Abstract—The paper considers the task allocation problem in the case where there is a small number of agents initialized at a single point. The objective is to achieve an even distribution of agents to tasks. To address this problem, this paper proposes a new method that endows agents with models of motivation and leadership to aid their coordination. The proposed approach uses the Particle Swarm Optimization algorithm with a ring neighborhood topology as a foundation and incorporates computational models of motivation to achieve the goals of task allocation more effectively. Simulation results show that, first, the proposed method increases the number of tasks discovered. Secondly, the number of tasks to which the agents are allocated increases. Thirdly, the agents distribute themselves more evenly among the tasks.

I. INTRODUCTION

The use of multiple agents to solve task allocation problems has recently received increasing attention [1], [2]. Given $M$ agents and $N$ tasks with unknown locations, a task allocation problem is defined as a problem of locating (discovering) tasks and distributing agents to the tasks so that a common goal of the system can be achieved. The problem of task allocation arises in various real-world situations, such as allocating agents to search for targets in hazardous areas [3], [4] and finding and rescuing victims inside a destroyed building [5]. One significant problem that arises is achieving an even distribution of agents to tasks, while permitting some agents to continue searching for new tasks [6]. In some cases, the problem is even harder when only a small number of agents are available to cover a large area. This is because a small number of agents may only cover a small part of the search space, which could possibly lead to fewer tasks being discovered. Furthermore, it is often assumed that agents are randomly positioned over the search space [7], [8]. However such an assumption requires one to pre-compute the starting configuration and then assign the agents to their starting positions. It is more practical to assume that the agents are initially located at a single departure point (for example, the point where they are unloaded from a truck.)

This paper considers the task allocation problem in the case where there is a small number of agents initialized at a single point. The objective is to achieve an even distribution of agents to tasks. To address this problem, the paper proposes a new method that endows agents with models of motivation and leadership to aid their coordination. The proposed approach uses the Particle Swarm Optimization (PSO) methodology using a ring neighborhood topology [9], [10] as a foundation and incorporates computational models of motivation to achieve the goals of task allocation more effectively.

There are three main contributions of this paper. The first contribution is to construct a heterogeneous multi-agent system, in which the agents are endowed with different motive profiles and can potentially exhibit leadership to aid coordination. In this paper, we consider one such a system where the agents are endowed with one of two motive profiles named affiliation and power motive profiles [11], [12]. Affiliation-motivated agents are characterized by a higher probability of avoiding high-risk tasks. Power-motivated agents, on the other hand, tend to pursue more risky goals with higher incentive. By using motive profiles that have been validated against the results of human experiments, we expect that the method will influence different risk-taking behaviors in task selection by agents and improve team coordination in solving a task allocation problem.

The second contribution is the modification of the PSO algorithm using a ring topology to include motive profiles. PSO is a well-established approach to solve optimization problems and it has been effectively implemented in multi-agent robotic search applications [7], [13]. Within the context of the task allocation problem, agents are thought of as particles, while the tasks are associated with the maxima in an optimization problem. To measure how close the position of an agent is to the task, the agents use a fitness function $f$ which is assumed to be given. In this paper, this allows us to consider the task allocation problem as a problem of maximizing a fitness function using a swarm of particles. At each time step, the velocity and position of each agent in the proposed method are updated using a modified PSO rule that differs in the way...
the neighborhood best position is computed. While standard PSO algorithms using a ring topology are typically built to locate all global or local optima, the proposed PSO-based task allocation algorithm aims to allow agents to also consider more even distribution of agents to tasks by endowing the agents with tendency to pursue one of the potential neighborhood best positions. In order to choose which neighborhood best position to approach, we introduce the notion of an incentive function to value potential neighborhood best positions. The incentive function is the third key contribution of this paper. We define the incentive function as a nonlinear function that is sensitive to risk-taking behavior. This function evaluates the potential neighborhood best position perceived by each agent using its distance to the potential neighborhood best position and the number of agents around this position.

The results of our numerical experiments indicate that using the proposed method, the number of discovered tasks increases. The number of tasks that the agents discover and allocate themselves to also increases. Moreover, the motivated agents distribute themselves more evenly among the tasks than ordinary PSO agents.

This paper is organized in six sections. Section II provides an overview of the existing work in task allocation, motivation, and leadership as well as a brief review of the PSO algorithms in task allocation problems. Section III describes the MPSO algorithm proposed in this paper, including the introduction of its incentive function. Section IV presents our experimental hypotheses and the metrics used to evaluate our results. Section V describes the experimental setup and results of simulations. The paper concludes in Section VI with a summary of the results and what we intend to develop in the future.

II. BACKGROUND AND RELATED WORK

A. Task Allocation, Motivation, and Leadership

Several methods to solve task allocation problems have been inspired by economic systems [2], [14] and swarm intelligence [1], [6]. Within market-based approaches inspired by economic systems, the objective of the team of agents is to execute the task while minimizing the overall cost such as distance between the agents and the task [14].

Related work on multi-task allocation in multi robot systems has also been presented in the swarm intelligence field [6]. In this approach, the PSO algorithm was modified by integrating a task utility function that considers task weight, travel cost and also agent redundancy to the tasks. Other work using PSO methods has been proposed to control a group of agents to find the brightest spot of light in a room [13].

The proposed method in this paper is more closely related to the second category. We use the PSO methodology as a basic foundation and introduce decision-making mechanism that incorporate computational models of motivation in the agents to improve team coordination.

Recently, three models of motivation have been introduced that influence different risk-taking behavior depending on the obtained incentives: (1) Achievement motivation which is characterized by a preference for goals of intermediate difficulty; (2) Affiliation motivation that is characterized by a preference for avoiding conflict by minimizing risks; and (3) Power motivation that is characterized by a need to be influential by taking extreme risks and selecting higher incentive goals [11], [12]. According to Merrick and Shafi [11], there are six basic variables needed to construct computational models of motivation for achievement, affiliation, and power-motivated agents: turning points of approach ($M^+$) and turning points of avoidance ($M^-$) of a goal, gradients of approach ($\rho^+$) and gradients of avoidance ($\rho^-$) of a goal, relative motivation strength ($S$), and an incentive value for success ($I_S$). Since these computational models can be influential in goal selection, we adopt this approach to aid agents’ coordination in selecting tasks. Although existing research [11] has shown that computational models of motivation can be useful in single-shot goal selection, iterative computation of the incentive function has not been considered. In this paper, we address this issue by introducing a new incentive function that can be applied in a task allocation domain.

The application of the PSO methodology to task allocation problems is based on the intelligence of a swarm where agents are allocated to tasks without the help of leaders [8]. The idea to mimic models of human motivations that influence different behaviors such as leadership, however, is a new approach that we intend to address. A typology of computational models of leadership can be found in [15]. The existing computational models of leadership, however, have not yet considered the role of motivation in leadership to support team coordination in a task allocation domain. The proposed method therefore represents a solution to the task allocation problem within the scope of meso-leadership.

B. Particle Swarm Optimization Algorithms in Task Allocation problems

Originally, the PSO methodology has been introduced as one promising strategy to solve unimodal optimization problems [16]. In order to find multiple solutions (tasks), a number of approaches to modify the original PSO algorithm have been proposed. Examples include the NichePSO method [17] and the Species-based PSO technique [18]. Most of these niching algorithms, however, require specification of several niching parameters and have high computational complexity since global communication is used [9]. To overcome those problems, the Ibest PSO algorithm using a ring neighborhood topology [9] has been shown to exhibit stable niching behavior which eliminates the need to tune any niching parameters.

We now outline a generic scheme for application of the PSO methodology to solving a task allocation problem adopted in [3]. In this scheme, agents are regarded as particles, which update their positions in accordance to the PSO rules to approach the location of a task in the search space. Consider the swarm i.e., an ordered set of agents $S = \{s^1, s^2, ..., s^k, ..., s^M\}$ where $M$ is the total number of agents. Each agent has a position at time $t$, $x^k_t$ and a velocity, $v^k_t$, that determines the direction for the agent to move. To update their positions, the agents use a fitness function whose extrema correspond to the
value of task visibility and task utility functions defined in [3]. In accordance with the above scenario, at each time step, each agent $s_k$ updates its velocity and position as follows:

$$
\begin{align*}
ux_{t+1} &= \chi (ux_t + \varphi_1 r_1 (yx_t - x_k^t) + \varphi_2 r_2 (gy_t^k - x_k^t)) \quad (1) \\
x_{t+1} &= x_k^t + \nu_{t+1} \\
vy_{t+1} &= \nu_{t+1} + \nu_{t+1} (2)
\end{align*}
$$

Here, $vx_0^k$ can be initialized to a random value and $x_k^t$ is a random variable sampled from a uniform distribution in the range of the search space; $\chi$ is a constriction coefficient used to prevent explosion of the agent’s velocity suggested in [19] and calculated according to $\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4}|}$ with $\phi = \varphi_1 + \varphi_2 = 4.1$; $r_1$ and $r_2$ are random values in the range [0, 1], sampled from a uniform distribution. In [3], the general form of Equations (1),(2) is presented using the inertia model and computational models of motivation to achieve the goals of task allocation more effectively.

### A. The Proposed Algorithm

This section introduces the proposed MPSO algorithm. Let $x_k^t$ and $v_k^t$ be the position and velocity of agent $k$ at time $t$. To update their positions, the agents use a fitness function whose extrema correspond to the tasks. The construction of such a function is problem specific and is outside the scope of this paper. One possible scenario assumed in this paper is to equip the tasks with transmitters to radiate signals, and equip each agent with a sensor to detect that signal. The agents then can use the strength of the received signal to determine their distance to the task. In potential real world scenarios, the tasks can be associated with radio base stations or humans, while the signal can be a radio transmission, infrared radiation of a human body, or the sound of survivors calling for help. To increase the chance of finding the task, the agent uses the signal strength to guide towards the location of the strongest source.

In this paper, we consider MPSO algorithm over a directed ring, where each agent, $s_k$, has one immediate agent, $s_{k+1}$, as its neighbor of [9]. The neighborhood of agent $s_k$ is therefore the set $S_k = \{s_k, s_{k+1}\}$ for $1 \leq k < M$, and that of agent $s_M$ is $S_M = \{s_1, s_M\}$.

In the proposed method, at each time step, each agent $s_k$ updates its velocity and position according to Equations (1), (2). To control global exploration, the bound on the agent velocity is imposed, $v_{max}$. We define the personal best position in the same way as the lbest PSO algorithm of [9]. The personal best position, $y_k^t$, for each agent is selected based on the highest fitness value it has found so far. At each time step, the personal best position is hence updated as follows:

$$
\begin{align*}
y_{t+1}^k &= \begin{cases} 
    y_t^k & \text{if } f(x_{t+1}^k) \leq f(y_t^k) \\
x_{t+1}^k & \text{if } f(x_{t+1}^k) > f(y_t^k)
\end{cases} \quad (3)
\end{align*}
$$

where $y_0^k = x_0^k$. Accordingly, associated with each neighborhood $S_k$ is the set of personal best positions $N_k = \{y_k^k : s_k \in S_k\}$.

While the MPSO algorithm utilizes the same update Equations (1)-(3) of [9], it differs in the way the neighborhood best position is calculated. In [9], the neighborhood best position $g_t^k$ is defined as follows:

$$
g_i^k = \arg \max_{y^k_i \in N_k} (f(y^k_i)) \quad (4)
$$

In our approach, Equation (4) is replaced by the function (6), as described below. The main objective of this modification is to enhance the role of motivation in selecting which neighborhood best position to follow, and to provide a mechanism to identify potential neighborhood best positions.
the maximization in Equation (6) yields a unique maximum point over \( G^k_i \), this point becomes agent’s \( s^k \) neighborhood best position. In the special case, when \( T_{res}(I^k_i(y^k_l)) = T_{res}(I^k_i(y^k_{l+1})) \), then the agent \( s^k \) is directed to use the closest neighborhood best position:

\[
g^k_i = \arg \min_{y^k_l \in G^k_i} \{ |x^k_i - y^k_l| \} \quad (7)
\]

The in-depth discussion of the incentive function and the motivation function will be presented in the next two subsections.

1) The Incentive Function: The incentive of the position \( y^k_l \in G^k_i \) is defined as a function of the distance between the current position \( x^k \), and the point \( y^k_l \), \( d^k_t(y^k_l) \), and the number of agents within a circle of a given radius \( h \) centered at \( y^k_l \), \( a_t(y^k_l) \). Furthermore, let \( d^k_t, \max \) be the maximum distance between agent \( s^k \) and any element of the set \( G^k_i \).

\[
d^k_t, \max = \begin{cases} d^k_t(y^k_l) & \text{if } G^k_i = \{ y^k_l \} \\ d^k_t(y^k_{l+1}) & \text{if } G^k_i = \{ y^k_{l+1} \} \\ \max \{ d^k_t(y^k_l), d^k_t(y^k_{l+1}) \} & \text{otherwise} \end{cases} \quad (8)
\]

Using notation (8), define the incentive function as is follows:

\[
I^k_i(y^k_l) = c_1(a_t(y^k_l)) + c_2(a_t(y^k_l))e^{-\left( \frac{d^k_t, \max - d^k_t(y^k_l)}{d^k_t, \max} \right)} \left( \frac{M - a_t(y^k_l)}{M} \right),
\]

with \( 0 \leq c_1(a_t(y^k_l)) + c_2(a_t(y^k_l)) \leq 1 \) (9)

where \( \frac{d^k_t, \max - d^k_t(y^k_l)}{d^k_t, \max} \) is set to zero if \( d^k_t, \max = 0 \). Since the incentive function requires calculating the Euclidian distances between the personal best position, the current position, and all other agents, this adds computational cost compared to the standard PSO algorithm using a ring topology. The exponential function in the incentive function is used to make the incentive function more sensitive to large distance changes. On the other hand, incentive values in this paper decrease linearly with the increase of the number of agents around the position \( y^k_l \) for \( a_t(y^k_l) > 1 \), and \( a_t(y^k_l) \leq 1 \) with a prominent nonlinear transition between these regions. The exponential increase of incentive as the distance increases is to modify agent’s incentive in accordance with risk sensitivity principles [20].

In the case where all agents are concentrated within a small region in the search space, to increase the chance of discovery of new tasks, incentive needs to be given to the agents to explore areas outside the current region. This is achieved by augmenting the set \( G^k_i \) to include an additional randomly generated position point in the search space, and assign \( d^k_t(y^k_l)/d^k_t, \max \) a sufficiently high value \( D \in (0, 1] \), and zero number of surrounding agents. This in turn should increase incentive for exploration and thus should potentially lead to better task discovery.

To adjust the shape of the incentive function, \( c_1(.) \), \( c_2(.) \) functions are used. The idea behind introducing these functions is to provide the means to control the incentive of a position \( y^k_l \) depending on the number of agents around this position. These
control parameters are found to influence the convergence rate of the algorithm. The parameters of the incentive function were chosen in this paper to be:

\[
c_1(a_t(y^r_i)) = \begin{cases} 
0.74 & \text{if } a_t(y^r_i) \leq 1 \\
0.38 & \text{if } a_t(y^r_i) > 1 
\end{cases} 
\]

\[
c_2(a_t(y^r_i)) = \begin{cases} 
0.13 & \text{if } a_t(y^r_i) \leq 1 \\
0.38 & \text{if } a_t(y^r_i) > 1 
\end{cases} 
\]

The incentive parameters chosen above have been experimentally found to provide an acceptable balance between exploration and convergence in the proposed MPSO algorithm. The graph of the incentive function for a fixed \( y^r_i \) is shown in Fig. 1. For each position \( y^r_i \), it represents a family of \( M \) curves parameterized by \( a_t(y^r_i) \). The two cases in Equation (10) are to define higher incentives for the position \( y^r_i \) having one \( (a_t(y^r_i) = 1) \) or no \( (a_t(y^r_i) = 0) \) agents in their vicinity, compared to more populated positions \( y^r_i \), for which \( a_t(y^r_i) > 1 \). In an ideal situation where the position \( y^r_i \) correspond to a discovered task, this prevents agents from overpopulating the task. Thus our incentive function should encourage a more even allocation of agents to the tasks. In addition, this leads to division of labor, where some agents allocate to tasks and others explore to find new tasks. To understand how a motive profile relates to the incentive function, a discussion of the motive profile and the motivation function will be covered in the next subsection.

2) The Motivation Function: In order to endow agents with individual perception of incentives, this paper uses models of motivation [11]. The motivation function is constructed by combining three individual motivation models for affiliation, achievement, and power as follows:

\[
T_{res}(I^k_t(y_i)) = T_{ach}(I^k_t(y_i)) + T_{aff}(I^k_t(y_i)) + T_{pow}(I^k_t(y_i))
\]

\[
T_{res}(I^k_t(y_i)) = \frac{S_{ach}}{1 + e^{\rho_{ach}(M_{ach} - (1-I^k_t(y_i)))}} + \frac{S_{aff}}{1 + e^{\rho_{aff}(I^k_t(y_i) - M_{aff})}} + \frac{S_{pow}}{1 + e^{\rho_{pow}(M_{pow} - I^k_t(y_i))}}
\]

where \( T_{ach}(I^k_t(y_i)) \), \( T_{aff}(I^k_t(y_i)) \), and \( T_{pow}(I^k_t(y_i)) \) are the motivational tendencies for achievement, affiliation, and power respectively. \( M_{ach} \), \( M_{aff} \), \( M_{pow} \), \( \rho_{ach} \), \( \rho_{aff} \), and \( \rho_{pow} \) are the parameters needed to construct computational models of motivation and \( \theta = ach, aff, pow \) represent index of achievement, affiliation, and power-motivated agents respectively [11].

To represent agents with different motives, the agents are initialized with different parameter values as listed in Table I. Using (11), affiliation and power motive profiles can be constructed as shown in Fig. 2. Since affiliation-motivated agents are characterized by a tendency to avoid conflict, these agents may select tasks with lower risks (low incentives). Power-motivated agents, on the contrary, take more extreme risks to be influential in the population. In this paper, achievement-motivated agents, were not used since their participation did not seem to critically affect success in the experiments.

IV. EXPERIMENTS

A. Hypotheses

To compare the performance of the task allocation method proposed in this paper, we will use the lbest PSO algorithm using a ring topology based on Equation (1)-(4) as a benchmark algorithm. This study began with six hypotheses H1-
H6. The first two hypotheses are: If a model of self-motivated leadership is embedded in agents, the agents will exhibit leadership according to the following definitions:

(H1) Leaders visit more tasks
(H2) Leaders spend more time exploring the search space and less time attaching themselves to the task.

If some agents exhibit leadership:
(H3) the MPSO algorithm will discover more tasks than the lbest PSO algorithm using a ring topology
(H4) the MPSO algorithm will allocate agents to more tasks
(H5) the MPSO algorithm will have a more even distribution of agents to tasks
(H6) the MPSO algorithm will discover more tasks and allocate agents to more tasks from one starting point.

To measure the performance of the MPSO algorithm, the next subsection will provide performance metrics such as the number of discovered tasks, task allocation, dwell time, and distribution deviation.

B. Metrics

This section is divided into two subsections. The first subsection characterizes the leadership metrics and the second subsection characterizes the performance metrics. These metrics are not properties of agents, but calculated by the simulation environment for analysis purposes.

1) Leadership Metrics:

Visited tasks: Suppose task \( l \) is located at position \( x^*_{l} \). Agent \( s^k \) is said to be visiting this task if the distance between its current position, \( x^k \), and the task at \( x^*_{l} \) is less than or equal to a certain threshold, \( \varepsilon \) (we will use \( \varepsilon = 0.3 \) in the simulation). More formally, an agent \( s^k \) is visiting the task \( x^*_{l} \) at time \( t \) if the following condition is satisfied:

\[
|x^k_t - x^*_{l}| \leq \varepsilon \tag{12}
\]

Dwell time: For agent \( s^k \), a time interval \([t_1, t_2]\) is a dwell interval of duration \( t_2 - t_1 \) if the condition in (12) is fulfilled for every \( t \) within that interval, and does not hold for \( t \) within \([t_1 - 1, t_2 + 1]\). An agent can possibly visit the same task more than once, and may also visit more than one task. Let \( \tau^{V,k}_{V} \) be the dwell time of agent \( s^k \) in task \( l \) during its \( V \)th visit at this task, \( \hat{L}^{k,l} \) be the total number of visits to task \( l \) by agent \( s^k \). The average dwell time of agent \( s^k \) per task can be calculated as follows:

\[
\tau^k = \frac{\sum_{V=1}^{N} \sum_{l=1}^{\hat{L}^{k,l}} \tau^{V,k}_{V}}{\sum_{l=1}^{N} \hat{L}^{k,l}} \tag{13}
\]

In the case where an agent never visit a task, we specify that the average dwell time for this agent is undefined.

2) Performance Metrics:

Definition of task allocation: An agent is allocated to task \( l \) if the condition in (12) is fulfilled for at least a certain number of consecutive time steps, i.e., it dwells at the task time for at least a given number of time steps. In this paper, we define that an agent is allocated to a task if it dwells for at least 200 time steps.

Number of discovered tasks: An agent is considered as having discovered a task if it has visited a task and no agent has ever visited the task before. Let \( p_l \) denote a binary vector indicating whether a task has been discovered at time step \( t \) as follows:

\[
p_0^l = 0; \quad p_t^l = [p_1^l, p_2^l, ..., p_l^1, ..., p_l^N] \tag{14}
\]

where \( p_0 = 0 \). A task \( l \) which has never been visited by any agent at time step \( t \) is represented as \( p_l^t = 0 \). By using (12) and (14), \( p_l^t \) is updated as follows:

\[
\bigvee_{k=1}^{M} (|x^k_t - x^l_t| \leq \varepsilon) \lor p_{l-1}^t, \quad \text{for} \ t \geq 1 \tag{15}
\]

where \( \bigvee \) denotes logical "or" operator. The total number of discovered tasks (collectively) can be calculated by adding all the elements of matrix \( p_l \) in (14) as follows:

\[
\hat{P} = \sum_{l=1}^{N} p_l^T \tag{16}
\]

where \( T \) is the maximum number of time steps in the simulation.

Distribution deviation of the agents: The distribution deviation of the agents is calculated by using the concept of standard deviation as follows:

\[
\sigma = \sqrt{\frac{1}{N} \sum_{l=1}^{N} (m^l - \bar{m})^2} \tag{17}
\]

where \( m^l \) is the number of agents allocated to task \( l \) and \( \bar{m} \) is the average number of agents allocated to any task. A low distribution deviation indicates that the number of agents allocated to any task tend to be close to the average number of agents allocated to task. Thus, a low distribution shows that the agents are more evenly distributed among the tasks. In the case where there is no agent allocated to a task (not even a single task discovered), we specify that the distribution deviation of the agents is undefined.

V. EXPERIMENTAL SETUP

The proposed algorithm is implemented in Matlab version 7.8.0 (R2009a). The Vincent function [21] was used as a testing problem as it has many global maxima, no local maxima and its height in each position can be thought of as representing the strength of a signal from a task. In the range of \( 1.5 < x_1 < 7 \) and \( 1.5 < x_2 < 7 \), this function has 9 global maxima, which are regarded as tasks. The number of agents used in each experiment is denoted as numbers inside brackets which represent the number of homogenous agents used for the lbest PSO algorithm using a ring topology method.
and the number of affiliation and power-motivated agents used for MPSO method. That is, Ring (100) which means 100 agents were used in PSO algorithm using a ring topology and MPSO (80+20) that means 80 affiliation and 20 power-motivated agents were used in MPSO algorithm. In this paper, a proportion of MPSO (9+6) for a small number of agents and MPSO (80+20) for a large number of agents were chosen for the experiments. This is because it has been observed that having more affiliation-motivated agents provides higher allocation and an adequate number of power motivated agents facilitates sufficient exploration to search for new tasks. All the parameters and values used in the experiments are listed in Table I. The experiments were repeated 50 times with different random seeds for Experiments (1)-(2) and 50 different single point initializations for Experiments (3)-(4). The maximum number of time-steps is set to 500.

A. Results and Discussion

1) Leadership Results: The results from 50 experiments indicate that power-motivated agents visit more tasks compared to affiliation-motivated agents. The 95% confidence interval for the average number of tasks visited by each agent using MPSO (9+6) is 2.165-2.506 tasks for affiliation-motivated agents and 5.454-5.899 tasks for power-motivated agents. Power-motivated agents also show higher number of tasks visited when using MPSO (80+20), which is 7.040-7.192 tasks and 2.989-3.124 tasks for affiliation-motivated agents. Fig. 3 shows that affiliation-motivated agents tend to have more varied and longer average dwell times, whereas the power-motivated agents have very short average dwell times which falls within the range of 1-50 time-steps. By endowing the agents with different motive profiles, we observed that some leadership behavior exhibits from those with power motive profiles consistent with hypotheses (H1) and (H2). By visiting more tasks and having short average dwell times, these agents can potentially increase the number of tasks discovered and reduce the number of agents around tasks.

2) Performance Results:

Random initialization using small (Experiment 1) and large numbers of agents (Experiment 2): We tested hypotheses (H3) and (H4) for 15 and 100 agents. The result of the experiment displayed in Fig. 4 shows how the number of discovered tasks changed over time using 15 agents. In early time steps, the number of discovered tasks increases sharply for both the lbest PSO algorithm with a ring topology and MPSO algorithm. As the power-motivated agents kept exploring, the number of discovered tasks for MPSO algorithm then rose more rapidly than the standard lbest PSO algorithm. The result of the experiments for a small number of agents indicate that MPSO algorithm also increases the number of tasks to which the agents are allocated and decreases the distribution deviation of agents in each task as shown in Table II.

When a larger number of agents was used, both the lbest PSO algorithm and the proposed algorithm show an increase in the number of tasks to which the agents were allocated. This is because the agents cover a larger part of the search space, and thus it is easier for the agents to find more tasks. MPSO algorithm, however, allocate agents to more tasks compared to the lbest PSO algorithm and show more even distribution when (H5) was tested.

Single point initialization using small (Experiment 3) and large numbers of agents (Experiment 4): In order to test hypothesis (H6), we also consider the case where the agents were generated from one starting point. From the experiments, we observed that the MPSO algorithm shows a higher number for both discovered tasks and the number of tasks to which the agents are allocated as shown in Table II. As the lbest PSO algorithm works by selecting the best personal best position in the neighborhood, then if all agents are initialized from one similar starting point, the agents will tend to be concentrated within a small region in the search space. Thus, it leads to fewer tasks being discovered. However, as the MPSO algorithm will consider the number of agents around the task, if all agents are initialized from one starting point, the
ability to start from a single point. lbest modifications endow the initialized point and encourage the other agents to follow.

With leadership behavior will be attracted to move away from those with power motive profiles. The results of the numerical experiments show that compared to the PSO algorithm using a ring topology, the proposed method has certain advantages as follows:

- MPSO discovers more tasks and allocates agents to more tasks.
- When initializing the agents from one starting point, MPSO discovers more tasks and allocates agents to more tasks.
- MPSO results in a more even distribution of agents to tasks.

In the future research, some adjustment to the incentive parameters and other possible variations of the number of each type of agent will also be performed to increase both the role of leader and the performance of the new proposed method by using an optimization strategy. Other potential work that can be done in the future is to increase the number of tasks to which agents are allocated by developing evolutionary approaches that permit agents’ motivations to autonomously adapt to a dynamic environment. We also note that there are other methods to determine agent neighborhoods, for example, the Nearest Neighbors methods [22] where the selection of neighborhoods is done dynamically, based on the distance between agents in the search space. These methods are a potential direction for the future research.

VI. CONCLUSION AND FUTURE WORK

In this paper, an approach named Motivated Particle Swarm Optimization (MPSO) algorithm has been presented that embeds the agents with a model of motivation and leadership for coordination. By endowing the agents with different motive profiles, some aspects of leadership behavior are observed from those with power motive profiles. The results of the numerical experiments show that compared to the lbest PSO algorithm using a ring topology, the proposed method has certain advantages as follows:

- MPSO discovers more tasks and allocates agents to more tasks.
- When initializing the agents from one starting point, MPSO discovers more tasks and allocates agents to more tasks.
- MPSO results in a more even distribution of agents to tasks.

In the future research, some adjustment to the incentive parameters and other possible variations of the number of each type of agent will also be performed to increase both the role of leader and the performance of the new proposed method by using an optimization strategy. Other potential work that can be done in the future is to increase the number of tasks to which agents are allocated by developing evolutionary approaches that permit agents’ motivations to autonomously adapt to a dynamic environment. We also note that there are other methods to determine agent neighborhoods, for example, the Nearest Neighbors methods [22] where the selection of neighborhoods is done dynamically, based on the distance between agents in the search space. These methods are a potential direction for the future research.

REFERENCES