Abstract—The use of multiple autonomous robots can significantly reduce the time to finish a given mission. In this paper, a totally distributed coordination algorithm is developed to provide robustness to multirobot missions, in the context of the exploration of unknown environment. The analysis and design of the coordination algorithm require formal modeling methodologies. We propose a stochastic Petri net (SPN) based method to analyze and design efficient distributed coordination algorithms. By running the Petri net model, optimized parameters are obtained to improve the efficiency of the multirobot team. Experimental results have validated the proposed method.

I. INTRODUCTION

A. Motivation

Exploration of unknown environment is an important topic in mobile robot research due to its wide real-world applications, such as the map creation for large area, search and rescue, fast locating hazardous material, and some military actions, etc. Exploration missions require that a mobile robot have the capability of localization and mapping. Recently, multirobot exploration has received much attention from the research community [1]. The obvious advantage of multirobot exploration is the concurrency, which can greatly reduce the time needed for the mission. Coordination among multiple robots is necessary to achieve certain degree of exploration efficiency and robustness [2]. The exploration efficiency can be represented by the overall battery power consumed, the total time spent, etc. The robustness implies that the failure of any robot should not cause the whole team to fail. The problem of robust and efficient multirobot exploration is stated as follows:

n identical robots are to explore an unknown area which is cluttered with obstacles. Each robot has sensing, localization, mapping and communication capability. Design a robust exploration algorithm so that the exploration can be completed with less battery power consumption and less time.

Usually most of the battery power of a mobile robot is consumed in the movement of a robot during the exploration process, therefore, minimizing the total power consumption requires the reduction of the total distance travelled by all the robots in the exploration mission.

Multirobot coordinated localization, mapping and exploration is a challenging topic [3]. Some research efforts have already been dedicated in this area and it is attracting more and more interest recently.

B. Previous work in multirobot exploration

Latimer et al. [4] introduced an algorithm to cover an unknown space with a homogeneous team of circular robots. The multirobot coverage algorithm is based on their previous work in single robot coverage. The goal is to guarantee the full coverage of the whole area while achieving certain efficiency by minimizing the repeated coverage. However, their algorithm can not be directly extended to exploration applications where each robot has a relatively large sensor range compared to its physical size and the environment is cluttered with obstacles. Also their algorithm is semi-distributed, which implies that a single robot failure will make the mission impossible. Yamachi [5] developed a totally distributed, asynchronous multirobot exploration algorithm which introduces the concept of frontier cells, or the free cell between explored and unexplored area. The basic idea is to let each robot move to the closest frontier cell independently. This brings the fault tolerance capability. However the algorithm does not achieve sufficient coordination so that multiple robots may end in moving to the same frontier cell, which introduces inefficiency. Based on the similar frontier concept, Simmons et. al [6] developed a semi-distributed multirobot exploration algorithm which requires a central agent to evaluate the bidding from all the other robots to obtain the most information gain while reducing the travel distance. Fujimura and Singh [7] presented a cooperative terrain acquisition strategy for distributed mobile agents, which can handle heterogeneous robots. However, efficiency is not their main concern. Recently, Zlot et. al. [2] developed a multirobot exploration strategy based on a market economy, which inherits the basic concept from the bidding protocol used in Simmons et. al.’s work. Their algorithm can improve the reliability, robustness and efficiency of the exploration through negotiation in a market architecture. However, the goal generation algorithm they use is not as efficient as the frontier-based algorithm and
the negotiation process is complicated.

To summarize, although some work has been done in multirobot exploration, a simple, totally distributed and efficient exploration algorithm has not been reported yet.

C. Previous work in Petri net modeling of distributed coordination

Petri net is a powerful tool to model, analyze, design and validate distributed, sequential and/or concurrent systems [8]. Within the robotics and automation area, Petri nets have been mainly applied to the study of flexible manufacturing systems (FMS) [8] where the major concerns are the modeling, performance analysis and scheduling. However, for the analysis and design of the coordination among multiple robots, Petri nets have not achieved the same success. In the Artificial Intelligence community, Petri nets have been used to model the conversation between autonomous agents. These work include [9]–[12]. But in these work, Petri nets are usually used as a tool to describe the conversation between agents, usually at a high level. Therefore the resulted Petri nets can not help much on the analyzing and designing of distributed coordination algorithms. Recently, some researchers are applying Petri nets to model the interaction among multiple autonomous robots. Rongier et al. proposed a stochastic Petri net based method to analyze and predict the behavior of one class of foraging robots [13]. They studied the robot collective behavior using Markov Chains analysis based on the stochastic model. However, they did not explicitly address the coordination among multiple robots.

To summarize, despite the success of Petri nets in Flexible Manufacturing Systems, the use of Petri net model in modeling, analyzing and designing coordination algorithm for multirobot teams has not been fully investigated.

In this research, we aim to map the developed distributed coordination algorithm into a stochastic Petri net. Due to the closeness between the algorithm and the stochastic Petri net model, we are able to design the critical parameters in the coordination algorithm using Petri net simulation. The remainder of the paper is organized as follows. Section 2 introduces the distributed exploration algorithm. Section 3 describes the Petri net model of this algorithm. Parameter design based on this Petri net model is discussed in Section 4. Section 5 reports the experimental results and Section 6 concludes the work.

II. THE DISTRIBUTED EXPLORATION ALGORITHM

A. Environment and robot model

In this section, we define the environment model and the mobile robot model. The environment is modelled as a 2D occupancy grid of a specified resolution. There are arbitrary-shape stationary obstacles in the area. During the exploration, each cell of the grid has one of the three status: occupied, free or unknown. Here “occupied” means that the cell is occupied by an obstacle such as a wall; “free” means that no obstacle exists in this cell and “unknown” means that the cell has not been detected by the range sensor of any of the robots yet. The physical shape of a robot can be modelled as a square. In this paper, for simplicity, its dimension is set to be the size of a grid cell.

Each robot \( R_i \) has localization, mapping and communication capabilities. It can sense the neighboring cells using range sensors such as a sonar or a laser-ranger-finder. Its sensing range is denoted by a circle of radius \( r \) centered on the robot. The robot moves fast when it is travelling and it stops for sensing and map building. When the robot is travelling, it does not carry out sense and mapping. Each robot is capable of localizing itself with regard to its own local map. The communication capability enables the robot to talk to any other robot with very small time delay. Each robot also builds up a global map by combining the local maps sent by other robots. A typical scenario of multirobot exploration is shown in Figure 1.

B. The detailed algorithm

To achieve reliability and robustness, the exploration algorithm should be totally distributed, which means that there should be no central agents and that the exploration algorithm on every robot should be identical. We assume that all the robots start from initial positions close to each other and the relative positions are known to all the robots. At any moment, the robot team seeks to explore the area with the maximum information gain while reducing the travelling distance from the current position to the target position. Instead of using a central-agent-based semi-distributed model for the bidding process [2], [6], we develop a totally distributed bidding algorithm. Our bidding algorithm is also much simpler than the market
Each robot works asynchronously and makes its own decision to find the next target frontier cell. At any time instant, a robot can be in one of the following three states: 1) sensing and mapping, 2) bidding 3) travelling. Usually the sensing and mapping takes the longest time among the three states. Once a robot finishes sensing and mapping, it broadcasts its newly obtained local map to all the other robots so all the robots are able to update their global map in their memory. If the map is not complete, which means there still exist frontier cells, the robot then decides the next target frontier cell to move to.

```c
void robot_exploration()
{
    mode=SENSING-MAPPING;
    local_map=sensing_mapping();
    broadcast(local_map, robots);
    while (!local_map_complete(map))
    {
        mode=BIDDING;
        winner_declared=false;
        bid_finsihed=false;
        while (bid_finsihed)
        {
            my_bid=calculate_bid(map, robots);
            broadcast(my_bid, robots);
            start_timer(bid_time);
            timer_expired=false;
            while (!timer_expired) 
                if (better_bid_received()) & & !winner_declared
                {
                    if (new_bidder_enter())
                        break;
                }
                if (timer_expired & & !winner_declared)
                {
                    bid_finsihed=true;
                    broadcast(my_id, robots); // I am the winner!
                    modes=TRAVELLING;
                    move_to(my_bid_destination);
                    local_map=sensing_mapping();
                    update_globalmap(map, local_map);
                    broadcast(local_map, robots);
                }
        }
    }
}
```

Fig. 2. The skeleton of the exploration algorithm for each robot.

The robot calculates the information gain \( I_i \) for each frontier cell based on (i) the current global map (ii) the current positions of the sensing-and-mapping robots and (iii) the target cell positions of those travelling robots. The information gain \( I_i \) for a specific frontier cell \( i \) is the number of unknown cells within the sensor range, but not within the sensor range of any other sensing-mapping robots or the target cells of those travelling robots. This information gain definition keeps different robots separated so that their sensing areas will not overlap too much. The robot calculates the travel distance \( D_i \) to each frontier cell. Due to the existence of obstacles and unknown areas, the shortest distance may not be the straight line distance. Dijkstra’s shortest distance algorithm [14] is adopted for the calculation of shortest distance between any two cells on the map. The net gain \( g_i \) for frontier cell \( i \) is the weighted combination of the two measures:

\[
g_i = \omega_1 I_i - \omega_2 D_i
\]

Here \( \omega_1, \omega_2 \in (1, 2) \) is the weight for each component measure. By changing the weights, the importance of each component can be varied. The bid \( B \) of a bidding robot is the maximum net gain of all the frontier cells.

\[
B = \max_i g_i
\]

The bidding robot broadcasts its bid \( B \) to all the other robots and waits for a constant bidding time \( t_{bid} \) to hear responses from other robots. If during that bidding time, no other robot participates in the bidding, or no other robot provides better bids, this robot wins the bidding. If during that time, one or several other robots finish sensing and mapping and send in the new local maps, this bidding robot recalculates its bid based on the new global map and broadcasts the bid again. If this robot receives a better bid from other robot, it will wait for the winner to broadcast its ID. Once the winner is identified and its assigned target is known, this robot recalculates the bid and the process repeats until it wins the bid. The winning robot then moves to its bid target frontier cell and starts the sensing and mapping-bidding-travel cycle again. All the robots stop when no new frontier cell is available. The overall skeleton of the exploration algorithm is shown in Figure 2. It is worth noting that some of the shared variables are not explicitly modified in this skeleton algorithm, actually, they are automatically updated when certain events happen.

Obviously the bidding time \( t_{bid} \) is a crucial parameter. If \( t_{bid} \) is very small, it is highly possible that only one robot will participate in the bidding and be the winner. If \( t_{bid} \) is very big, it is very likely that many robots will participate in the bidding, which implies better bids will be generated but the waiting time will be longer. In the following section, Petri net is used to find the best bidding time so that a tradeoff can be achieved between the bid quality and the total exploration time.

III. STOCHASTIC PETRI NET MODEL FOR THE COORDINATION ALGORITHM

By carefully choosing the robot states and actions, as well as associating random and/or constant time delays to certain actions, we can map the above coordination algorithm into a stochastic Petri net.

A. Petri subnet for a single robot

Stochastic Petri nets can be used to model the distributed coordination algorithm. A stochastic Petri net [8] is a 6-tuple \((P, T, I, O, \omega, \Lambda)\) where \(P\) is a finite set of places, \(T\) is a finite set of transitions, \(I\) is an input function that defines the set of directed arcs from \(P\) to \(T\), \(O\) is an output function that defines the set of directed arcs from \(T\) to \(P\). \(\omega\) represents an initial marking, \(\Lambda : T \rightarrow \mathbf{R}^+\) is
Fig. 3. The stochastic Petri subnet model for each robot.

a mapping from all transitions to random and/or constant delay times.

The Petri net model for the exploration algorithm in each robot is shown in Figure 3. The main loop of the exploration algorithm consists of 7 status: sense and map, calculate-bid, bid-ready, wait, win-bidding, travel and arrive target. The time delays in transition \( T_1 \) and \( T_7 \) are exponentially distributed random numbers, which simulate the random time spent on sensing-mapping and on travel. A deterministic time delay \( t_{\text{bid}} \) is associated with transition \( T_4 \) to represent the bidding time. A one unit deterministic time delay is associated with transition \( T_3 \) to allow the safe cleaning of residual tokens in place Start-Timer and Time-Expired.

The stochastic Petri net model for the interaction among multiple robots is shown in Figure 4. Here, a comparison Petri subnet is used to generate the comparison results among multiple bids. It is worth noting that in Figure 4 the comparison Petri subnet is shown as a central subnet separated from each robot’s Petri subnet. Actually, this comparison is carried out in each robot’s own exploration algorithm implementation, which reflects the distributed nature of the coordination algorithm.

A typical scenario of transition firing for one robot is as follows: the initial token stays at place Sense-Map, representing that the robot is carrying out sensing and mapping task; after a random time, transition \( T_1 \) fires, which changes the robot status to Calculate-Bid and notifies other robots about the arrival of this new bidding robot; transition \( T_2 \) immediately fires which changes the robot status into Bid-Ready and sends the bid to the comparison subnet; the result from the comparison subnet feeds back to the robot subnet, indicating if this robot loses the bid; if yes, a token will be put in place Better-Bid-Received; after one unit time delay, the robot goes into the Wait status and the timer is activated to count the bidding time; there are two situations that can cause the robot to recalculate its bid: (1) any new bidder arrives or (2) it loses the current bidding and the winner is declared; if at the moment the bidding time is up and no better bid is received, transition \( T_5 \) fires and this robot wins the bidding; it then broadcasts the Winner-Declared message to other robots; this winning robot then travels to its target cell with a random travelling time, represented by transition \( T_7 \); after arriving target, the robot repeats the sensing-mapping, bidding and travelling process again.

B. The bid comparison Petri subnet

One critical part in the distributed coordination algorithm is the bid comparison, which tests if \( b_1 \leq b_2 \) for any two bids \( b_1 \) and \( b_2 \). Noticing that it is the comparison results, not the actual bid values that drive the evolution of the coordination algorithm, or the firing orders of the stochastic Petri net, here we develop a novel comparison
Petri subnet which simulates the comparison between two random bids. The comparison Petri subnet is shown in Figure 5. This Petri subnet utilizes the random firing feature between transition $T_0$ and $T_1$ to simulate the comparison results. Once both tokens are in place, either of the two transitions may fire. If $T_0$ fires, we define it to represent $b_1 \leq b_2$. If $T_1$ fires, we have $b_2 \leq b_1$. By including the equal case in both results we can restrict the comparison outcome to be either lose or win. In fact, when equal bids are received, one of them is randomly selected as the winner. This comparison Petri subnet closely simulates the comparison results among random bids.

Fig. 5. The comparison Petri subnet for two bids. Randomly, either $T_0$ or $T_1$ will fire, which simulates the comparison between two random bids.

To simulate the comparison among $n$ bids, we have to compare a total of $C^2_n$ different pairs. However, the comparison results are not independent. There exists a transitivity constraint. That is: if $b \leq a$ and $a \leq c$, then we have $b \leq c$. So if we have one comparison Petri subnet for $a$ and $b$, one for $a$ and $c$, then the comparison subnet for $b$ and $c$ should be constrained by the comparison results from the first two comparison subnets. Inhibitor arcs are used to satisfy this constraint. The comparison subnet for the three bids $a$, $b$, and $c$ is shown in Figure 6. There is a slight deterministic delay added to the transition $T_5$ and $T_6$, which keeps them from firing before the results from the first two comparisons are obtained. For example, if there are $b \leq a$ and $a \leq c$, then $T_7$ fires, which inhibits the firing of $T_6$ so that $c \leq b$ will not appear.

C. Token-clearing Petri subnet

To make the overall stochastic Petri net work, some auxiliary subnets are needed to clear the residual tokens in some places. For example, in Figure 4, when Transition $T_9$ or $T_{10}$ fires, the residual tokens in place Start-Timer or Time-Expired should be cleared before the next bidding session starts. For the comparison network, similar token clearing actions should be taken so that the bids from the previous bidding session are removed before the current bidding session starts. Figure 7 displays the subnet which clears the token in $P_0$ when a control token is generated in $P_1$. This control token can be generated by the firing of desired transitions. For example, in the case of clearing residual tokens in place Start-Timer and Time-Expired, the control tokens can be generated by the firing of transition $T_2$, which directly results from the firing of either $T_1$, $T_9$, or $T_{10}$. In Figure 7 the inhibitor arc and $T_1$ are used so that even if there is no residual token in $P_0$, the Clear Control token will be removed.

Fig. 6. The comparison Petri subnet for three bids.

IV. DESIGN USING PETRI NET MODEL

Once the stochastic Petri net model is built for the interaction among multiple robots, the distributed coordination algorithm can be analyzed through the Petri net simulation. Furthermore, certain parameters in the coordination algorithm can be designed using this model so that performances of the multirobot coordination can be improved.

The major performance of concern is the cost of the exploration, which can be defined as the weighted sum of the total travelling distance and the overall exploration.
time. On the stochastic Petri net model, a new exploration cost can be defined to evaluate the efficiency of the algorithm. It is our observation that the exploration cost is closely related to the bidding time $t_{bid}$. For each bidding session, the possibility of finding a better bid (or a bid with higher $I/C$ ratio) is roughly proportional to the number of bidders in that session. It is obvious that the longer the bidding time is, the more bidders will enter in the session. However, longer bidding time increases the ineffective time in exploration, which slows down the exploration process. Therefore, a compromise should be made between the number of bidders in each session and the overall exploration time. The stochastic Petri net can be a good tool to design the bidding time $t_{bid}$ so that a minimum exploration cost can be obtained.

Supposing the stochastic Petri net runs for $k$ consecutive bidding sessions, the numbers of bidding robots in each bidding session are $n_i$ ($i = 1, 2, \ldots, k$), denoting the total time for the $k$ bidding sessions as $t_{total}$, the exploration cost in the stochastic Petri net for the $k$ consecutive bidding sessions can be represented by

$$\Omega_P(k) = \omega_1 Q(k) + \omega_2 t_{total}(k),$$

where $Q(k) = \sum_{i=1}^{k} 1/n_i$ is called the overall bidding number index and $\omega_1, \omega_2$ are positive weights.

By running the stochastic Petri net with different $t_{bid}$, we can get different $\Omega_P$. The $t_{bid}$ which results in the smallest $\Omega_P$ is selected as the optimal $t_{bid}$.

V. EXPERIMENTAL RESULTS

A. Implementation and results

The simulation of the stochastic Petri net is carried out using the Petri net simulation software $HP Sim$ [15]. The following parameters are set for the experimental testing: Sensing-Mapping delay is exponentially distributed and the average delay is 30 milliseconds. The travelling delay is also exponentially distributed and the average delay is 20 milliseconds. Here one millisecond is chosen as the time unit for measurement purpose.

The first step is to verify the distributed coordination algorithm. The test results show that the Petri net model is live and deadlock-free. The second step is to design the optimal parameter $t_{bid}$. For any bidding time $t_{bid}$ we record the correspondent bidder numbers $n_i$ in each bidding session for 18 consecutive sessions and the total time $t_{total}(18)$. The weights for $\Omega_P$ are selected as: $\omega_1 = 45.0, \omega_2 = 1.0$. So we have

$$\Omega_P(18) = 45.0 \times \sum_{i=1}^{18} 1/n_i + 1.0 \times t_{total}(18)$$

generated. However, the total time for the 18 sessions also increases, which indicates that more time is spent on the bidding.

The graphical representations of the relationship between bidding time $t_{bid}$ and the overall bidding number index, total exploration time, exploration cost are shown in Figure 8, Figure 9 and Figure 10 respectively. From the above table and figures, it is obvious that when $t_{bid} = 15 ms$, the overall exploration cost in the stochastic Petri net is minimum.

### Table I

<table>
<thead>
<tr>
<th>$t_{bid}$ (ms)</th>
<th>No. of Bidders in 18 Sessions</th>
<th>$Q(18) = \sum_{i=1}^{18} 1/n_i$</th>
<th>$t_{total}(18)$ (ms)</th>
<th>$\Omega_P(18)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1</td>
<td>18</td>
<td>308</td>
<td>1118</td>
</tr>
<tr>
<td>5</td>
<td>1;2;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1</td>
<td>16.3</td>
<td>345</td>
<td>1079</td>
</tr>
<tr>
<td>10</td>
<td>2;2;1;2;1;1;1;1;1;1;1;1;1;1;1;1;1;1</td>
<td>14.5</td>
<td>410</td>
<td>1063</td>
</tr>
<tr>
<td>15</td>
<td>3;2;1;3;2;1;2;1;1;3;2;1;2;1;1;1;1;1;1;1;1;1;1;1</td>
<td>13</td>
<td>472</td>
<td>1057</td>
</tr>
<tr>
<td>20</td>
<td>1;2;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1</td>
<td>12.5</td>
<td>530</td>
<td>1093</td>
</tr>
<tr>
<td>25</td>
<td>3;2;1;3;2;1;2;1;1;3;2;1;2;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1</td>
<td>10.5</td>
<td>684</td>
<td>1157</td>
</tr>
<tr>
<td>30</td>
<td>3;2;2;3;2;1;2;3;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2</td>
<td>9</td>
<td>878</td>
<td>1283</td>
</tr>
<tr>
<td>35</td>
<td>3;2;2;3;2;1;2;3;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2</td>
<td>7.8</td>
<td>954</td>
<td>1306</td>
</tr>
</tbody>
</table>

Fig. 8. The overall bidding number index vs. bidding time.
The exploration cost–a weighted combination of the travel distance and total time spent on the mission. This bidding algorithm is simpler and more robust compared to existing multirobot exploration algorithms. A stochastic Petri net is derived to model the coordination algorithm and used as a design tool to obtain optimized parameters in the coordination algorithm.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. Henryk Anschuetz for the free use of HPSim software.

REFERENCES