Path Planning of Multi-Type Robot Systems with Time Windows Based on Timed Colored Petri Nets

Zhou He 1, Ruijie Zhang 2, Ning Ran 3 and Chan Gu 1,*

1 School of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi’an 710021, China; hezhou@sust.edu.cn
2 College of Mechanical and Electrical Engineering, Shaanxi University of Science and Technology, Xi’an 710021, China; 1905020@sust.edu.cn
3 College of Electronic and Information Engineering, Hebei University, Baoding 071002, China; ranning87@hotmail.com
* Correspondence: guchan@sust.edu.cn

Abstract: Mobile robots are extensively used to complete repetitive operations in industrial areas such as intelligent transportation, logistics, and manufacturing systems. This paper addresses the path planning problem of multi-type robot systems with time windows based on timed colored Petri nets. The tasks to be completed are divided into three different types: common, exclusive and collaborative. An analytical approach to plan a group of different types of mobile robots is developed to ensure that some specific robots will visit task regions within given time windows. First, a multi-type robot system and its environment are modeled by a timed colored Petri net. Then, some methods are developed to convert the task requirements that contain logic constraints and time windows into linear constraints. Based on integer linear programming techniques, a planning approach is proposed to minimize the total cost (i.e., total travel distance) of the system. Finally, simulation studies are investigated to show the effectiveness of the developed approach.

Keywords: multi-type robot system; discrete event system; timed colored Petri net; path planning; time window

1. Introduction

With the increase in complexity and diversity of tasks, a single robot may not satisfy the task requirement in some industrial applications, which has led to the creation of multi-robot systems (MRSs) [1]. The main challenge for such systems is to coordinate the control and planning of multiple robots so that tasks are effectively finished. A large number of studies in the literature have been proposed to choose the optimum path trajectories for multiple robots to reach some task regions under certain requirements such as shortest travel distance, time windows, and obstacle avoidance [2–7].

The collaboration of multiple robots was a critical issue that has garnered much attention in recent decades. An improved particle swarm algorithm has been developed to deal with the collaborative navigation of MRSs [8]. To tackle the problem of collaborative coverage of multiple robots in complicated environments, a novel multi-robot persistent coverage strategy was proposed in [9] to take into account the environmental complexity and the differences in robot path lengths. If robots have different priorities and there are both static and dynamic obstacles in the workspace, a navigation strategy with priorities was presented [10]. To complete some collaborative tasks within time constraints, the distributed planning of robots was researched in [11,12] to ensure that enough robots are present at the task destinations within the given time constraints. In [13], an integer linear programming method was proposed for optimizing the path trajectories of warehouse robots so that pickup and delivery points are reached within given time windows. The path planning of MRSs with complex tasks has also drawn a great deal of attention in the
literature [14–19] in recent decades, where the tasks were expressed by Boolean formulae or linear temporal logic.

In an MRS, the system’s environment was divided into zones, and the movement of any robot from one zone to another was treated as a discrete event [20–23]. Petri nets (PNs) are an efficient tool to model, analyze, and control discrete event systems [24–31], such as robotics and intelligent transportation. PN can effectively avoid the state explosion problem as the number of robots increases. In [32], an MRS was modeled by a PN and an integer linear programming problem (ILPP) was provided to find the shortest path on the condition that the types of robots are identical (i.e., only one type). Owing to the restricted sensors and actuators, an optimal PN controller for MRSs was devised [20] to minimize crashes when some events were indistinguishable and uncontrollable. The problem of path planning for MRSs with logic and time constraints was discussed using timed Petri nets [22], and an ILPP was established to determine the optimum paths of robots so that the task zones were visited by a robot within a given time window and the forbidden regions were always avoided.

To finish some complex tasks cooperatively, multiple types of robots with different functions were required [33]. For instance, in many practical systems such as logistical and manufacturing systems, not all the tasks can be completed by all robots, i.e., some tasks may need specific types of robots to complete [34]. Moreover, the time factors of tasks are also crucial in real-world applications. A multi-type robot system (MTRS) is a system that consists of multi-types and multi-quantities of robots. Therefore, path planning and task allocation are the key issues for multi-type robot systems that have received much attention in recent years. It requires to achieve a trade-off between the coordination of the robots to finish the tasks and the total cost of the system. This paper advances this research by developing an optimal path planning algorithm for multi-type robot systems with time windows.

2. Related Works

During the past few years, there have been numerous works focusing on the task allocation and path planning problem of MTRSs. Heuristic algorithms were applied to find near optimal solutions in real time [35,36]. In [37], an improved particle swarm optimization algorithm was used to search for the combination of robots and tasks, and the greedy algorithm was used to sort the task execution order and calculate the cost of the task execution strategy. Taking the resources required for tasks into consideration, a novel coordinated task allocation approach based on cross-entropy was presented in [38]. A comparative study between optimization-based and market-based approaches for the task allocation problem of MTRSs was discussed in [39]. In [40], an optimal task allocation solution was developed based on the mixed integer linear programming techniques. In [33], a high-level PN model called colored Petri net (CPN) [41,42] was used to model the MTRSs, where each token was associated with a color that represents the type of the robot. Then, a mixed integer linear programming problem (MILPP) was proposed to obtain the optimal sequences and paths for operating the tasks. A Petri net toolbox for multi-robot planning under uncertainty was developed in [43] to deal with multi-robot coordination problems and obtain robust, efficient, and predictable strategies.

In this paper, the path planning problem for MTRSs with time windows is discussed by using timed colored Petri nets (TCPNs). Compared with general Petri nets, colored Petri nets allow a lot of folding techniques in order to reduce the complexity of the system model. This paper focuses on a scenario in which a group of different types of mobile robots performs in a known environment that contains several tasks. We assume that there are three types of tasks in the working environment: common (performed by any type of robot), exclusive (performed by only a certain type of robot), and collaborative (requires multi-types of robots to work cooperatively). The problem consists in finding an optimal trajectory for each robot such that all task regions are reached by one or more specified robot types within a given time window and the forbidden regions are avoided.

The main contributions of this paper are as follows:
1. We extend the previous works on the path planning of MTRSs (i.e., multi-type and multi-quantity of robots) by introducing time windows. In addition, the tasks to be finished are divided into three different types: common, exclusive, and collaborative.

2. An analytical approach based on integer linear programming techniques is developed so that both the time windows and the logic requirements imposed on the tasks are satisfied.

3. Simulation results are developed to show the effectiveness of the developed methodology.

The rest of the paper is structured as follows: The basic concepts of TCPNs and the modeling framework are given in Section 3. In Section 4, the path problem of MTRSs with time windows is formally formulated. In Section 5, we develop an MILPP to solve the path planning problem. Simulation studies for the developed approach are given in Section 6. Conclusions and future work are in Section 7.

3. Preliminary

In this section, the basic concepts of the TCPNs used in this paper are introduced. It should be noticed that the definitions of TCPNs used in this paper are slightly modified versions of [41] in order to simplify the definitions. More details can be found in [44–46].

3.1. Colored Petri Nets

**Definition 1.** Let $S$ be a set. A multiset [45] $\alpha$ over $S$ is defined by a mapping $\alpha : S \rightarrow \mathbb{N} = \{0, 1, 2, \ldots\}$ and is represented as

$$\alpha = \sum_{s \in S} \alpha(s) \otimes s$$

where $s \in S$ is an element of set $S$ and $\alpha(s)$ denotes the number of occurrences of the element $s$ in the set $S$.

A Colored Petri Net (CPN) is a quintuple $N = (P, T, Co, Pre, Post)$, where $P$ is a set of $n$ places; $T$ is the a of $m$ transitions; $Co$ is a set of colors; $Co : P \cup T \rightarrow Cl$ is a function that associates a set of colors from $Cl$ to each element in $P \cup T$. For each $p_i \in P$, $Co(p_i) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,n_i}\} \subseteq Cl$ is the set of possible token colors in $p_i$, and $u_i$ denotes the total number of possible token colors in $p_i$. Analogously, for each $t_j \in T$, $Co(t_j) = \{b_{j,1}, b_{j,2}, \ldots, b_{j,n_j}\} \subseteq Cl$ is the set of possible occurrence colors of $t_j$, and $v_j$ denotes the total number of possible occurrence colors of $t_j$.

Matrices $Pre, Post \in \mathbb{N}^{n \times m}$ are the pre- and post-incidence matrix. Each element $Pre(p_i, t_j)$ is mapped from the set of occurrence colors of $t_j$ to a non-negative multiset over the set of colors of $p_i$, i.e., $Pre(p_i, t_j) : Co(t_j) \rightarrow \mathbb{N}(Co(p_i))$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. We use $Pre(p_i, t_j)$ to represent a matrix of $u_i \times v_j$ non-negative integers, where $Pre(p_i, t_j)(h, k)$ denotes the weight of the arc from place $p_i$ with respect to color $a_{i,h}$ to transition $t_j$ with respect to color $b_{j,k}$. The mapping for $Post(p_i, t_j)$ can be analogously defined. The incidence matrix $C$ is an $n \times m$ matrix, where $C(p_i, t_j) : Co(t_j) \rightarrow \mathbb{Z}(Co(p_i))$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, and $C(p_i, t_j) = Post(p_i, t_j) - Pre(p_i, t_j)$.

The marking $M(p_i)$ of place $p_i \in P$ is a non-negative multiset over $Co(p_i)$. The mapping $M(p_i) : Co(p_i) \rightarrow \mathbb{N}$ is associated with each possible token color in $p_i$, which is a non-negative integer representing the number of tokens of that color contained in place $p_i$, and

$$M(p_i) = \sum_{s \in Co(p_i)} M(p_i)(s) \otimes s,$$

where marking $M(p_i)$ is denoted as a column vector of $u_i$ non-negative integers that has an $h$-th component $M(p_i)(h)$ equal to the number of color tokens $a_{i,h}$ contained in place $p_i$.

The marking $M$ is defined as an $n$-dimensional column vector of multisets, where

$$M = [M(p_1), \ldots, M(p_n)]^T.$$

A colored Petri net system $(N, M)$ is a colored Petri net $N$ with a marking $M$. 
A transition \( t_i \in T \) is enabled at a marking \( M \) with respect to color \( b_{i,k} \) if for each \( p_i \in P \) it holds \( M(p_i)(h) \geq \text{Pre}(p_i, t_i)(h, k) \) \( (h = 1, \ldots, u_i) \), and an enabled transition \( t_i \) may fire yielding a new marking \( M' \), denoted by \( M(t_i(k))M' \), where \( M'(p_i)(h) = M(p_i)(h) - \text{Pre}(p_i, t_i)(h, k) + \text{Post}(p_i, t_i)(h, k), \forall p_i \in P \). A firing sequence \( \sigma = t_{i_1}(k_{j_1})t_{j_2}(k_{j_2}) \ldots t_{j_r}(k_{j_r}) \) from \( M \) is a sequence of transitions, each one firing with respect to a given color such that

\[
M(t_{i_1}(k_{j_1}))M_1[t_{j_2}(k_{j_2}), M_2 \ldots t_{j_r}(k_{j_r}))M_r.
\]

Marking \( M' \) is said to be reachable in \( \langle N, M \rangle \) if there exists a firing sequence \( \sigma \) such that \( M[\sigma]M' \), and it holds that

\[
M' = M + C \cdot \sigma,
\]

where \( \sigma = [\sigma(t_1), \ldots, \sigma(t_m)]^T \) is called the firing count vector of \( \sigma \), and \( \sigma(t_j) \in \mathbb{N}(\text{Co}(t_j)) \) \( (j = 1, \ldots, m) \) is a multiset that indicates the firing times of transition \( t_j \) with respect to each of its colors.

### 3.2. Timed Colored Petri Net for the Multi-Type Robot Systems

A timed Colored Petri net (TCPN) is a pair \( \langle N, \theta \rangle \) [44], where:

- \( N = (P, T, \text{Co}, \text{Pre}, \text{Post}) \) is a CPN,
- \( \theta(t_j) \) is a non-negative multiset over \( \text{Co}(t_j) \). The mapping \( \theta(t_j) : \text{Co}(t_j) \rightarrow \mathbb{N} \) associates to each occurrence color of transition \( t_j \) a non-negative integer fixed firing duration. We denote by \( \theta = [\theta(t_1), \ldots, \theta(t_m)] \) the firing delay vector, where \( \theta(t_j) = [\theta(t_j)(v_1), \ldots, \theta(t_j)(v_{|v|})] \).

We denote by \( G = \langle N, \theta, M_0 \rangle \) a TCPN system, where \( M_0 \) is an initial marking.

In the following, it is assumed that the moving environment \( R = \{r_1, r_2, \ldots, r_n\} \) of an MTRS is divided into a set of \( n \) triangular regions that connect to each other via several predefined line segments [47]. Each type of robot can move freely inside any triangular region except the forbidden one. We model the MTRS and its working environment by a TCPN according to Algorithm 1. To illustrate the construction of a TCPN better, a simple MTRS is discussed in the following example.

**Example 1.** Let us consider the MTRS in Figure 1. Its moving environment is divided into ten triangular regions, i.e., \( R = \{r_1, \ldots, r_{10}\} \). Two robots with different functions are initially deployed in regions \( r_5 \) (represented by blue circle \( C_1 \)) and \( r_5 \) (represented by red circle \( C_2 \)), respectively, i.e., \( K = 2 \).

The TCPN system of the MTRS is shown in Figure 2. It consists of 10 places, \( P = \{p_1, \ldots, p_{10}\} \), each of which represents a region, and 26 transitions, \( T = \{t_1, \ldots, t_{26}\} \), each of which represents the movement of a robot from one region to an adjacent one. The description of transitions and places is presented in Table 1.
Algorithm 1: Construct the TCPN model for the multi-type robot system

Require: The working environment with $K$ robots and $n$ regions $R = \{r_1, r_2, \ldots, r_n\}$
Ensure: A TCPN system $G = (N, \theta, M_0)$ and a distance vector $w$
1: Let $P = \{p_1, \ldots, p_k\}$, $T = \emptyset$, $Pre = 0$, $Post = 0$, $\theta = 0$, $M_0 = 0$, $w = 0$, $m = 0$
2: Associate each region $r_i$ with a place $p_i$ in the place set $P$, i.e., each place $p_i \in P$ represents a region $r_i \in R$;
3:  
   for each $p_i \in P$
5:    
      for each $p_j \in P, j \neq i$
6:       if $p_i$ and $p_j$ are adjacent, i.e., region $r_i$ and region $r_j$ are adjacent then
7:        $m = m + 1$;
8:        $T = T \cup \{t_m\}$; %transition $t_m$ represents the movement from $r_i$ to $r_j$
9:       $\theta(t_m)(k)$ equals the average time for a $C_k$ type of robot moving from region $r_j$
10:  to region $r_j, k = 1, \ldots, K$;
11:   $w(t_m)(k)$ equals the average distance for a $C_k$ type of robot moving from region
12:   $r_j$ to region $r_j, k = 1, \ldots, K$;
13: end if
14: end for
15: end for

Figure 2. The TCPN system corresponding to the MTRS in Figure 1.

Table 1. Description of places and transitions in Figure 2.

<table>
<thead>
<tr>
<th>Place</th>
<th>Description</th>
<th>Transition</th>
<th>Description</th>
<th>Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>Region $r_1$</td>
<td>$t_1$</td>
<td>Moving to $r_1$ from $r_3$</td>
<td>$t_{14}$</td>
<td>Moving to $r_8$ from $r_4$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>Region $r_2$</td>
<td>$t_2$</td>
<td>Moving to $r_3$ from $r_1$</td>
<td>$t_{15}$</td>
<td>Moving to $r_5$ from $r_2$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>Region $r_3$</td>
<td>$t_3$</td>
<td>Moving to $r_6$ from $r_1$</td>
<td>$t_{16}$</td>
<td>Moving to $r_2$ from $r_3$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>Region $r_4$</td>
<td>$t_4$</td>
<td>Moving to $r_3$ from $r_6$</td>
<td>$t_{17}$</td>
<td>Moving to $r_4$ from $r_5$</td>
</tr>
<tr>
<td>$p_5$</td>
<td>Region $r_5$</td>
<td>$t_5$</td>
<td>Moving to $r_4$ from $r_6$</td>
<td>$t_{18}$</td>
<td>Moving to $r_5$ from $r_4$</td>
</tr>
<tr>
<td>$p_6$</td>
<td>Region $r_6$</td>
<td>$t_6$</td>
<td>Moving to $r_4$ from $r_8$</td>
<td>$t_{19}$</td>
<td>Moving to $r_{10}$ from $r_7$</td>
</tr>
<tr>
<td>$p_7$</td>
<td>Region $r_7$</td>
<td>$t_7$</td>
<td>Moving to $r_{17}$ from $r_8$</td>
<td>$t_{20}$</td>
<td>Moving to $r_{17}$ from $r_8$</td>
</tr>
<tr>
<td>$p_8$</td>
<td>Region $r_8$</td>
<td>$t_8$</td>
<td>Moving to $r_4$ from $r_7$</td>
<td>$t_{21}$</td>
<td>Moving to $r_9$ from $r_2$</td>
</tr>
<tr>
<td>$p_9$</td>
<td>Region $r_9$</td>
<td>$t_9$</td>
<td>Moving to $r_7$ from $r_3$</td>
<td>$t_{22}$</td>
<td>Moving to $r_2$ from $r_9$</td>
</tr>
<tr>
<td>$p_{10}$</td>
<td>Region $r_{10}$</td>
<td>$t_{10}$</td>
<td>Moving to $r_3$ from $r_2$</td>
<td>$t_{23}$</td>
<td>Moving to $r_{10}$ from $r_9$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>
The possible color sets of each place \( p_i \in P \) \((i = 1, \ldots, 10)\) and each transition \( t_j \in T \) \((j = 1, \ldots, 26)\) are \( \text{Co}(p_i) = \{C_1, C_2\} \) and \( \text{Co}(t_j) = \{C_1, C_2\} \), respectively. The initial marking \( M_0 = [M_0(p_1), \ldots, M_0(p_{10})]^T \) of the TCPN system is a column vector that contains ten subvectors, where \( M_0(p_1) = [0 \ 0 \ 0 \ 1]^T \), \( M_0(p_2) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_3) = [1 \ 0 \ 0 \ 0]^T \), \( M_0(p_4) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_5) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_6) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_7) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_8) = [0 \ 0 \ 0 \ 0]^T \), \( M_0(p_9) = [0 \ 0 \ 0 \ 0]^T \), and \( M_0(p_{10}) = [0 \ 0 \ 0 \ 0]^T \). Each element in subvector \( M_0(p_i) \) denotes the number of two different robots in region \( r_j \). The incidence matrix \( C \) of the TCPN system is a \( 10 \times 26 \) matrix, where \( C(p_i, t_j) \) is a \( 2 \times 2 \) matrix, for \( i = 1, \ldots, 10 \) and \( j = 1, \ldots, 26 \). For instance, the sixth row of \( C \) indicates that the input (resp., output) transitions of place \( p_6 \) are \( t_3 \) and \( t_6 \) (resp., \( t_4 \) and \( t_5 \)).

\[
C = \begin{bmatrix}
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & -\text{Pre}(p_1, t_5) & Post(p_1, t_4) & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & -\text{Pre}(p_2, t_4) & Post(p_2, t_6) & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & Pre(p_2, t_5) & Post(p_2, t_3) & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & Pre(p_3, t_1) & Post(p_3, t_2) & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} \\
\end{bmatrix}
\]

Since every robot can have access to every region, we have

\[
\text{Pre}(p_i, t_j) = \text{Post}(p_i, t_j) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

Assume that the distance and the moving time between every two adjacent regions are equal to one and two, respectively; we have the distance vector \( w = [w(t_1), \ldots, w(t_{26})] \), where \( w(t_j) = [w(t_j)(1), w(t_j)(2)] = [1, 1] \), and the firing delay vector \( \theta = [\theta(t_1), \ldots, \theta(t_{26})] \), where \( \theta(t_j) = [2, 2], j = 1, \ldots, 26 \).

4. Problem Statement

In this paper, we assumed that there were three type of tasks in the working environment for the MTRSs: common (performed by any type of robot), exclusive (performed by
only a certain type of robot), and collaborative (requires multi-types of robots to work co-
operatively). We were interested in a variety of regions of the moving environment \( R^I \subseteq R \),
particularly set \( R^I \), which contains four disjoint subsets \( R^E, R^S, R^U, \) and \( R^A \) such that
\[
R^I = R^E \cup R^S \cup R^U \cup R^A,
\]
where
- \( R^E \): set of exclusive task regions for \( C_k \) type of robot;
- \( R^S \): set of collaborative task regions for \( C_k \) type of robot;
- \( R^U \): set of public task regions that can be visited by any type of robot;
- \( R^A \): set of forbidden regions that all robots should always avoid.

For every interest region \( r \in R^I \), a state function \( \Delta(r) \) is defined as follows:
\[
\Delta : R^I \to \{ E^k, S^k, U, A \},
\]
where
- \( \Delta(r) = E^k \) means that the exclusive task in region \( r \) can only be performed by \( C_k \) type
  of robot;
- \( \Delta(r) = S^k \) means that the task in region \( r \) should be cooperatively performed by \( C_k \)
  type of robot and other types of robots, i.e., two or more robots are required to visit
  region \( r \);
- \( \Delta(r) = U \) means that the task in region \( r \) can be performed by any type of robot;
- \( \Delta(r) = A \) means that region \( r \) should be always avoided.

Furthermore, we assumed that each task region \( r_i \) was associated with a time window
\([e_i, l_i]\) to indicate the earliest and the latest (global) time instant for visiting the task region \( r_i \).
Note that \([0, +\infty] \) was used to indicate that no time constraint was attached to a task region.

Given an MTRS and its moving environment \( R = \{ r_1, r_2, \ldots, r_n \} \) with \( n \) regions, a task
requirement for an MTRS that contains logic requirement and time windows is the sequence.
(To simplify the notation, we ignore the time window \([0, +\infty] \) in \( \varphi \) in the remainder of
the paper for all regions of interest with no time constraints.)
\[
\varphi = (r_1, \Delta(r_1), [e_1, l_1]), (r_2, \Delta(r_2), [e_2, l_2]), \ldots, (r_n, \Delta(r_n), [e_n, l_n])
\]
(2)
where \( r_i \in R^I \) is an interest region; \( \Delta(r_i) \) is a state function defined in Equation (1), and
\([e_i, l_i]\) is a given time window corresponding to region \( r_i \).

Example 2. The following task requirement for the MTRS depicted in Figure 1,
\[
\varphi = (r_1, E^1, [2, 5]), (r_8, S^1), (r_8, S^2), (r_{10}, U, [20, 30]), (r_7, A),
\]
(3)
requires that the task in region \( r_1 \) can only be performed by a \( C_1 \) robot in the time window \([2, 5]\); the
task in region \( r_8 \) should be performed by two types of robot, i.e., \( C_1 \) and \( C_2 \); the task in region \( r_{10} \)
can be performed by any type of robot in time window \([20, 30]\); and all robots should avoid region \( r_7 \).
Note that the task in region \( r_8 \) does not have a time window constraint.

Problem 1. In this paper, we considered an MTRS that contained \( K \) different types of mobile robots
moving in a static surrounding \( R = \{ r_1, r_2, \ldots, r_n \} \). For a given task requirement \( \varphi \) that
contains logic constraints and time windows as defined in Equation (2), the goal of this paper was to find
optimal trajectories for all robots to meet the task requirement, while the total travel distance of the
MTRS was minimized.

5. Path Planning for MTRSs Based on TCPNs

In this section, some methods were presented to convert the logic constraints and time
windows for each type of task into linear constraints. Based on the linear transformations,
an analytical approach was developed so that both the time windows and the logic requirement imposed on the tasks were fulfilled while minimizing the total cost, i.e., total travel distance.

5.1. Preprocessing of the Forbidden Regions

Any forbidden region \( r_q \in \mathbb{R}^A \) should not be visited by robots at any state (i.e., marking). This can be done by removing the places that correspond to the forbidden regions and their input and output arcs from the TCPN model. In Algorithm 2, the place \( p_q \) corresponding to forbidden region \( r_q \) was removed from the TCPN system along with its corresponding row information from \( \text{Pre, Post} \), and \( M_0 \). In addition, all the input and output transitions of \( p_q \) and the column information from \( \text{Pre} \) and \( \text{Post} \) were removed. As a consequence, the forbidden regions were excluded from the planned trajectories. Note that \( t'_i \) and \( t_i \) denote the output places and input places of transition \( t_i \), respectively.

### Algorithm 2: Preprocessing of the forbidden regions

```
Require: A TCPN system \( G = (N, \theta, M_0) \), where \( N = (P, T, Co, \text{Pre, Post}) \);
Ensure: A simplified TCPN system \( G = (N, \theta, M_0) \) that does not contain forbidden regions

1: for each forbidden region \( r_q \in \mathbb{R}^A \) do
2:   remove the row corresponding to the place \( p_q \) from \( \text{Pre, Post} \), and \( M_0 \), respectively, where \( p_q \in P \) is the place that represents region \( r_q \);
3:   \( P := P \setminus \{p_q\} \); % remove place \( p_q \) from the set \( P \)
4:   for each \( t_i \in T \) do
5:     if \( t'_i = p_q \) or \( t_i = p_q \), then
6:       remove the column corresponding to transition \( t_i \) from \( \text{Pre, Post, w, and } \theta \), respectively; % remove the paths related to the forbidden region \( r_q \)
7:       \( T := T \setminus \{t_i\} \); % remove the input transition or output transition of \( p_q \) from the set \( T \)
8:   end if
9: end for
10: end for
```

5.2. Transformation of Logic Constraints and Time Windows for Exclusive Tasks and Collaborative Tasks

In the rest of the paper, we denote the total number of tasks to be completed by \(|\text{task}|\). Each marking \( M \) of the TCPN represents the intermediate positions of all robots. According to Algorithm 1, each task can be represented by a reachable marking of the TCPN. In another word, completing the tasks of the MTRSs corresponds to some specific markings of the TCPN. Therefore, there are at most \(|\text{task}|\) intermediate markings for the TCPN to finish all tasks, i.e.:

\[
M_j = M_{j-1} + C \cdot \sigma_j, \; j = 1, \ldots, |\text{task}|
\]

(4)

where \( \sigma_j = [\sigma_j(t_1), \sigma_j(t_2), \ldots, \sigma_j(t_m)]^T \) is the firing count vector from \( M_{j-1} \) to \( M_j \). In other words, vector \( \sigma_j \) represents all the movements of the MTRS from state \( M_{j-1} \) to state \( M_j \).

The exclusive tasks refer to those that can only be completed by a specific type of robot in exclusive regions. Only a \( C_k \) robot can visit an exclusive task region \( r_q \in \mathbb{R}^E_k \). To fulfill this logic requirement, the following constraint should be satisfied:

\[
\begin{align}
 z_{q,k,j} 
- 1 
& \leq M_j(p_q)(k) \cdot H \\
 1 
- z_{q,k,j} 
& \leq M_j(p_q)(k) \cdot H \\
 \sum_{j=1}^{|\text{task}|} z_{q,k,j} 
& \leq |\text{task}| - 1 \\
 z_{q,k,j} 
& \in \{0,1\} \\
 j 
& = 1, \ldots, |\text{task}|
\end{align}
\]

(5)

where \( H \in \mathbb{R}_{\geq 0} \) is a large enough number.
Proof. If \( z_{q,k,j} = 1 \) holds, then constraint (5a) can be simplified as follows:

\[
0 \leq M_j(p_q)(k) \cdot H
\]  

(6)

Since \( H \) is a large enough number and \( M_j(p_q)(k) \) is a non-negative integer, constraint (6) becomes redundant. On the other hand, when \( z_{q,k,j} = 0 \) holds, constraint (5a) can be simplified as follows:

\[
1 \leq M_j(p_q)(k) \cdot H
\]  

(7)

which is equivalent to \( M_j(p_q)(k) \geq 1 \). This condition indicates that the exclusive task region \( r_q \in R^E_k \) is visited by a \( C_k \) robot at marking \( M_j \). Since \( z_{q,k,j} \in \{0,1\} \) is a binary variable, constraint (5b) guarantees that at least one variable \( z_{q,k,j} (j = 1, \ldots, |\text{task}|) \) is equal to zero. As a consequence, the exclusive task region \( r_q \) will be visited by a \( C_k \) robot at marking \( M_j \) at least once. \( \square \)

For the time window \([e_q, l_q] \) imposed on an exclusive task region \( r_q \in R^E_k \), it can be converted into a linear constraint as follows:

\[
\begin{cases}
  z_{q,k,j} - 1 \leq M_j(p_q)(k) \cdot H \\
  1 - z_{q,k,j} \leq M_j(p_q)(k) \cdot H \\
  \sum_{j=1}^{|\text{task}|} z_{q,k,j} \leq |\text{task}| - 1 \\
  z_{q,k,j} \in \{0,1\} \\
  \tau_{k,j} = \tau_{k,j-1} + \theta \cdot \sigma_j \\
  \tau_{k,0} = 0 \\
  e_q - \tau_{k,j} \leq [1 - M_j(p_q)(k) + z_{q,k,j}] \cdot H \\
  \tau_{k,j} - l_q \leq [1 - M_j(p_q)(k) + z_{q,k,j}] \cdot H \\
  j = 1, \ldots, |\text{task}|
\end{cases}
\]

(8a)

where \( \theta = [\theta(t_1), \ldots, \theta(t_m)] \) denotes the firing delay vector, and \( \sigma_j = [\sigma_j(t_1), \ldots, \sigma_j(t_m)]^T \) denotes the firing count vector of all types of robots from \( M_{j-1} \) to \( M_j \). Parameter \( \tau_{k,j} \in \mathbb{R}_{\geq 0} \) represents the time instant when a \( C_k \) robot arrives at marking \( M_j \).

Proof. According to Equation (5), condition (8a) ensures the logic requirement for the exclusive tasks. If a \( C_k \) robot visits the exclusive task region \( r_q \) at any marking \( M_j \), \( (j = 1, \ldots, |\text{task}|) \), i.e., \( z_{q,k,j} = 0 \) and \( M_j(p_q)(k) = 1 \), the third and fourth constraints of (8b) can be simplified as follows:

\[
\begin{cases}
  e_q - \tau_{k,j} \leq 0 \\
  \tau_{k,j} - l_q \leq 0 \\
\end{cases} \quad \Rightarrow \quad e_q \leq \tau_{k,j} \leq l_q
\]

As a consequence, it ensures that the time instant when a \( C_k \) robot visits the region \( r_q \) at marking \( M_j \) will satisfy the prescribed time window constraint.

On the other hand, if the robot does not visit the exclusive task region \( r_q \) at any marking \( M_j \), \( (j = 1, \ldots, |\text{task}|) \), i.e., \( M_j(p_q)(k) = 0 \), then the third and fourth constraint of (8b) can be simplified as follows:

\[
\begin{cases}
  e_q - \tau_{k,j} \leq [1 + z_{q,k,j}] \cdot H \\
  \tau_{k,j} - l_q \leq [1 + z_{q,k,j}] \cdot H \\
\end{cases}
\]

Since \( H \) is a large enough number and \( z_{q,k,j} \in \{0,1\} \), the above condition becomes redundant for a \( C_k \) robot. \( \square \)
The collaborative tasks refer to those that should be cooperatively completed by different types of robots. For a collaborative task region \( r_q \in R^S_k \), region \( r_q \) should be visited by different types of robots. This can be analogously achieved by imposing constraints (5) and (8) separately to each required type of robot.

5.3. Transformation of Logic Constraints and Time Windows for Public Tasks

Public tasks refer to those that can be completed by any type of robot, i.e., for a public task region \( r_q \in R^U \), region \( r_q \) should be visited by any type of robot. For each public region \( r_q \in R^U \), we defined a vector \( v_{q,k} = [v_{q,k}(p_1), v_{q,k}(p_2), \ldots, v_{q,k}(p_1), \ldots, v_{q,k}(p_n)(u_n)] \), \((k=1, \ldots, K)\) whose elements are zeros except for the \( q \)-th elements that are equal to one, i.e.,

\[
v_{q,k}(p_i)(k) = \begin{cases} 1, & \text{if } i = q \\ 0, & \text{else} \end{cases}
\]

Therefore, the constraint imposed on a public task region \( r_q \in R^U \) can be converted into the following linear constraints:

\[
\begin{align*}
1 - v_{q,k} \cdot \sum_{j=1}^{\text{|task|}} M_j & \leq z_{q,k,j} \cdot H & (a) \\
\sum_{k=1}^{K} \sum_{j=1}^{\text{|task|}} z_{q,k,j} & \leq K \cdot \text{|task|} - 1 & (b) \\
z_{q,k,j} & \in \{0, 1\} \\
k = 1, \ldots, K
\end{align*}
\]

where \( H \in \mathbb{R}_{\geq 0} \) is a large enough number.

**Proof.** Constraints (9b) and (9c) conjointly ensure that at least one variable \( z_{q,k,j} \) is equal to zero. When \( z_{q,k,j} = 0 \) holds for any robot \( C_k \) \((k = 1, \ldots, K)\) at any marking \( M_j \) \((j = 1, \ldots, \text{|task|})\), so the constraint (9a) can be simplified as follows:

\[
1 \leq v_{q,k} \cdot \sum_{j=1}^{\text{|task|}} M_j
\]

which guarantees that the public task region \( r_q \in R^U \) will be visited by a \( C_k \) robot at marking \( M_j \).

On the other hand, when \( z_{q,k,j} = 1 \) holds for any type of robot \( C_k \) \((k = 1, \ldots, K)\) at any marking \( M_j \) \((j = 1, \ldots, \text{|task|})\), i.e., the public task region \( r_q \in R^U \) is not visited by a \( C_k \) robot at marking \( M_j \), constraint (9a) can be simplified as follows:

\[
1 - v_{q,k} \cdot \sum_{j=1}^{\text{|task|}} M_j \leq H
\]

Since \( H \) is a large enough number, this condition becomes redundant for a \( C_k \) robot. \( \Box \)

In the rest of this paper, we use \( o_{q,k,j} = 0 \) (resp. \( o_{q,k,j} = 1 \)) to indicate that the public task region \( r_q \) is visited by a \( C_k \) robot at marking \( M_j \) within (resp. outside) the predefined time window. The time window \([c_q, l_q] \) imposed on a public task region \( r_q \in R^U \) can be converted into a linear constraint as follows.
\[
\begin{align*}
1 - \nu_{q,k} \cdot \sum_{j=1}^{\text{task}} M_j & \leq z_{q,k,j} \cdot H \\
\sum_{k=1}^{\text{task}} \sum_{j=1}^{\text{task}} z_{q,k,j} & \leq K \cdot |\text{task}| - 1 \\
\tau_{q,k,j} & \in \{0,1\} \\
\tau_{k,j} &= \tau_{k,j-1} + \theta \cdot \sigma_j \\
\tau_{k,0} &= 0 \\
z_{q,k,j} &\leq o_{q,k,j} \\
\sum_{k=1}^{\text{task}} \sum_{j=1}^{\text{task}} o_{q,k,j} &\leq K \cdot |\text{task}| - 1 \\
e_q - \tau_{k,j} &\leq o_{q,k,j} \cdot H \\
\tau_{k,j} - l_q &\leq 1 - M_j(p_q)(k) + o_{q,k,j} \cdot H \\
o_{q,k,j} &\in \{0,1\} \\
k &= 1, \ldots, \text{task} \\
j &= 1, \ldots, |\text{task}|
\end{align*}
\]

where \( \theta = [\theta(t_1), \theta(t_2), \ldots, \theta(t_m)] \) and \( \sigma_j = [\sigma_j(t_1), \ldots, \sigma_j(t_m)]^T \). Parameter \( \tau_{k,j} \in \mathbb{R}_{\geq 0} \) represents the time instant when a \( C_k \) robot arrives at marking \( M_j \).

**Proof.** According to Equation (9), condition (10a) ensures the logic requirement for the public tasks. For a \( C_k \) robot that visits the public task region \( r_q \) at any marking \( M_j \) \( j = 1, \ldots, |\text{task}| \), we have \( z_{q,k,j} = 0 \) and \( M_j(p_q)(k) = 1 \). Therefore, the constraint (10c) can be reduced to:

\[
\begin{align*}
0 &\leq o_{q,k,j} \\
\sum_{k=1}^{\text{task}} \sum_{j=1}^{\text{task}} o_{q,k,j} &\leq 1 - K \cdot |\text{task}| - 1 \\
e_q - \tau_{k,j} &\leq o_{q,k,j} \cdot H \\
\tau_{k,j} - l_q &\leq o_{q,k,j} \cdot H \\
o_{q,k,j} &\in \{0,1\}
\end{align*}
\]

Since variable \( o_{q,k,j} \) is binary, we have the following two situations: (1) if \( o_{q,k,j} = 0 \) holds, condition (11) can be reduced to:

\[
\begin{align*}
e_q - \tau_{k,j} &\leq 0 \\
\tau_{k,j} - l_q &\leq 0 
\Rightarrow \quad e_q &\leq \tau_{k,j} \leq l_q
\end{align*}
\]

which consequently ensures that the time instant when the public task region \( r_q \) is visited by a \( C_k \) robot at marking \( M_j \) satisfies the predefined time window \([e_q, l_q]\); (2) if \( o_{q,k,j} = 1 \) holds, condition (11) can be reduced to:

\[
\begin{align*}
e_q - \tau_{k,j} &\leq H \\
\tau_{k,j} - l_q &\leq H
\end{align*}
\]

which becomes redundant. As a consequence, a \( C_k \) robot will visit the public task region \( r_q \) without any specified time constraint. The first and second constraints of (10c) ensure that at least one robot will visit the public task region \( r_q \) within the time window \([e_q, l_q]\).

For a \( C_k \) robot that does not visit the public task region \( r_q \) at any marking \( M_j \) \( j = 1, \ldots, |\text{task}| \), i.e., \( M_j(p_q)(k) = 0 \), the constraint (10c) can be reduced to:

\[
\begin{align*}
e_q - \tau_{k,j} &\leq H \\
\tau_{k,j} - l_q &\leq H
\end{align*}
\]

which is a redundant constraint and imposes no time constraint for a \( C_k \) robot. \( \square \)
5.4. Path Planning of the MTRSs by an MILPP

Combining the linear transformations for the exclusive, collaborative, and the public task regions, i.e., Equations (4), (8), and (10), Problem 1 can be solved by the following MILPP:

$$
\begin{align*}
\text{min } w & \cdot \sum_{j=1}^{\text{|task|}} \sigma_j \\
M_j & = M_{j+1} + C \cdot \sigma_j \\
z_{q,k,j} - 1 & \leq M_j(p_q)(k) \cdot H \\
1 - z_{q,k,j} & \leq M_j(p_q)(k) \cdot H \\
\sum_{j=1}^{\text{|task|}} z_{q,k,j} & \leq \text{|task|} - 1 \\
e_q - \tau_{k,j} & \leq [1 - M_j(p_q)(k) + z_{q,k,j}] \cdot H \\
\tau_{k,j} - \ell_q & \leq [1 - M_j(p_q)(k) + z_{q,k,j}] \cdot H \\
1 - v_{q,k} & \cdot \sum_{j=1}^{\text{|task|}} M_j \leq z_{q,k,j} \cdot H \\
\sum_{k=1}^{\text{K}} \sum_{j=1}^{\text{|task|}} z_{q,k,j} & \leq \text{K} \cdot \text{|task|} - 1 \\
z_{q,k,j} & \leq o_{q,k,j} \\
\sum_{k=1}^{\text{K}} \sum_{j=1}^{\text{|task|}} o_{q,k,j} & \leq \text{K} \cdot \text{|task|} - 1 \\
e_q - \tau_{k,j} & \leq [1 - M_j(p_q)(k) + o_{q,k,j}] \cdot H \\
\tau_{k,j} - \ell_q & \leq [1 - M_j(p_q)(k) + o_{q,k,j}] \cdot H \\
\tau_{k,j} & = \tau_{k,j-1} + \theta \cdot \sigma_j \\
\tau_{k,0} & = 0 \\
z_{q,k,j} & \in \{0,1\} \\
o_{q,k,j} & \in \{0,1\} \\
k & = 1, \ldots, K \\
j & = 1, \ldots, \text{|task|} 
\end{align*}
$$

(15)

Note that $w = [w(t_1), \ldots, w(t_n)]$ is a vector that contains the travel distance between each adjacent region. The objective function of (15) represents the total travel distance for all robots from initial to final position. Condition (15a) represents all the states of the MTRSs from initial to final position. Condition (15b) ensures the logic requirements and time windows for the exclusive and collaborative tasks. Condition (15c) ensures the logic requirements and time windows for the public tasks. The optimal solution $\sigma_j$ $(j = 1, \ldots, \text{|task|})$ of MILPP (15) denotes the firing count vector of the TCPN system that corresponds to the movement strategy of each type of robot. Therefore, the path of each type of robot $k$ that contains a sequence of regions, denoted by $\text{path}_k$, can be iteratively constructed by using Algorithm 3.

The computational complexities of the developed algorithms are as follows. For Algorithms 1–3, the complexities are $O(n^2)$, $O(n \cdot m)$ and $O(|\text{Task}| \cdot m^2)$, respectively, where $n$ denotes the number of regions (or places), $m$ denotes the number of transitions and $|\text{Task}|$ denotes the number of tasks. For MILPP (15), it belongs to the NP-hard class. Some efficient tools such as LINGO and CPLEX can be used to solve the MILPPs.
Algorithm 3: Iterative construction of the paths of each type of robot

Require: TCPN system $G = \langle N, \theta, M_0 \rangle$, $N = (P, T, C_0, Pre, Post)$, incidence matrix $C = Post - Pre$, firing count vector $\sigma_j (j = 1, \ldots, |task|)$

Ensure: Path of each type of robot $path_k$, $k = 1, 2, \ldots, K$

1: Let $M = M_0, j = 0$, $path_k = c_k, k = 1, \ldots, K$
2: while $j \leq |task|$ do
3:   while $1^{T} \cdot \sigma_j \neq 0$ do
4:     for each $t \in T$ do
5:       while $\sigma_j[t](k) > 0$ and $M[t](k) > 0$ do
6:         Pick the corresponding type of robot $C_k$ from the input places $'t$ of $t$;
7:         Assign movement corresponding to $t$ to a $C_k$ robot, i.e., a $C_k$ robot moves from the input places $'t$ of $t$ to the output places $'t$ of $t$ (i.e., from region $r_1$ to region $r_7$);
8:         Let $path_k := path_k \cdot r_1 \cdot r_7$; update the path of $C_k$ type of robot
9:         Let $M := M + C[, t]$; update the current state
10:        $\sigma_j[t](k) := \sigma_j[t](k) - 1$;
11:       $\sigma_j := \sigma_j$; update the firing count vector
12:     end while
13:   end for
14: end while
15: $j := j + 1$;
16: end while

6. Illustrative Example

In this following, the developed approach is illustrated using an example formed with the Robot Motion Tool RMTool [48]. The static environment of the system is partitioned [47] into 74 regions, as shown in Figure 3, i.e., $R = \{r_1, r_2, \ldots, r_{74}\}$. There are three different types of robots (i.e., $|K| = 3$) working in the environment whose initial locations are marked by different colored dots in Figure 3, i.e., $C_1$ robot in region $r_{16}$ (blue dot); $C_2$ robot in region $r_{36}$ (purple dot); and $C_3$ robot in region $r_{49}$ (brown dot).

![Figure 3](image)

The TCPN model of the MTRS is shown in Figure 4. It consists of 74 places and 218 transitions. We assumed that the distance and the moving time between adjacent regions were equal to one, i.e., $w = [w(t_1), w(t_2), \ldots, w(t_{218})] = 1_{1 \times 218}$ and $\theta = [\theta(t_1), \theta(t_2), \ldots, \theta(t_{218})] = 1_{1 \times 218}$. For the MTRS in Figure 3, we assumed that the set of interest regions are:

- $R^E = \{r_{65}, r_{49}\}$, $R^E_2 = \{r_3\}$, $R^E_3 = \{r_{24}, r_{45}\}$
- $R^S = \{r_{10}\}$ $R^S_2 = \{r_{21}\}$, $R^S_3 = \{r_{21}, r_{10}\}$
\( R^U = \{ r_{11}, r_{37}, r_{68} \} \);
\( R^A = \{ r_{13}, r_{48} \} \).

**Figure 4.** The TCPN system corresponding to the MTRS in Figure 3.
For the temporal requirement, we assumed that the task regions $r_{69}$, $r_3$, $r_{24}$, $r_{45}$, $r_{21}$, $r_{10}$, and $r_{11}$ should be visited in the time windows $[10, 12]$, $[3, 5]$, $[10, 15]$, $[17, 25]$, $[8, 22]$, $[3, 8]$, and $[15, 20]$, respectively. Therefore, the task requirement $\phi$ for the MTRS can be represented as follows:

$$\phi = (r_{65}, E^1), (r_{69}, E^1, [10, 12]), (r_3, E^3, [3, 5]), (r_{24}, E^3, [10, 15]), (r_{45}, E^3, [17, 25]), (r_{21}, S^2, [8, 22]), (r_{21}, S^3, [8, 22]), (r_{10}, S^1, [3, 8]), (r_{10}, S^3, [3, 8]), (r_{13}, A), (r_{48}, A), (r_{11}, J, [15, 20]), (r_{37}, J), (r_{68}, J),$$

which requires that task region $r_{65}$ be visited by a $C_1$ robot; task region $r_{69}$ be visited by a $C_1$ robot in the time window $[10, 12]$, task region $r_3$ be visited by a $C_2$ robot in the time window $[3, 5]$, task region $r_{24}$ be visited by a $C_3$ robot in the time window $[10, 15]$; task region $r_{45}$ be visited by a $C_3$ robot in the time window $[17, 25]$; task region $r_{21}$ be visited by $C_2$ and $C_3$ robots in the time window $[8, 22]$; task region $r_{10}$ be visited by $C_1$ and $C_3$ robots in the time window $[3, 8]$; task region $r_{11}$ be visited by any type of robot in the time window $[15, 20]$; tasks regions $r_{37}$ and $r_{68}$ be visited by any type of robot; and all robots should avoid regions $r_{13}$ and $r_{48}$. Note that the task regions $r_{65}$, $r_{37}$, and $r_{68}$ do not have time constraints.

We implemented the proposed approach by MATLAB with toolbox YALMIP (YALMIP solver (2019) [Online]. Available online: yalmip.github.io/allsolvers/, accessed on 1 August 2021.) on a laptop equipped with a 1.8 GHZ Core i5 Processor. The highest computational effort of the proposed approach was spent in solving the MILPP (15). The paths of the robots were obtained by solving Algorithm 3 as follows:

- **path**$_1$ (Blue): $r_{16} \rightarrow r_{71} \rightarrow r_{19} \rightarrow r_{10} \rightarrow r_{66} \rightarrow r_{68} \rightarrow r_{35} \rightarrow r_{65} \rightarrow r_{57} \rightarrow r_{67} \rightarrow r_{69};$
- **path**$_2$ (Purple): $r_{36} \rightarrow r_1 \rightarrow r_5 \rightarrow r_{30} \rightarrow r_3 \rightarrow r_{30} \rightarrow r_5 \rightarrow r_1 \rightarrow r_{37} \rightarrow r_{21};$
- **path**$_3$ (Brown): $r_{49} \rightarrow r_{52} \rightarrow r_{56} \rightarrow r_9 \rightarrow r_{62} \rightarrow r_{61} \rightarrow r_{60} \rightarrow r_{10} \rightarrow r_{19} \rightarrow r_{71} \rightarrow r_{73} \rightarrow r_{72} \rightarrow r_{24} \rightarrow r_{29} \rightarrow r_2 \rightarrow r_{20} \rightarrow r_{39} \rightarrow r_{11} \rightarrow r_6 \rightarrow r_{21} \rightarrow r_{45}.$

The overall CPU time for the developed approach is 23 seconds. In addition, we also gave the visiting time of each task region in Table 2 and the Gantt chart of the obtained paths in Figure 5. As one can see, both the logic requirements and the time windows were satisfied by the proposed approach.

**Table 2.** Visiting time of each task region.

<table>
<thead>
<tr>
<th>Task Regions</th>
<th>Time Window</th>
<th>Visiting Time of Each Task Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{65}$</td>
<td>$[0, +\infty]$</td>
<td>8</td>
</tr>
<tr>
<td>$r_{69}$</td>
<td>$[10, 12]$</td>
<td>11</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$[3, 5]$</td>
<td>4</td>
</tr>
<tr>
<td>$r_{24}$</td>
<td>$[10, 15]$</td>
<td>12</td>
</tr>
<tr>
<td>$r_{45}$</td>
<td>$[17, 25]$</td>
<td>20</td>
</tr>
<tr>
<td>$r_{21}$</td>
<td>$[8, 22]$</td>
<td>9</td>
</tr>
<tr>
<td>$r_{21}$</td>
<td>$[8, 22]$</td>
<td>19</td>
</tr>
<tr>
<td>$r_{10}$</td>
<td>$[3, 8]$</td>
<td>3</td>
</tr>
<tr>
<td>$r_{10}$</td>
<td>$[3, 8]$</td>
<td>7</td>
</tr>
<tr>
<td>$r_{11}$</td>
<td>$[15, 20]$</td>
<td>17</td>
</tr>
<tr>
<td>$r_{37}$</td>
<td>$[0, +\infty]$</td>
<td>8</td>
</tr>
<tr>
<td>$r_{68}$</td>
<td>$[0, +\infty]$</td>
<td>6</td>
</tr>
</tbody>
</table>

Total travel distance 40
In order to better test the efficiency of the developed approach, we used Robot Motion Toolbox [48] to randomly generate some examples with different numbers of robots and regions. In Table 3, the number of regions, the number of robots, the number of task regions and the CPU time for each tested case are presented. Case 1 is the MTRS in Figure 3. As one can observe, the CPU time grows significantly as the number of regions and robots increases, since the number of constraints and variables of MILPP (15) increase as the the number of regions and robots increases. However, our developed approach can always provide an optimal solution.

Table 3. Simulation results for different cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>Nb. of Regions</th>
<th>Nb. of Task Regions</th>
<th>Nb. of Robots</th>
<th>CPU Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>74</td>
<td>12</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>Case 2</td>
<td>144</td>
<td>16</td>
<td>5</td>
<td>243</td>
</tr>
<tr>
<td>Case 3</td>
<td>180</td>
<td>23</td>
<td>8</td>
<td>5408</td>
</tr>
<tr>
<td>Case 4</td>
<td>230</td>
<td>31</td>
<td>14</td>
<td>84256</td>
</tr>
</tbody>
</table>

7. Conclusions

This paper addressed the path planning of multi-type robot systems with time windows using TCPNs. To deal with complex tasks that need specific types of robots to complete, the tasks of the systems were classified into three different types, i.e., common, exclusive, and collaborative. Combining the state equations of TCPNs, some approaches to convert the logic constraints and time windows imposed on the different types of tasks into a set of linear constraints was developed. Then, an efficient path planning method was
proposed so that regions containing different types of tasks were visited by a specified type of robot in a predefined time window. Simulation studies illustrated the effectiveness of this approach. TCPN can effectively avoid the state explosion problem as the number of robots increases. The developed approach can be used in application areas where tasks may need specific types of robots to complete. It should be noticed that the collision avoidance problem is not addressed in this paper. Our future research aims to study the path planning of MTRSs with high-level specifications such as Boolean specifications or linear temporal logic, which can be used for patrolling path planning and path planning with multi-task selections.

**Author Contributions:** Conceptualization, Z.H.; Data curation, R.Z.; Formal analysis, Z.H.; Funding acquisition, C.G.; Methodology, C.G.; Project administration, N.R.; Software, R.Z.; Supervision, Z.H.; Validation, N.R.; Visualization, N.R.; Writing—original draft, R.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China under Grant Nos. 61803246, 61903119, and 62003201, the China Postdoctoral Science Foundation under Grant No. 2019M663608, the Young Talent fund of University Association for Science and Technology in Shaanxi China under Grant No. 20210117, the Natural Science Foundation of Hebei Province under Grant A2020201021, the Hebei Province Foundation for Returned Overseas Chinese Scholars under Grant C20190319, and the Foundation of Hebei Education Department under Grant BJ2021008.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**


24. Ma, Z.; He, Z.; Li, Z.; Guia, A. Design of supervisors for linear marking specifications in labeled petri nets. *Automatica* 2022, 136, 110031. [CrossRef]

25. He, Z.; Ma, Z.; Tang, W. Performance safety enforcement in strongly connected timed event graphs. *Automatica* 2021, 128, 109605. [CrossRef]


27. He, Z.; Ma, Z.; Li, Z.; Guia, A. Parametric transformation of timed weighted marked graphs: Applications in optimal resource allocation. *IEEECAA J. Autom. Sin.* 2020, 8, 179–188. [CrossRef]


30. Rocha, J.; Dias, P.; Gomes, L. Improving synchronous dataflow analysis supported by petri net mappings. *Electronics* 2018, 7, 448. [CrossRef]


38. Huang, L.; Qu, H.; Zuo, L. Multi-type UAVs cooperative task allocation under resource constraints. *IEEE Access* 2018, 6, 17841–17850. [CrossRef]


