modeling by the PN and HPN can be is expressed by the following example. The workspace of Figure (2) represented by (10×10) matrix with (S) as a start position and (G) as the goal position as displayed in Figure (2).

1) Simple places will be referred to with small p and places with capital P, and for transitions with $p_i \rightarrow p_j$ where p_i is the input place and p_j the output place.
2) Since the workspace is so small, modeling is for the workspace not for C-space. But real implementation is for C-space.

1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
S	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99
10	20	30	40	50	60	70	80	90	G

Figure (2) the workspace is represented with (10×10) matrix with (S) as a start position and (G) as the goal position.

PN construction: Each position has a number, which represents the corresponding place, which models this position. The set of places and set of transitions are:

$P = \{p1, p2, p3, \ldots, p100\}$

$T = TLR \cup TUD \cup TD1 \cup TD2$

$TLR = \{p1 \rightarrow p11, p11 \rightarrow p1, \ldots, p90 \rightarrow p100, p100 \rightarrow p90\}$

$TUD = \{p1 \rightarrow p2, p2 \rightarrow p1, \ldots, p99 \rightarrow p100, p100 \rightarrow p99\}$

$TD1 = \{p1 \rightarrow p12, p12 \rightarrow p1, \ldots, p89 \rightarrow p100, p100 \rightarrow p89\}$

$TD2 = \{p2 \rightarrow p11, p12 \rightarrow p3, \ldots p89 \rightarrow p98, p89 \rightarrow p80\}$

Now free space is C_{free}=PN

4.4 Searches for Collision Free Shortest Path

In this section, a method for a collision free shortest path for an object moving between two points will be described. This method finds a collision free path through PN which is used as a process model for finding a collision free path. The analysis of their corresponding

reachability tree has turned out to be a suitable search technique. A reachability tree path is derived as a sequence of selected legal movement set. Traditionally the analysis of the reachability tree is a two phased process. The first step consists of the construction of the complete reachability while in the second step the tree is analyzed. Existing approach exploring of reachability tree for searching a collision free path might be divided into heuristic and optimizing approach. The executed reachability (*Pathfinder*) in this work is a modified version of the normal reachability described in section 3. It works *heuristically* under the control of the distance function:

$$f(M)=h(M)+g(M) \text{ ---------(6)}$$

Here, *f(M)* is an estimate of the cost, i.e., the make span from initial marking to final marking along optimal path which goes through the marking *M*. g*(M)* is the current cost obtained from initial marking to the current marking *M*. *h(M)* is an estimate of the cost from current marking to the final marking along the optimal path which goes through the marking *M*. Under *optimization* conditions it skips any path which has a back step and skip any path its cost exceeds the current shortest path cost that, or if the current place is reached from other place with g*(M)* smaller than the current. *Pathfinder* is described in algorithm (1). The input to this algorithm is the initial marking of the HPN model of the workspace M_0 and the final marking *Mg*. The algorithm output is a collision free path.

Algorithm (1): Pathfinder

Input: initial marking M_0, and goal marking Mg
Output: collision free shortest path
Step1: Initialize
- *1.1 Current marking $M=M_0$*
- *1.2 Let Open= [] a list contains all the marking to be processed*
- *1.3 Let shp= [] the shortest path cost, path is the shortest path, and cst is the path cost*
- *1.4 Tails =X the number of trails to find shortest path*
- *1.5 M_{dead}= [] a list containing markings in which there is no enabled transition in it*
- *1.6 $M_{backstep}$= [] a list which contains markings which appeared previously in the tree*
- *1.7 σ = [] is the firing sequence of transitions*

Step2: Include M in Open list
Step3: Test Open list for
- *3.1 If Open is empty and shp=0, terminates with failure*
- *3.2 If Open is empty, return the current generated path as the shortest path*

Step4: For each marking in Open Do
- *4.1 Remove the first marking M from Open*
- *4.2 If M is the final marking Mg, then*
  - *4.2.1 Path Generator with σ firing sequence of transitions, which lead to the goal marking*
  - *4.2.2 Put the generated path length in len*
  - *4.2.3 If cst < shp then shp =cst and path =generated path*
  - *4.2.4 If this is the last trail then terminate return shp as shortest path else go to 3*
- *4.3 Find the enabled transitions of the marking M*
- *4.4 Generate the next marking, or successor, for each enabled transition, and set pointers from the next marking to M. Compute g(M') for every successor M'*
- *4.5 For every successor M' of M do the following*
  - *4.5.1 If M' is the parent of M then calls it $M_{backstep}$*
  - *4.5.2 If the distance from M0 to M' is greater than shp then calls it M_{dead}*

4.5.3 If M' is already in Open but g(M') smaller than the one existing in the Open list, then remove the one in the Open list and call it M_{dead}. Put M' in Open as M_{new} and compute $f(M_{new})$

4.5.4If M' is not one of that above conditions, then call it M_{new}, include it in the Open list and compute $f(M_{new})$

Step5. Reorder Open in increasing order of f

Step6. Go to step3

Step7. Return (path)

4.5 Example2: Search Collision Free Path

The modeled workspace in example-1 is used for search. For a collision free path between the start place and goal place (*p5, p100*). *Pathfinder* generates the path as shown in Figure (3).

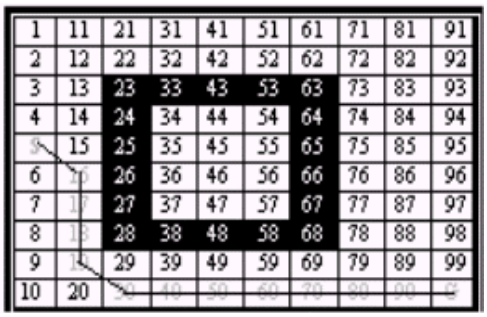

Figure (3) shortest path pathfinder algorithm

5. The Proposed Method for Single Robot

The proposed method for single robot and is called PN based Motion Planning Method for Single robot (PN-MPMS) formulated in algorithm (2):

Algorithm (2): PN-MPMS

Input: Workspace bitmap image, start position S and goal position G

Output: Collision free path connecting S to G

Step1: Generate the PN model for the configuration space

Step2: find path using Pathfinder

Step3: return shortest path

6. Path Planning with Many Robots

In multi-robot systems, path planning can be performed for each robot independently with the same method (decoupled approach section 2). One problem with this simple approach is that paths will often overlap, and two robots having a path segment in common and traveling towards each other will be in danger of collision.

In this method paths for each robot are planned using PN-MPMS independently. To assure that collisions do not occur between multiple robots traveling through the same free region several conditions to be tested before firing any transition. These conditions suppose all robots are invariant robot and are identical in their shapes, size, and capability. The proposed conditions are below.

Waiting time and priorities

The problem may arise if multiple robots to cross the same path intersection. To solve this problem defining waiting time may use a simple approach. To Force one robot to wait requires modification of its enabling time M_d with ω which is the required time to let the collided robot to cross the intersection. ω is defined as the required time to cross the safe

distance between two robots. Conversely, α is defined as the required time to cross single place.

$$M_d(a)=M_d(a)+ \omega \text{ --------(7)}$$

The robot with shortest path length assigned the highest priority and the other ordered according to their lengths. The robot with low priority is forced to wait in the path intersections to avoid collisions. The proposed method for multiple robots is called PN based Motion Planning Method for Multiple robots (PN-MPMM) is formulated in algorithm (3).

Algorithm (3): PN-MPMM

Input: workspace bitmap image, number of robots (nr), start position s_i and goal position g_i for each one

Output: coordinated collision free paths for all robots

Step1: path planning

For i=1 to nr Do

Path (i) =PN-MPMS (s_i,g_i)

Step2: Test path intersection between the planned path and update time of traveling each path intersection using ω

7. Simulation Results

In this section simulation result of running TPN-MPM using Pentium III 1.2 GHz personal computer and image size 300X300 are presented. The purpose of simulations is to show that the implementation works as planned and with different workspaces cluttered with different shapes of obstacles the output is shown in Figure(4).

8. Conclusions

This work comes out with the following conclusions:

1. PN-MPM is off-line motion planning method since it solves motion-planning problem for static environment.
2. Complete and sound method (PN-MPM) is generated. Complete since it guarantees to yield a collision free paths if one exits, and sound since it guarantees that all its solutions are collision free.
3. PN-MPM generates collision free path since the path has no contact with obstacles.
4. The derived method can plan a path for the robot in the workspace cluttered with rectangle shapes obstacles (example in Figure (4A) or different shape obstacles as in example in Figure (4 B and C).
5. PN-MPM solves motion planning problem for single mobile system by using PN-MPMS part, and solves motion planning problem for multiple mobile system by using PN-MPMM part.

References

1- Chen C. and Hwang K.,"SANDROS: A Dynamic Graph Search Algorithm for Motion Planning", IEEE Transactions on Robotics and Automation, vol. 14, no. 3, June 1998.

2- De Berg M., Van Kreveld M., Overmars M., Schwarzkopf O.,"Computational Geometry Algorithms and Aplications", Springer, second revised Edition, 2000.

3- Kortenkamp D., Bonasso R., Murphy R., "Artificial Intelligence and Mobile Robots: Case Studies of Successful Robots Systems", MIT Press, 1998.

4- Javier L., Dario M., and Jose G. ,"Topological Modeling with fuzzy Petri Nets for Autonomies Mobile Roobot" , Proc. of 11th Int'l. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA-98-AIE, A.P. del Pobil, J.

Mira and M. Ali (eds.), Lecture Notes in Artificial Intelligence, 1416, Springer-Verlag, pp. 290-299, Berlin, 1998.
5- Lui H. ,Kuroda S., Nanwa T., Noborio "A practical algorithm for planning collision-free coordinated motion of multiple mobile robots" ,Proceedings conference on Robotics and Automation ,IEEE Comput. Soc Press vol 3, pp. 1427-32, 1989.
6- Murata, T.,"Petri Nets: Properties, Analysis, and Applications", Proc.IEEE, Vol.77, pp.541-579, April 1989.
7- Johansson R. "Intelligent Motion planning for a Multi-Robot System", Master's thesis in Computer Science at the school of Computer Science and Engineering, Royal Institute of Technology 2000.
8- Steven M. L.," Motion Strategy: Algorithms and applications", University of Illinois, Copyright 1999-2001.
http\\www.cs.rpi.edu/~sakelia/rmp01, latest visit 6-1-2004.
9- Latombe C. "Robot Motion Planning", NewYork, Klwuer, 1991.
10- Lozano T.," Spatial planning A configuration space approach", IEEE Tr. Computers, C-32 (2), 108-120, 1983.
11- Lumelsky J. and Harinarayan K.R.," Decentralized Motion Planning for Multiple Mobile robots: The Cocktail Party Model", Autonomous robots 4, 121-135, 1997.
12- Narahari, Y., "Petri Nets: Applications", Reasonance, pp. 44-52, September 1999.
13- Keigo K., Kengo I., and Toshimistu U.," LLP Supervisory Control with Timed Petri Net Models in Mobile Robots", Department Of Systems and Human Science, Osaka University 1-3 Machikaneyam, Toyonaka, Osaka 560-8531, Japan.
http\\www.kklabb.ces.kyutech.ac.jp/~kobayasi/smc01.pdf last visit 6-1-2004.
14- Agerwala T., "Putting Petri Nets to Work", IEEE, pp85-94, December 1979.

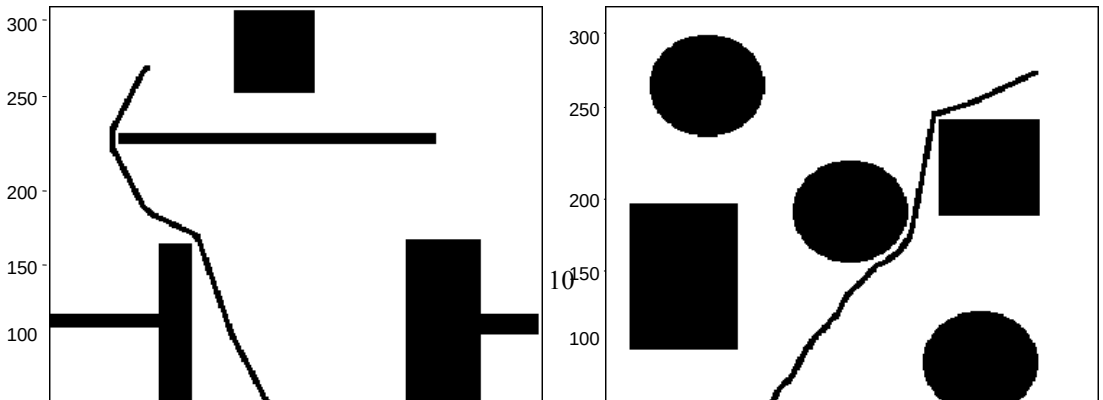

Figure (4) A: workspace with rectangle shape obstacles and single robot. B: workspace with different shape obstacles and single robot. C: workspace with different shape obstacles, five robots (A1, A2, A3, A4, and A5). Each robot has its planned path (shortest path is the thick lines).