Dependable Navigation for Multiple Autonomous Robots with Petri Nets Based Congestion Control and Dynamic Obstacle Avoidance

Lan Anh Trinh¹ · Mikael Ekström¹ · Baran Cürükli¹

Received: 7 January 2021 / Accepted: 2 February 2022 / Published online: 2 April 2022
© The Author(s) 2022

Abstract
In this paper, a novel path planning algorithm for multiple robots using congestion analysis and control is presented. The algorithm ensures a safe path planning solution by avoiding collisions among robots as well as among robots and humans. For each robot, alternative paths to the goal are realised. By analysing the travelling time of robots on different paths using Petri Nets, the optimal configuration of paths is selected. The prime objective is to avoid congestion when routing many robots into a narrow area. The movements of robots are controlled at every intersection by organising a one-by-one passing of the robots. Controls are available for the robots which are able to communicate and share information with each other. To avoid collision with humans and other moving objects (i.e. robots), a dipole field integrated with a dynamic window approach is developed. By considering the velocity and direction of the dynamic obstacles as sources of a virtual magnetic dipole moment, the dipole-dipole interaction between different moving objects will generate repulsive forces proportional to the velocity to prevent collisions. The whole system is presented on the widely used platform Robot Operating System (ROS) so that its implementation is extendable to real robots. Analysis and experiments are demonstrated with extensive simulations to evaluate the effectiveness of the proposed approach.

Keywords Dependable path planning · Dipole field · Obstacle avoidance · Congestion control

1 Introduction
Path planning is one of the key components of a robotic control system, with the function to drive a robot from a starting to a goal position without crossing through obstacles, and do this in an optimal manner. Although the path planning problem has been addressed since the 1950s [1, 2], it has not been solved thoroughly until today [3]. Over the years many solutions have been restricted to static environments or single robot cases. In general, the path is optimised with respect to a cost function, which is primarily based on the total length of the path, and in some cases other factors, such as the complexity of the terrain, areas of interest, and other environmental factors. Recently, new challenges have emerged for path planning of fully autonomous robots within the Industry 4.0 context [4], where a mobile industrial robot is no longer separated from humans in space. More specifically, the robot shares the workshop floor with workers, and it interacts with them as well as other robots.

The navigation is crucial in such problem domains, since these scenarios demand collision avoidance capability even if there are only means to predict the exact movement of moving obstacles to a certain degree. Thus, when a path planning algorithm is designed, industrial standards should be considered. Dependability is important in industrial system engineering with its focus on a system’s (i) availability, (ii) reliability, (iii) safety, and other properties such as (iv) maintainability, (v) integrity, etc.

In this paper, a dependable path planning algorithm is evaluated with respect to the first of those properties, i.e the points (i)-(iii). Reliability is assessed by the correctness of the control system to drive a robot to the goal. Availability is reflected by continuity of the system within a time limit, not having the dead-lock situation which happens when the
robot takes a very long time to reach a target, or even not able to do so. Lastly, safety is evaluated by the obstacle avoidance abilities of the robot to prevent collisions with static and moving objects.

In general, a path planning algorithm is separated into global and local path planning. The former with the target to find a way from a start position to a goal having no intersection with any static obstacles in the map, while the latter will help the robot to avoid collisions with moving obstacles. So far, numerous global and local path planning algorithms have been developed with the ambition to improve dependability of navigation systems. Efforts have been made with the global path planning to find a feasible and effective route for a robot [5–7]. In many cases, a map of configuration space is represented as a graph and a graph-based searching algorithm like A*, is used to find a collision-free path [2, 8]. The Theta* algorithm [9] adds line-of-sight checks into A* so that the path to the goal becomes more straight, thus contains fewer turns. Dynamic A* adapts to the changes of a static map in real-time, allowing the path planning to be more robust in unknown environments. Other common approaches explore the map by sampling, such as probabilistic road map (PRM) [10], with the goal of building the map by one-time sampling or rapidly-exploring random trees (RRTs) [11] to generate samples incrementally from the initial position until reaching the goal.

For multiple robots, the multi-agent path finding (MAPF) algorithms have been developed to search for optimal paths to drive the robots to their goals without conflicts. The conflicts happen when several robots occupy the same place at an instance of time. In general, the MAPF problem is nondeterministic polynomial (NP) hard [12] where the searching space grows exponentially with respect to the number of robots. A* -based search [13, 14] was one of the early solutions to perform a search with heuristics in higher dimensional space with multi robots. Conflict-based search (CBS) [15] can be used to find an optimal solution by a repeated process. First, CBS initialises a root node with all paths as the shortest paths of robots to the goal without concerning other robots. Continuously CBS expands the tree search starting from the root node by adding child nodes with conflicting constraints to avoid collisions. The constraints about potential conflicts are used to refine the paths at each node. Enhanced CBS [16] solved the MAPF problem within a suboptimal bound. Cooperative A* (CA*) [17], also a suboptimal and incomplete approach, prioritises the robot to assign the path by checking the number of collisions of the shortest path as compared to the planned paths. Priority-based search (PBS) [18] is the combination of CBS and CA* to improve the priority order of CA*. In overall, there has been a lack of a MAPF solver dealing with complex constraints like allowing several robots to meet at an intersection or one robot to wait for others to pass through the intersection. This paper investigates an effective tool to analyse and find optimal settings for global paths of multiple robots on those situations.

Once the global path to the goal has been generated, obstacle avoidance with local path planning ensures the safety for both robots and humans by preventing collisions between them. In velocity-obstacle (VO) approaches [19], a velocity obstacle is defined as a forbidden zone in the velocity space of the robot where a velocity vector in that zone could result in a collision with moving obstacles. The reciprocal velocity obstacle (RVO) [20, 21] implements VO in a distributed manner where each robot has responsibility to avoid collisions with the rest by choosing a velocity outside VO. However, after escaping from the collision, the robot tends to come back to its preferred velocity, possibly leading into a reciprocal dance. To address this problem, the optimal reciprocal collision-avoidance (ORCA) [22] casts the solution with a linear programming optimisation. Simultaneously, the hybrid reciprocal velocity obstacle (HRVO) [23] seeks for the velocity of the robot more towards one side of the half plane of VOs. Nonetheless, VO-based methods work with the assumption that the velocity of a robot is not changed over a time short interval. Therefore, these kind of approaches are more suitable to prevent potential collisions within a local range. Alternatively, a learning-based approach with reinforcement learning, has been investigated using Markov decision process to learn from history to predict the next movement of the robot with respect to observation [24]. Yet, a long training process needs to be done to collect sufficient amount of data to perform planning in both local and global scale. Conventional, however still widely applicable for local path planing, are potential field and dynamic window methods [25]. Potential field is defined as the combination of a repulsive field to push robots away from obstacles and attractive fields to drive a robot to a goal. Meanwhile, dynamic window approach (DWA) [26] generates a set of possible future trajectories of a robot with respect to the current dynamic configuration and evaluates these trajectories to find the optimal solution. As aforementioned, the manner global and local path planning systems allowing navigation of several robots into one narrow area might make the robots to block each other, i.e., lead to dead- or live-lock situations. To address the problem of congestion when routing multiple robots into a crowded place, a group of the robots should proactively define effective routes to reduce conflicts with each other. The problem presented here has some similar key features with traffic control in urban areas. The most common method used to ensure safe transportation is to have traffic signals to control the vehicles and pedestrians through cross ways. A modern monitoring system nowadays can collect information about traffic jams and send notification to drivers to avoid
moving toward a crowded area. These approaches lead to the introduction of a congestion control for path planning that is presented in this work to achieve dependability in a navigation system for autonomous robots in a dynamic environment.

Previously, with respect to solving conflicts and sharing of resources, it has been shown that Petri net (PN) provides an effective solution to such problems for an autonomous system [27]. The uses of PN to design autonomous robots are demonstrated by Yasuda et al. [28], Iocchi et al. [29]. This is further explored with the use of other extension with ROS on real robots by Fabre et al. [30] and also with fault tolerance by Miyagi et al. [31], Lusier et al. [32]. Although there have been previous works of using PN on path planning [33], they have mainly focused on the discrete system of robots changing states from one node to another in a static graph, where a node is a particular place or area representing a conflict resource to be resolved with PN. Such a static configuration is highly dependent on the precise estimation of travelling time of robots among the nodes. For an example of a highly accurate navigation system, an automated guided vehicle (AGV) using marked lines on a floor is able to exactly control the moves of robot to follow the planned path. The traffic control for AGV, like zone control, has been well investigated in literature [34, 35] to synchronise multi AGVs to enter an area safely. This paper deals with a broader scenario when the robot must be able to recognise and work with humans and/or other multiple robots, in an unstructured and unknown environment.

With regard to PN as an effective tool in dependable autonomous control to resolve conflicting problems, this paper presents a new approach of using it in generating the paths of multiple robots to reduce congestion by avoiding routing many robots into the same place. There are three main contributions of the paper. First, the paper shows how to model the path finding problem for multiple robots with PN and how to analyse alternative paths of each robot to find the optimal configurations with less traffic conflicts along the paths. The optimisation on PN allows to solve a MAPF problem with complex constraints such as stopping robots at intersection to wait for other robots. Second, the paper proposes the way of monitoring moving tokens in the PN to control the traffic of robots at every path crossing to reduce conflicts. The first contribution aims at finding the optimal global path for robots, whilst the second one realises the PN control to synchronise the moving of the robots on the assigned paths. Since the PN is mainly designed to control robot movements, the collisions between robots and humans are addressed by DWA. However, the use of DWA is not effective if it only prevents collisions in a passive manner, meaning that a robot doesn’t react until the moving obstacles have moved close enough and then considers them as normal static obstacles. This type of behaviour needs to be avoided. In the third contribution, this paper shows the implementation of the dipole field algorithm on top of the DWA algorithm to present a new method for obstacle avoidance, which is based on the information about directions and speeds of the moving objects. Moving obstacles are described by magnetic dipoles with their moments aligned with the direction of velocities. As a consequence, opposite dipoles will generate repulsive forces to turn the robot so that they do not collide with the obstacles. The capability of PN control with dipole field on DWA allows the whole proposed path planning system to handle both static and dynamic obstacles for the robots’ paths. This paper has been extended from initial studies presented in [36]. In [36], the method to construct a PN is to divide the working space into non-overlapping regions with a grid layout where each cell of the grid corresponds to a PN control place. However, a robot usually crosses many grid cells along its trajectories and some of them never play a role in controlling congestion as they are not the junctions of two or more robots’ trajectories. This leads to unnecessary defined spaces and transitions, resulting in a sparse PN to control robot movements. In this paper, the following developments have been made to further improve the prior work:

- A comprehensive method developed to seek for all intersections of global paths to construct a topological map of intersection nodes.
- A new way to construct a more compact and efficient PN where the control places are created only at intersections.
- Multiple path planning and a new algorithm to find alternative paths to the goals.
- The formulation of an optimisation problem to select appropriate global path for every robot.
- The utilisation of firing tokens of the created PN to find the optimal path selection.
- The uses of multiple PNs in parallel to handle a large number of robots in a large terrain.

In overall, the whole system is evaluated with extensive evaluations of different scenarios. The system is also integrated in the navigation stacks of ROS, one of the most widely used robotic frameworks. The rest of the paper is organised as follows. Through Section 2, the problem of this work is stated, following by the proposed path planning algorithm. Thereafter, in Section 3 the simulation results to evaluate the proposed system are presented. Finally, contributions of the paper and future directions are summarised and concluded in Section 4.
2 Methodology

2.1 Problem Statement

The basic annotations and preliminaries for the equations to be used are defined as follows.

Robots This work copes with a team of \( N \) identical robotic agents \( \mathcal{A} = \{a_1, a_2, ..., a_N\} \) which are simply presented by a unit-circle model with the identical radius \( r \). The robots are freely moving on 2-D plane \( \mathcal{M} = \mathbb{R}^2 \). The location of each robot \( i \) at time \( t \) is denoted by \( x_i(t) \in \mathbb{R}^2 \), with corresponding velocity \( \dot{x}_i(t) \). The area occupied by a robot is denoted by \( C_i(x_i(t)) \subset \mathbb{R}^2 \), a circle with the centre at position \( x_i(t) \) and radius \( r \).

Map and Static Obstacles The working environment is presented with a simple binary grid map where a set of static obstacles (white pixels) are described by \( \mathcal{O} \subset \mathbb{R}^2 \) and robots are freely moving within the black regions of the map \( \mathcal{M} \setminus \mathcal{O} \). Let \( \mathcal{O}_r \) be the subset of \( \mathcal{O} \) dilated by the robot’s radius \( r \). It is obvious that if there is no intersection of the moving trajectory of the robot trajectory with \( \mathcal{O}_r \), then the motion of the robots is collision free with respect to the static obstacles.

Humans and Moving Obstacles There are \( M \) human subjects present in the working environments together with the robots. The information about current locations, planned paths, velocity, accelerations, etc. are shared among the robots, however, they are unable to know the exact trajectories of humans. The information about the dynamics of the humans, e.g., location \( o_i(t) \) and velocity \( \dot{o}_i(t) \), are only estimations that are available with every observation, where \( j \) is the index of a human observed by the robot \( i \), \( 1 \leq j \leq n_i, n_i \leq M \). Assume that the human is occupying a circle \( \mathcal{H}_{r_h}^i(o_j(t)) \) with the centre at \( o_j(t) \) and radius \( r_h \). In this paper, all humans will be presented by same radius. Each robot \( i \) will then see a set of moving obstacles/humans as \( \{\mathcal{H}_{r_h}^1(o_1^j(t)), \mathcal{H}_{r_h}^2(o_2^j(t)), ..., \mathcal{H}_{r_h}^{n_i}(o_{n_i}^j(t))\} \).

Global Paths Each robot is assigned a task to move from a starting point to a goal. Using a global guidance system, a number of alternative paths are planned. These global paths are updated frequently to the changes in robot trajectories. Let a set of global paths for a robot \( i \) be \( \{S_i^1, S_i^2, ..., S_i^k\} \) where the path consists of connected line segments. Different paths that are leading into the same goal will simply extend the area \( \mathcal{G}_i \), possibly defined by a circle with a predefined radius.

The problem to be addressed in the proposed work has the aim that all robots should be able to complete their navigation tasks without colliding with any static and moving obstacles in the working environment. For all times from \( t_i^0 \) to \( t_i^1 \) for a robot \( i \) to travel along its trajectory, the collision constraints are expressed by,

\[
C_i(x_i(t)) \cap \left( \bigcup_{j \in \{1,2, ..., N\}, j \neq i} \mathcal{O} \bigcup \bigcup_{k \in \{1,2, ..., n_i\}} \mathcal{H}_k^i(o_k^j(t)) \right) = \emptyset \quad \forall t \in [t_i^0, t_i^1] \quad (1)
\]

which is correspondent to no intersection of the trajectory with static obstacles, other robots and moving obstacles in its environment. The requirement to reach the goal is given by:

\[
C_i(x_i(t^1)) \cap \mathcal{G}_i \neq \emptyset. \quad (2)
\]

Routing several robots into narrow areas could lead to congestion, that may lead to a deadlock situation or prolong the time to complete the navigation task. This is due to the reason that robots need to go around to avoid collisions with each other. The main problem to be solved in this work is to tackle the issues defined by Eqs. 1 and 2 with the use of PN to synchronise the movements of robots to find less not only collision- but also congestion-free paths to reach their goals correctly.

The solution starts with the construction of multiple paths for each robot (Section 2.2). A path, which is presented by a set of line segments connecting from a starting point to a goal, is found by the Theta* algorithm. Continuously, each robot broadcasts its paths to all robots within the working space. By checking the intersections of the planned paths, PNs are constructed to analyse the movements of the robots when they enter the intersection regions (Section 2.3). In each PN, a single robot is correspondent to one token of the network. An optimisation algorithm is performed to choose the path for each robot to minimise the conflict/congestion and the travelling time to complete the task (Section 2.4). Once the optimal path is found, the robot then follows the path and synchronises with other robots to avoid collisions (Section 2.6). Collisions with other obstacles that are not able to communicate their planned trajectories with the robots, e.g., humans, are addressed by using DWA combined with a dipole field (Section 2.7). In the case that one of the robots is getting stuck, the global path planning is reactivated to generate a new path from current position to the goal, and PN planning is updated with a new moving control plan. The overall architecture of the system is described in Fig. 1.

2.2 Multiple Global Paths

A global path planning algorithm needs information regarding the environment e.g. as a scanned map beforehand to generate a path from a starting point to a desired goal.
The global path planning mainly applies for a static environment. This means that in order to compute the global path to a goal for the robot, the presence of other robots and humans, i.e. dynamic objects, is not taken into account. Theta* [9] has realised an any-angle path algorithm, which is able to find an optimal path in any direction, by adding a line-of-sight detection function to each search iteration. Unlike A* or Dijkstra, the path found by Theta* is a connection of line-of-sight nodes so that the generated path is smoother, more realistic, and has fewer turns. With regard to those advantages of Theta*, it has been used as the main global path planning in this paper. To generate multiple paths, Theta* is utilised to sequentially find a new path after inserting a set of found paths into an obstacle map. This way, the next found path does not overlap with the previous ones (Fig. 2). A backup path is critical if the robot needs to pass through a narrow area with the possibility of facing another robot so that it has to find an alternative way. On an empty space, the multiple paths created by this approach can be close to each other. Therefore, to reduce the complexity of the control PN, multiple paths are only added if separation among them are big enough. The algorithm to find multiple paths from a starting point to a destination is summarised in Algorithm 1.

2.3 Petri Net Construction

The basic definitions of a PN are given as follows.

Definition 1 (Petri net) PN is considered as a network of places, arcs and transitions. Mathematically, it is defined as a bipartite graph of a set of tuples \(\langle P, T, W \rangle\), where \(P = \{p_1, p_2, ..., p_P\}\) and \(T = \{t_1, t_2, ..., t_T\}\) are disjoint sets of places and transitions, \(|P|\) and \(|T|\) are the number of

Fig. 1 The overall architecture of the proposed navigation system

Fig. 2 Multiple path generation. (A) The first path created by Theta*. (B) Inserting the first path into the binary obstacle map. (C) The second path also made by Theta* but separated from the first one
Algorithm 1 Finding multiple global paths.

Input: A pair of a source $s$ and destination $d$, static obstacle map $O_r$, $N_p$, maximum number of paths

Output: A set of global paths $Ξ = \{ξ_1, ξ_2, ..., ξ_n\}$ from $s$ to $d$, $n \leq N_p$

\[
Ξ \leftarrow \emptyset;
\]
\[
i \leftarrow 0;
\]
\[
is\_find\_next \leftarrow \text{True};
\]
while $i \leq N_p \land is\_find\_next$ do
    Find a new global path $ξ_i$ from $s$ to $d$ on $O_r$ using Theta*;
    for $ξ_j \in Ξ$ do
        Present global paths by a sequence of points in 2-D (dimensional) space, $ξ_i = \{s_1^i, s_2^i, ..., s_l^i\}$,
        $ξ_j = \{s_1^j, s_2^j, ..., s_l^j\}$ where $l_i, l_j$ are the number of points of the two paths respectively;
        $d_1^i, d_2^i, ..., d_l^i$ $\leftarrow$ a set of minimum Euclidean distance of each point $s_l^i$ to $ξ_j$;
        $d_1^j, d_2^j, ..., d_l^j$ $\leftarrow$ a set of minimum Euclidean distance of each point $s_l^j$ to $ξ_i$;
        $d(ξ_i, ξ_j) = \max\{d_1^i, d_2^i, ..., d_l^i, d_1^j, d_2^j, ..., d_l^j\}$;
        if $d(ξ_i, ξ_j) < \text{threshold}$ then
            $is\_find\_next = \text{False}$;
            break;
        else
            Dilate $ξ_i$ by the radius of the robot and add it into $O_r$;
            $Ξ := Ξ \cup ξ_i$;
        end
    end
end

elements of $P$ and $T$ respectively, and $W \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting from a place to transition and vice versa. □

In a PN, an input place of a transition is defined as the place from which an arc goes into the transition. Meanwhile, the contrasting one, i.e, the output place, connects to an arc going out from the transition. Both the input and output arcs of the transition are added by non-negative integer weights. With regard to a set of output weights $O$ and a set of input weights $I$, PN is described as a set of five tuples $G = \{P, T, W, O, I\}$. Places in a PN may consist of a number of marks named tokens.

Definition 2 (Marking) The marking $M$ is expressed as a vector $[M(p_1), M(p_2), ..., M(p_i), ..., M(p_j)]^T$, in which $p_i$ is a place, $|P|$ is the number of places in PN, and $M(p_i)$ is the number of tokens at the place $p_i$. □

Let $O$ be a two dimensional matrix of weights $O(p_i, t_j)$ from the place $p_i$ to the transition $t_j$. $I$ is similarly defined by the weight $I(t_j, p_i)$ from the transition $t_j$ to the place $p_i$. It is noted that $1 \leq j \leq |T|$, where $|T|$ is the total number of transition. Thus, a change of the marking vector of a transition from $M$ to $M'$ is given by a finite sequence of transitions

\[
M'(p) = M(p) + I(p, t) - O(p, t), \forall p.
\]  (3)

It is said that the marking $M'$ is reached by the marking $M$ by firing $t$. In the vector form, this transition marking is presented as

\[
M' = M + I_p - O_p,
\]  (4)

where $I_p$ is the column vector $p$-th of the matrix $I$, and $O_p$ is the column vector $p$-th of the matrix $O$. In general, if the marking $M'$ is reachable from $M$ by a finite sequence of $k$ transitions $t_1, t_2, ..., t_k$, then Eq. 4 is rewritten as

\[
M' = M + C \cdot ξ
\]  (5)

where $ξ = [σ_1, σ_2, ..., σ_{|T|}]^T$ is a vector where each element $σ_j$ counts the number of times $t_j$ appears in the sequence $σ$. An example of PN described in Fig. 3 shows the changes of the marking from $M = [1, 2, 0]^T$ into $M' = [0, 1, 2]^T$ after the transition $t_0$ fires.

With an initial marking $M_0$, the full description of a PN consists of six tuples $G = \{P, T, W, O, I, M_0\}$. A full graph of all possible markings and transitions, i.e. a reachability set, is described by state-space analysis.

The pipeline to formulate the control PN from multiple global paths includes four main steps: First, a set of intersection regions are found by checking all crossings of the global paths; Second, the list of intersection candidates are reduced by merging close intersections; Continuously, the control places and the places before and after an intersection are located; Finally, flowing all the links between the places, the PN is created.

First of all, to find the area of intersection, the thickness of the path is dilated by the radius of the robot. For a pair of paths, there are one or more intersections formed along the paths. Due to the dilation of the path, an intersection is not presented as a point, but a polygon. Consequently, small polygons are filtered out based on their perimeters. The centre of a polygon is estimated by averaging all middle points of the segments passing through the polygon. PN is applied to avoid the congestion by granting permission to just one robot to pass through an intersection at an instance of time. The trajectory crosses that happen inside a narrow area may even lead to an unexpected situation of making
robots getting locked and not able to escape from the congestion area. Therefore, other control places are created to handle such an area to limit the number of robots allowed to pass through it. For simplicity, those types of areas are manually given in this work by providing a list of polygons to cover the restricted areas.

Meanwhile, it is also inefficient to have control places which are very close to each other. To address this issue, based on the positions of the intersection, the places are grouped by hierarchical agglomerative clustering [37] until the distances among every places are greater than a predefined threshold. It is also important to avoid creating places nearby the sources, goals, and the restricted areas. The combination of merging the paths and remaining only those distances are estimated by Algorithm 2.

In the next step, each place of the PN corresponds to a region of intersection and is assigned the location at the centre of that region. An example of creating a control place is described in Fig. 4 with three control places numbered 6, 7, and 8, respectively. The three green circles present the starting positions while the three blue circles are the destinations. Three control places are marked with red circles. Control place 8 lies at the middle of the intersected area of two paths. Meanwhile, the control place 7 is formed by grouping the intersection from three paths. The restricted area 6 is defined to be inside a corridor to allow only one robot to pass through at a time. Although there is a region of intersection and is assigned the location at the centre of that region.

In another case described by Fig. 6.B, \(p_i^{2j}\) and \(p_i^{2j+1}\) are placed at the starting or the ending point of the path.

Finally, the PN model is constructed by Algorithm 3 to synchronise the movements of multiple robots. It is noted that a cross is considered as a space resource allocated to a single robot each time. In this algorithm, a place of PN is added according to a cross and a transition represents a physical connection between two adjacent crosses on a path. An example to add a PN place to control the movements

---

**Algorithm 2 Finding intersections of global paths.**

**Input:** A set of global paths \(\Xi = \{\xi_1, \xi_2, \ldots, \xi_n\}\) planned for every robots of the team and a set of restricted areas \(\mathcal{R} = \{R_1, R_2, \ldots, R_r\}\)

**Output:** All intersections \(\Upsilon = \{I_1, I_2, \ldots\}\) among global planned paths

Let dilate each path \(\xi_i, \forall i \in [1, n]\) by the radius of the robot; \(\Upsilon \leftarrow \emptyset\);

for \(\xi_j \in \Xi\) do

for \(\xi_i \in \Xi\) do

Let \(I_{ij}\) be the intersected polygon of two paths \(\xi_i\) and \(\xi_j\);

if \(\text{perimeter}(I_{ij}) > \delta_p \wedge \text{distance}(I_{ij}, \text{source/goal}) > \delta_{sd} \wedge \text{distance}(I_{ij}, R_k) > \delta_r(\forall R_k \in \mathcal{R})\) then

\(\Upsilon := \Upsilon \cup \{I_{ij}\}\);

end

end

end

HierarchicalAgglomerativeClustering(\(\Upsilon\)); \(\Upsilon := \Upsilon \cup \mathcal{R}\);

---

Fig. 3 A PN example with three places \(p_0, p_1\) and \(p_2\) and one transition \(t_0\). (A) A PN with an enabled transition. (B) The PN after the transition fires.
Algorithm 3 Construction of PN model.

**Input:** A team of robots with initial positions, a set of global paths \( \Xi = \{\xi_1, \xi_2, ..., \xi_n\} \) planned for every robots of the team and the intersection set \( \Upsilon \)

**Output:** The PN model \((P, T, W, M_0)\) of the team

Let \( P \leftarrow \emptyset, T \leftarrow \emptyset, W \leftarrow \emptyset, M_0 \leftarrow 0 \);

for \( \xi_i \in \Xi \) do

Find a set of intersections \( I_1^i, I_2^i, ..., I_k^i, ..., I_n^i \in \Upsilon \) which are crossed by \( \xi_i \);

Add two places \( p_{2j}^i \) and \( p_{2j+1}^i \), before and after an intersection \( I_j^i \), into PN \( P = P \cup \{p_{2j}^i, p_{2j+1}^i, ..., p_{2n_i}^i\} \) for every \( \{I_j^i|\forall j\} \);

Add source place \( p_1^i \) and destination place \( p_{2n_i+2}^i \);

Add one token into the starting place \( M_0[p_1^i] = 1 \);

for \( j \leftarrow 1 \) to \( 2n_i+1 \) do

Add transition \( t_{i,j} \) to \( T \);

\( \tau_{i,j} \), the delayed time at \( t_{i,j} \), is approximated by the travelling distance between two places;

\( W := W \cup \{(p_j, t_{i,j}), (t_{i,j}, p_{j+1})\} \);

end

end

Find all cells \( C = \{c^1, c^2, ..., c^n\} \) that are passed by at least two paths;

for \( c^1 \in C \) do

Add a control place into PN \( P = P \cup \{c^1\} \);

Add one token into this control place \( M_0[c^1] = 1 \);

Add transitions to connect \( c^1 \) with related places with a description given in Fig. 7;

end

2.4 Estimation of Optimal Configuration

When a path has many crossings, it means that the robot’s travelling time is prolonged. Therefore, the PN construction is improved by first setting multiple paths for each robot to its goal and then choosing the optimal configuration of PN to minimise the travelling time of robots and to avoid deadlock situation. In Algorithm 3, \( T = [\tau_1, \tau_2, ..., \tau_{|T|}]^T \) is the approximation of the delayed travelling time over a transition. Assume that all robots are configured to have the same speed, the travelling time is therefore proportional to, or for simplicity is equal to, the travelling distance from one transition into another. With regards to \( T \), the firing rule is modified to consider the delayed time in the transition firing: After a transition \( t_i \) is enabled, it will fire after \( \tau_i \) time units. Those extensions actually introduce timed PN
with an associated finite firing duration to each transition in PN. Assume that after $n$ intermediate markings, all the moving tasks are accomplished from the initial marking $M_0$. Let $M_j$, $j = 1, 2, ..., n$ denote a sequence of markings and the firing count vector $ς_j$ that makes the moves of one marking to the next, $M_j = M_{j-1} + C \cdot ς_j$, $j = 1, 2, ..., n$.

The total of the corresponding travelling times of all robots after firing the sequence $ς_j$, $j = 1, 2, ..., n$ is estimated by

\[
\sum_{i=1}^{N} f(ς_j) = \sum_{j=1}^{n} f(ς_j), \quad \text{where } f(ς_j) \text{ is the delayed time of the whole PN after } ς_j \text{ is performed.}
\]

For simplicity, assume that each robot is setup with the same number, $k$, of alternative paths to reach the goal. Let $ς_{i,p}$ be a set of $N \times k$ binary variables defined according to how each robot choose its path,

\[
ς_{i,p} = \begin{cases} 
1 & \text{if the robot } i \text{ selects the path } p \\
0 & \text{otherwise, } \forall p = 1, 2, ..., N.
\end{cases}
\]

The constraint that only one path is assigned to a robot is expressed by,

\[
\sum_{p=1}^{k} ς_{i,p} = 1.
\]

Therefore, $ς_{j}(ς_{i,p})$ is a firing sequence decided by the selection of $ς_{i,p}$. The optimal problem to find best assignments of paths to robots is stated as follows,

\[
\begin{align*}
\arg\min_{ς_{i,p}} & \sum_{j=1}^{n} f(ς_j) \\
\text{s.t.} & \quad M_j = M_{j-1} + C \cdot ς_j, \quad j = 1, 2, ..., n \\
& \quad \sum_{p=1}^{k} ς_{i,p} = 1 \\
& \quad M_j \in \mathbb{N}_T, \quad ς_j \in \mathbb{N}_P, \quad j = 1, 2, ..., n \\
& \quad ς_{i,p} \in \{0, 1\}, \quad i = 1, 2, ..., N, \quad p = 1, ..., k.
\end{align*}
\]

Solving the optimal problem in Eq. 8 provides a solution for the optimal path for each robot to construct the control PN. Thanks to PN simulation tools [38], different combinations of assigning the $k-th$ paths to the robots are examined to find the best configurations of $ς_{i,p}$ to minimise $\sum_{j=1}^{n} f(ς_j)$. After this, the PN is fixed with the optimal path selection (the paths which are not selected are disabled) and the use of PN control in path planning is applied in the controlling stage.

### 2.5 Computational Complexity and Solution at a Large Scale

The computational complexity of the approach is the time needed to evaluate all combinations of assigning different paths to a robot. Assume that there is a maximum of $K$ paths for each robot and $N$ is the total number of robots, the evaluation of all samples runs in $O(K^N)$. As the computational complexity grows exponentially with respect to $N$, the algorithm will take a long time to find the optimal solution with a large number number of robots. Therefore, in order to apply the solution in a large scale, in this paper, the optimisation with a subgroup is proposed. The whole working space is divided into separated zones where the movements of robots inside each zone are synchronised by a PN. A robot travels through a sequence of zones until reaching its goal. As shown in the example of Fig. 8, Robot 1 ($r_1$) has to pass through Zone 1, Zone 2, and Zone 3 to approach its goal. To join a zone, robots wait at the boundary of the zone until the controlled PN of the zone is free. Therefore, Robot 1 and Robot 2 ($r_2$) must stop until the PN at Zone 2 completes controlling Robot 3 ($r_3$) and Robot 4 ($r_4$). Let $D$ be the maximum number of robots allowed to

---

**Fig. 5** The position of $p_i^{2j}$ and $p_i^{2j+1}$ before and after the control place. (A) For intersection outside restricted areas. (B) For restricted areas

**Fig. 6** The position of $p_i^{2j}$ and $p_i^{2j+1}$ under special cases. (A) The distance between two intersections is less than $D_i^{2j} + D_i^{2j+1}$. (B) The intersection with the distance to a start of a path is less than $D_i^{2j}$ or the distance to the end of the path is less than $D_i^{2j+1}$
enter the zone simultaneously. The complexity of running optimisation at each PN controller is reduced to $O(KD)$, which is fast if $D$ is small enough.

2.6 Movement Control on Optimal Paths

The whole system is organised in a centralised manner where one master ROS node is used to create the PN model for each robot and share information with all other robots via communication channels. Once the above PN model is built, the realisation of the path planning becomes the step-by-step moving of a robot from one place to another along the generated trajectory until the robot reaches its goal. The movement of each robot is controlled by firing the enabled transitions and following a sequence of places visited by the token assigned to that robot. Each robot is linked to one token given Algorithm 3. The movement controlling algorithm for a team of robots is implemented by the Algorithm 4. The PN planning mainly deals with the congestion of robots by preventing several robots to be navigated into a narrow place at the same time. The prerequisite requirement for this control with PN is that the global planner paths are shared by the robots. However, this is not applied for humans and other moving obstacles unless they explicitly share their moving trajectories. Thus, to avoid these objects, the local path planning with DWA and dipole field is applied.

2.7 Dipole Field and DWA for Moving Obstacle Avoidance

The dipole field has been introduced by the authors for obstacle avoidance with moving human or robotic agents [39]. In this method, robots and obstacles (named agents in general) are modelled by small magnets with their dipole moment vectors aligned with the moving directions of the robots and obstacles. In a consequence, two agents moving toward each other or having the same poles facing each other are pushed far away by magnetic forces. The magnetic field $B$ of the dipole moment vector $m$ generated by an agent is given by

$$B(m, d) = \rho (3(m \cdot \hat{d})\hat{d} - m)/d^3$$

where $d$ is the distance vector, $d = ||d||$, $\hat{d} = d/||d||$, in which $||.||$ denotes the norm of the vector, and $\rho$ is a constant. The magnetic moment $m$ is designed to be aligned with the moving directions of the agent and its magnitude is proportional to the speed of the agent. An agent with the

Fig. 8 PN control by subgroups
magnetic moment $\mathbf{m}_j$ within the magnetic field $\mathbf{B}_k$ of $\mathbf{m}_k$ would be affected by the force

$$F_{jk} = \nabla \mathbf{m}_j \cdot \mathbf{B}_k = \rho \nabla \left( \mathbf{m}_j \cdot \frac{3(\mathbf{m}_k - \hat{\mathbf{d}})\hat{d} - \mathbf{m}_k}{d^3} \right)$$  \hspace{1cm} (10)

where the gradient $\nabla$ presents the change of potential $\mathbf{m}_j \cdot \mathbf{B}_k$ generated per unit distance, and $\rho$ is a constant.

Dynamic window approach (DWA) was introduced by Fox et al. [40] and later developed in ROS [26] to use multiple constraints of velocity limits, of acceleration limits, and of following the predefined global path into the local path planning. The local searching space is reduced to dynamic windows in a three-step progress (Fig. 9). Firstly, DWA considers the robot’s trajectories as circular trajectories or curvatures determined by a set of translation and rotation velocities ($v_i = \dot{x}_i(t), \omega_i$). Secondly, only admissible pairs of ($v_i, \omega_i$) corresponding to their trajectories are considered if the robot is able to move forward without colliding with obstacles. Finally, the dynamic window limits the admissible velocities to those that the robot can safely reach to the goal in a short time with optimised accelerations.

Only the pair of $(v_i^*, \omega_i^*)$ is selected if the objective function reaches to maximum,

$$(v_i^*, \omega_i^*) = \arg\max_{(v_i, \omega_i)} F(v_i, \omega_i),$$ \hspace{1cm} (11)

where $F(v_i, \omega_i) = f \left( \sum_s q_s \Omega_s(v_i, \omega_i) \right), \Omega_s(v_i, \omega_i)$ and $q_s$ are the cost function and the corresponding weight, and $f(\cdot)$ is an optional function used to smooth the weighted sum of the above components. In the implementation of DWA on the ROS navigation stack\(^1\), the objective function given by Eq. 12 is used to score the admissible velocities, $F(v_i, \omega_i) = \alpha D(v_i, \omega_i) + \beta H(v_i, \omega_i) + \gamma P(v_i, \omega_i)$.  \hspace{1cm} (12)

With regards to the grid-based presentation of the cost map, $D(v_i, \omega_i)$ is the total cost of the grids passed through by the trajectory ($v_i, \omega_i$). Meanwhile, $H(v_i, \omega_i)$ is a goal heading which is the distance from the endpoint of the trajectory to the goal, $P(v_i, \omega_i)$ is the distance from the endpoint of the trajectory to the global path, $\alpha$, $\beta$ and $\gamma$ weight each term of the equation. The trajectory with the minimised cost $F(v_i^*, \omega_i^*)$ in Eq. 12 is the one with the highest possibility of avoiding static obstacles, going toward the goal, and being close to the planned path. Additionally, the dipole field has been introduced in this paper to take into account the moving directions and velocity of the obstacles to avoid the collisions with dynamic obstacles.

To integrate the dipole field into DWA, it is noted that the potential changes reflect the repulsive forces that prevent agents to collide with each other. Therefore, the new terms $\Omega_j(v_i, \omega_i)$ are added into Eq. 12 to weight the trajectories based on the possibility of collisions or to minimise the total dipole forces applied on the robots. The dipole field term is removed if $\Omega_j(v_i, \omega_i)$ does not generate repulsive forces (two objects do not move toward each other). The pair $(v_i, \omega_i)$ and its trajectory are also removed if they will cause the collisions with another robot or moving objects with regards to the velocity constraints introduced by Fiorini and Shiller [19]. It means for every neighbouring robots $j \neq i$ around robot $i$,

$$\|x_i(t) - x_j(t) + (\dot{x}_i(t) - \dot{x}_j(t))\tilde{t}\| \geq 2r$$ \hspace{1cm} (13)

\(^1\)http://wiki.ros.org/dwa_local_planner

\(\alpha\) Springer
Fig. 10  (A) The simulated working space with two robots that are placed opposite and move toward each other. (B) The global paths of the two robots

\[
\|x_i(t) - o_j(t) + (\dot{x}_i(t) - \dot{o}_j(t))\tilde{t}\| \geq r + r_h. \tag{14}
\]

3 Simulation and Evaluation

The whole system is implemented in the Robot Operating System (ROS) platform [41] with version Kinetic Kame in Ubuntu 16.04 on a laptop with Intel i7 CPU of 8 cores and a clock frequency of 2.90 GHz. The Gazebo simulator is used to provide realistic simulation of robots working in indoor environments. The simulated robotic agents are Husqvarna research platform (HRP)\(^2\) mowers with an extra depth sensor, and a LIDAR laser scanner. The centralised control is performed by a master ROS node and robots in the working space share their locations and global path information through ROS messages. The PN planning module is programmed in Python with the support of the SNAKES library [38]. Meanwhile, the global path planning with the Theta* algorithm and the obstacle avoidance with dipole field on DWA are implemented in C++. In all experiments, it is assumed that all robots travel with the same speeds. Therefore, instead of using the time to analyse PN, the length of the travelling path is used as the delayed time at each transition in all simulated experiments.

3.1 Simulation Scenarios

Two simple scenarios are simulated in this section to describe the working principle of the system.

**Scenario 1** In the first scenario, the two robots are placed at opposite sides of a corridor where one of them moves from top to bottom and the other robot travels in the opposite direction (Fig. 10). The size of the map is set to 10 × 10m. The global path planning returns two alternative paths for each robot where it has one goal location. The shortest path

\(^2\)https://github.com/HusqvarnaResearch/hrp
Fig. 12 The conventional PN to visualise the generated PN model in Fig. 11

goes through the corridor and the other goes around the sides, outside the corridor. For both two robots described in Fig. 10.B, Path 1 is the shortest path going through the corridor while Path 2 goes by the side of the map. The green and blue circles indicate the starting and ending positions of the robots while the red one is the intersection point if both robots choose the shortest paths. Thus, if both robots choose the shortest path, they will meet each other at the centre of the corridor. A restricted area lying at the middle of the corridor is defined to allow only one robot to pass through. A timed PN is created for the whole system to analyse the best configurations for all robots (with its visualisation given by the SNAKE library in Fig. 11 and by the conventional PN definition in Fig. 12). The delayed times at transitions are given in Table 1, where the delays at transition $t_1$ and $t_5$ are 6.45 and 6.51 meters, corresponding to the travelling distance of robots through the corridor.

Different combinations of pairs of paths for the robots are composed to find the best routes for the robots. If both Robot 1 and Robot 2 select to follow Path 2, the firing sequence of transitions is $\{t_3 \rightarrow t_7\}$, the total travelling time of two robots is 24.95. Similarly, if Robot 1 chooses the Path 2 and Robot 2 chooses Path 1, those values are $\{t_0 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7\}$ and 21.02 and finally when Robot 1 chooses Path 1 and Robot 2 chooses Path 1, those are $\{t_0 \rightarrow t_4 \rightarrow t_1 \rightarrow t_5 \rightarrow t_2 \rightarrow t_6\}$ and 23.42. The best configuration achieves when Robot 1 follows Path 1 and Robot 2 is at Path 2. With the best routes selected by analysing the timed PN, the congestion does not happen in this scenario (Fig. 13).

Scenario 2 In the second scenario (Fig. 14), Robot 2 moves as previously while Robot 1 moves from left to right. In this scenario, the shortest path of one robot crosses that of the other at the centre. The size of the map is also 10 × 10m, and each robot is able to find two different paths from starting to ending points. Similar to the previous experiment, the green and blue circles mark the starting and ending location of the robots, the red circles mark the control places. For Robot 1, starting and ending points are marked by green circle nr. 0 and blue circle nr. 1. Path 1 is the straight line from 0 to 1 through the control place nr. 7 and 8. The other path is Path 2. For Robot 2, the straight line from starting point marked with green circle nr. 2 through the control place nr. 7 and 5 to the ending position marked with blue circle nr. 3 is Path 1. The other is Path 2. Based on all the found paths, different moving scenarios are proposed to select the best scenario for the robots with the timing constraints. The delays at all transitions are depicted in Table 2, where the delays at transition $t_1$, $t_3$, $t_6$, $f_8$, $t_{11}$, $t_{13}$, $t_{16}$ and $t_{18}$ are shorter than the

| $\tau_0$ | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ | $\tau_5$ | $\tau_6$ | $\tau_7$ |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 0.84     | 6.45     | 1.14     | 12.36    | 0.64     | 6.51     | 1.53     | 12.59    |
values in Scenario 1 since the robots can pass through the crosses with a shorter time. No restricted areas are defined.

The travelling time for firing sequence \( t_0 \rightarrow t_1 \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_2 \rightarrow t_3 \rightarrow t_{17} \rightarrow t_4 \rightarrow t_{18} \rightarrow t_{19} \) is 35.14 when Robot 1 moves along Path 1 and Robot 2 chooses Path 2. For the sequence \( t_{10} \rightarrow t_0 \rightarrow t_{11} \rightarrow t_1 \rightarrow t_{12} \rightarrow t_2 \rightarrow t_{13} \rightarrow t_3 \rightarrow t_{14} \rightarrow t_4 \) which is in case that both of the robots choose to follow Path 1, the travelling cost is 20.20. If the robots select Path 2 which is indicated by the sequence \( t_{15} \rightarrow t_5 \rightarrow t_{16} \rightarrow t_6 \rightarrow t_{17} \rightarrow t_7 \rightarrow t_{18} \rightarrow t_8 \rightarrow t_{19} \rightarrow t_9 \), the total travelling time is 29.73. Finally, 46.65 is for the sequence \( t_{10} \rightarrow t_{11} \rightarrow t_5 \rightarrow t_{12} \rightarrow t_6 \rightarrow t_{13} \rightarrow t_{14} \rightarrow t_7 \rightarrow t_8 \rightarrow t_9 \) when Robot 1 chooses Path 2 and the other follows Path 1. With regard to those analyses, both robots select the path with the same index 1 to follow. In this case, the PN control (Figs. 15 and 16) is applied to synchronise the movements of two robots passing through the cross (Fig. 17).

### 3.2 Evaluation

In this section, the performance of the proposed method is verified through the evaluation of all the proposed scenarios.

### 3.2.1 Multiple Robots Controlled by a Single Petri Net

An extensive evaluation is performed with four robots that are placed at different sides of the map (Fig. 18). The size of the map is set to 20 × 20m. Similar to the previous examples, each robot is able to plan two different routes to reach its goal. PN is used to analyse the best routes and to synchronise the movements of robots. The experiment is repeated 100 times by randomly defining different goal locations. For each run the starting points are kept the same. One restricted area is established inside the corridor.

The results of three example runs are depicted in Fig. 19. The first row describes the situation of the cross present inside a corridor. Without PN control, the two robots follow the shortest paths leading to the collision inside the corridor even though the two robots manage to keep on moving to reach their goals (Fig. 19(A1)). The result with PN control is depicted in the Fig. 19(A2) showing that no collisions occur as one robot goes outside the corridor. In the second row, there are figures expressing the case in which the cross is on an empty space. Without the PN control, a collision happens (Fig. 19(B1)). The PN control helps to make the moving trajectories of both robots smooth (Fig. 19(B2)). A
comprehensive example of all paths included in a single PN model is given in the third row. There is no collision in two cases of not using (Fig. 19(C1)) or using the PN control (Fig. 19(C2)). However, in the latter, with the designed path of less intersections, a robot with the red global path is able to keep a smoother trajectory to the goal.

In Fig. 20 the histogram of all occasions of when the distance between two robots is less than 1.0 meters in

---

**Table 2** The time delay at all transitions in the PN model.
The time unit is not used in experiments

| τ₀  | τ₁  | τ₂  | τ₃  | τ₄  | τ₅  | τ₆  | τ₇  |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 3.09| 1.47| 1.07| 1.47| 0.69| 4.76| 1.45| 0.80|
| 1.52| 2.46| 2.73| 1.47| 0.90| 1.47| 1.19| 4.72|
| 1.49| 0.94| 1.52| 2.45|

---

**Fig. 15** Generated PN to control the movements of two robots in the second example
Fig. 16 The conventional PN to visualise the generated PN model in Fig. 15

Fig. 17 (A) Without PN control, the two robot are not able to slow down before reaching the cross, leading to a collision. (B) With PN control, one robot stop and wait for the other passing through the cross. Two robots smoothly approach the goals with no collision. Note that the two global paths are highlighted by bold lines while the thin ones are the real trajectories.
Fig. 18 Gazebo simulation of a working space with four robots

Fig. 19 A scenario with four robots with a cross inside corridor. The trajectories of the robots without PN control $(A_1), (B_1), (C_1)$ and with PN control $(A_2), (B_2), (C_2)$. Robot 1: red, Robot 2: green, Robot 3: purple, and Robot 4: blue. Thin lines are planned paths while the thick ones are the real trajectories.
the overall 100 trials. The results are compared with the DWA baseline algorithm on ROS [26]. The mean of those distances is 0.76m for the PN control with a standard deviation (STD) of 0.14m. Similarly, the mean and STD are 0.73m and 0.14m for the DWA-baseline algorithm. Using PN to synchronise the movements of robots, the occasions when the distance is less than 0.7 meters are reduced significantly. Note that collisions risk to happen when the distance between two robots is in the range [0.5, 0.7] meters, dependent on whether they are moving toward each other in an opposite direction or they glide over each other. The summary of the number of collisions and dead locks are given in Table 3. The deadlocks happen when the two robots have a strong hit that makes them stop or a robot is locked inside a corridor. There are several cases where a goal lies on the designed global path of another robot. Once a robot finishes its path and ends at the goal, it consequently stops its PN control, leading to a collision and/or even a dead lock. In these cases, two dead locks are recorded although PN control is used.

### 3.2.2 Multiple Robots Controlled by Multiple Petri Nets

The optimisation by subgroups proposed in Section 2.5 shows the way to search for optimal global paths with a large number of robots in a working space. An experiment is designed in this section to evaluate the effectiveness of the proposed approach. The map with the size of 40 × 40m with several obstacles and narrow corridors is used. Sixteen robots are distributed equally on each side of the map. The starting positions of the robots are fixed while their goals are uniformly assigned with the separation of 3 meters. The minimum distance between a starting point and its corresponding goal is 5 meters. As depicted in Fig. 21, the whole working space is divided into 9 non-overlapping zones where each zone is controlled separately by one PN controller. Zone 4 takes into account only restricted areas (narrow corridors in the map) to create a PN. The maximum number of 8 robots is allowed to simultaneously enter a zone while at most three global paths are assigned for a source-destination pair. The experiment is repeated 10 times so that a total number of 160 path planning tasks are evaluated. The proposed algorithm is compared against the baseline DWA [26] and VO-based DWA path planning [25]. Table 4 summarises the success rate of completing the navigation tasks of the three methods. The baseline DWA mainly fails due to the collisions of robots in both empty spaces and narrow areas. The VO-based DWA has a deadlock problem when two robots meet each other in a

|                  | DWA Baseline | w PN Control | w PN Control** |
|------------------|--------------|--------------|----------------|
| Collision        | 18           | 8            | 5              |
| Deadlock         | 8            | 2            | 0              |

Table 3 The number of collisions/deadlocks of multiple robots with DWA baseline, with PN control (*), and with PN control when the cases of a goal on the global path of another robots are not considered (**).
Fig. 21 Multiple PN controllers on different zones. Note that the thin lines are the multiple planned paths and the thick ones are the actual moving trajectories.

narrow corridor. Meanwhile, the failures of the PN-based control usually happens with a single robot when it has a sharp turn and collides with a wall. In overall, the PN-based control is superior to the other two methods on its reliability to complete the path planning tasks in this experiment. The processing time (averaged over 50 trials) of running a PN to find the optimal configuration of global paths is **8.7 seconds**. Note that the optimisation is only required once a group of robots enters a new zone.

### 3.2.3 Robots and Humans Sharing Working Space

In this experiment, two humans acting in the simulated working space, which contains three robots, are added. The

|                  | DWA Baseline | VO-based DWA | PN Control |
|------------------|--------------|--------------|------------|
| Success rate     | 82%          | 92%          | 97%        |

Table 4 Comparisons of the proposed PN-based control with other path planning methods
human actor plugin of Gazebo simulator is customised to allow control the moving trajectory of a human subject by repeated patterns (Fig. 22). The blob-based detection using the laser scanner developed by the SPENCER project [42] is applied to track the trajectories and velocities of moving human subjects and obstacles in the working space. The same size of the map as earlier experiments of 20x20m is used. The experiments were repeated 5 times. For each trial, the static obstacles, start and goal positions of both robots and humans are randomised. The minimum distances between human and robot are recorded for evaluation purposes. The overall result is summarised in Table 5. With the combination with the dipole field on top of PN path planning, there are no collisions between robots and humans recorded.

### 4 Conclusions and Discussion

This paper has introduced a novel path planning algorithm for multiple robots using PN in combination with obstacle avoidance with dipole field. To avoid the congestion of routing several robots into an intersection area, PN is utilised for path optimisation and then movement control of robots. Particularly, the path planning system establishes alternative routes to the goal for each robot. Due to the effectiveness PN demonstrates in analysing the travelling time from the starting position to the goal point, the optimal paths are assigned to the robots to increase the chances of a robot to reach its goal without facing dead-lock situations inside narrow areas. In the second stage when the global paths are configured with optimal selection, the PN model is applied to control the movement of robots to ensure that a robot is allowed to pass through a cross one by one. The experimental results with Gazebo simulator have revealed that the PN control is able to synchronise the movements of multiple robots passing through the intersection, which helps to both avoid dead-lock and shorten the travelling paths of the robots. Meanwhile, the dipole-field implemented with DWA is able to advance the local path planning with an ability to avoid moving humans and other robots as well as uncontrolled obstacles in the shared workspace, as is also shown in simulations. Thus, these results show that the presented algorithm can improve the dependability aspect of a navigation system, based on path planning, for multiple robots system.

Some limitations of the proposed method are listed to be addressed in the future. First, in the current configuration one token is assigned at the control PN place of a narrow area, which allows only a single robot to pass through. This is to minimise the possibility of a deadlock happening inside the area. If the control area is big enough, it is more optimal to allow several robots to enter it at the same time. Therefore, the correlation between the size of the region and the number of a token at the control place should also be considered as a factor to be optimised with the modelled PN. Second, the proposed system is mainly implemented in a centralised manner. Thus, a shift into a distributed algorithm is considered so that the system is not so dependent on a single central server to operate. Third, recent searches [43] have shown the new way to address multi-agent path planning with uncertainty using Markov decision process and congestion probabilities. Studies about extending this approach into the PN control model presented

| Table 5 Minimum distance (in meters) between each human and robot in every trials |
|---------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
|                                | T1    | T2    | T3    | T4    | T5    | Mean/STD |
|---------------------------------|-------|-------|-------|-------|-------|--------|
| without                         |       |       |       |       |       |        |
| H1-R1                           | 3.38  | 3.52  | 0.63  | 0.86  | 3.48  |         |
| H1-R2                           | 0.74  | 3.42  | 0.88  | 1.41  | 0.81  | 2.01/1.28 |
| H1-R3                           | 1.43  | 0.90  | 3.94  | 4.25  | 0.87  |         |
| H2-R1                           | 3.46  | 5.28  | 0.69  | 1.99  | 2.76  |         |
| H2-R2                           | 6.64  | 2.84  | 0.33  | 1.43  | 3.13  |         |
| H2-R3                           | 7.40  | 5.00  | 5.47  | 3.16  | 3.77  |         |
| with                            |       |       |       |       |       |        |
| H1-R1                           | 1.02  | 3.53  | 0.89  | 1.03  | 3.84  |         |
| H1-R2                           | 2.46  | 3.13  | 1.68  | 1.33  | 1.64  |         |
| H1-R3                           | 1.80  | 1.25  | 3.98  | 3.19  | 1.13  | 2.26/1.12 |
| H2-R1                           | 1.47  | 5.29  | 1.43  | 1.70  | 1.77  |         |
| H2-R2                           | 4.09  | 1.89  | 1.03  | 1.68  | 2.55  |         |
| H2-R3                           | 5.81  | 4.69  | 5.50  | 2.90  | 3.90  |         |

H1: Human 1, H2: Human 2, R1: Robot 1, R2: Robot 2, and R3: Robot 3. The **bold** numbers mark the distance between a robot and a human less than one meter. Without dipole field, there is one collision present in Trial 3 (marked with a **bold and underlined** number). Mean/STD are calculated on the distances less than 5 meters.
in this paper may help to find a faster way to solve the path selection optimisation. Lastly, the evaluation of the proposed approach is currently performed only as a Gazebo simulator. It is plausible to assume that an implementation in real robots will help to evaluate the proposed algorithm.

**Author Contributions** All authors, including Lan Anh Trinh, Mikael Ekström and Baran Cürükli, contributed to the writing of the manuscript. Lan Anh Trinh implemented the algorithms and performed the testing experiments.

**Funding** Open access funding provided by Mälardalen University.

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control, funded by the Swedish Knowledge Foundation.

**Availability of data and materials** The data are available upon request.

**Code Availability** The source codes are available upon request.

**Declarations**

**Conflict of Interests** The authors declare that the research was conducted in the absence of any commercial or financial relationships construed as a potential conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

**References**

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1(269), 269–271 (1959)
2. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4(2), 100–107 (1968)
3. Yang, G.Z., et al.: The grand challenges of science robotics. Science Robotics 3 eaar7650 (2018)
4. Kagermann, H., Lukas, W.D., Wahlster, W.: Industrie 4.0: Mitdem internet der dinge auf dem weg zur 4. industriellen revolution. VDI nachrichten 13(11) (2011)
5. Soulignac, M.: Feasible and optimal path planning in strong current fields. IEEE Trans. Robot. 27(1), 89–98 (2011)
6. Paden, B., Čap, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A Survey of motion planning and control techniques for self-Driving urban vehicles. IEEE Trans. Intell. Veh 1(1), 33–55 (2016)
7. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. J. Robot. Auton. Syst. Elsevier 61, 1258–1276 (2013)
8. Martelli, A.: On the complexity of admissible search algorithms. Artificial intelligence. Elsevier 8, 1–13 (1977)
9. Nash, A., Danial, K., Koenig, S., Felner, A.: Theta*: Any angle path planning on grids. J. Intell. Robot Syst. 39, 533–579 (2000)
10. Kavraki, L.E., vestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic road maps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. 12(4), 566–580 (1996)
11. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Computer Science Department Iowa State University (1998)
12. Yu, J., LaValle, S.M.: Structure and intractability of optimal multi-robot path planning on graphs. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence, pp. 1443–1449 (2013)
13. Sturtevant, N., Holte, R.C., Schaeffer, J.: Enhanced partial expansion A*. J. Artif. Intell. Res. 50, 141–187 (2014)
14. Sharon, G., Stern, R., Felner, A., Sturtevant, N.: Conflict based search for optimal multi-agent pathfinding. Artif Intell 219, 40–66 (2015)
15. Barer, M., Sharon, G., Stern, R., Felner, A.: Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: Proceedings of the Seventh Annual Symposium on Combinatorial Search, pp. 19–27 (2014)
16. Silver, D.: Cooperative pathfinding. In: Proceeding of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 117–122 (2005)
17. Ma, H., Harabor, D., Stuckey, P.J., Li, J., Koenig, S.: Searching with consistent prioritization for multi-agent path finding. In: Proceedings of AAAI19-Thirty-Third AAAI Conference on Artificial Intelligence, pp. 7643–7650 (2019)
18. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. 17(7), 760–772 (1998)
19. Berg, J.P., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocally n-body collision avoidance. Int. J. Robot Res. 70, 3–19 (2011)
20. Daman, B., Berg, J.P.: Generalized reciprocally collision avoidance. Int. J. Robot. Res. 34(12), 1501–1514 (2015)
21. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P.A., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics, vol. 83, pp. 203–216 (2010)
22. Snape, J., Berg, J.P., Guy, S.J., Manocha, D.: The hybrid reciprocal velocity obstacles. IEEE Trans. Robot. 27, 696–706 (2011)
23. Long, P., Liu, W., Pan, J.: Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robotics and Automation Letters 2, 656–663 (2016)
24. Claes, D., Tuyls, K.: Multi robot collision avoidance in a shared workspace. Auton. Robot. 42, 1749–1770 (2018)
25. Filotheou, A., Tsardoulias, E., Dimitriou, A., Symeonidis, A., Petrou, L.: Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2D static environments. Journal of Intelligent and Robotic Systems, 98, 567–601 (2019)
27. Samanta, B., Sarkar, B.: Application of Petri nets for systems modeling and analysis. OPSEARCH (2012)
28. Yasuda, G.: Discrete event behavior-based distributed architecture design for autonomous control of mobile robots with embedded Petri nets. Advances in chaos theory and intelligent control, p. 37. Springer, Berlin (2016)
29. Iocchi, L., Lazaro, M.T., Jeannotte, L., Mouaddib, A., Sahli, H.: COACHES-Cooperative autonomous robots in complex and human populated environments. LNCS. Springer, Berlin (2015)
30. Fabre, J.C., Lauer, M., Rot, M., Amy, M., Excoffon, W., Stoicescu, M.: Towards resilient computing on ROS for embedded applications. In: Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS) (2016)
31. Miyagi, P.E., Riascos, L.A.M.: Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets. Journal of Control Engineering Practice, 14 (2006)
32. Lussier, B., Lampe, A., Chatila, R., Ingrand, F., Killijian, M.O., Powell, D.: Fault tolerant planning: Towards dependable autonomous robots. Research Report, LAAS-CNRS (2015)
33. Mahulea, C., Kloetzer, M.: Robot planning based on boolean specifications using Petri Net models. IEEE Transactions on Automatic Control 63(7) (2018)
34. Fantini, M.P., Mangini, A.M., Pedroncelli, G., Ukovich, W.: A decentralized control strategy for the coordination of AGV systems. Control. Eng. Pract. 70, 86–97 (2018)
35. Roszkowska, E., Reveliotis, S.A.: On the liveness of guidpath-based, zone-controlled dynamically routed, closed traffic systems. IEEE Trans. Autom. Control 53, 1689–1695 (2008)
36. Trinh, L., Ekström, M., Cürikü, B.: Petri Net based navigation planning with dipole field and dynamic window approach for collision avoidance. In: Proceedings of the 6th International Conference on Control, Decision and Information Technologies (CoDIT) (2019)
37. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
38. Pommerau, F.: SNAKES: a flexible high-level Petri nets library. In: Proceedings of PETRI NETS’15, LNCS 9115. Springer, Berlin (2015)
39. Trinh, L., Ekström, M., Cürikü, B.: Toward shared working space of human and robotic agents through dipole flow field for dependable path planning. Frontiers in neurorobotics 12, Article 28 (2018)
40. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine, 23–33 (1997)
41. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: An open-source robot operating system. In: Proceedings of the Open-source Software Workshop (2009)
42. Linder, T., Breuers, S., Leibe, B., Arras, K.O.: On multi-modal people tracking from mobile platform in very crowded and dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 5512–5519 (2016)
43. Street, C., Pütz, S., Mühlig, M., Hawes, N., Lacerda, B.: Congestion-aware policy synthesis for multirobot systems. IEEE Transactions on Robotics, 1–19 (2021)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Lan Anh Trinh has been a PhD student at Mälardalen University since May 2016. She received her B.Eng. degree in Electronics and Telecommunications from Hanoi University of Technology, Vietnam in 2009. She got her M.Sc. degree in Automatic Control from Department of Electronics and Radio Engineering, Kyung Hee University, South Korea in 2012. From 2013 to 2016, she was a lecturer at Electronics Engineering Department, Post and Telecommunications Institute of Technology (PTIT), Hochiminh, Vietnam.

Mikael Ekström is a professor in robotics at Mälardalen University, Västerås, Sweden, where he is the head of the robotics research group. He received his M.Sc. and Ph.D. in physics from Uppsala University, Sweden in 1993 and 1999 respectively. His research interests include robotics, autonomous vehicles, sensors, and communication.

Baran Cürikü is Senior Lecturer in Artificial Intelligence at Mälardalen University, Västerås, Sweden. He received his M.Sc. and PhD in Computer Science from Mälardalen University in 1998 and 2005, respectively. His research interests are in multi-agent systems, human-robot interaction, and autonomous systems.