Designing and Predicting the Performance of Agent-based Models for Solving Best-of-N

Puneet Jain *Dept. of Computer Science*
Brigham Young University
Provo, USA
puneetj@byu.edu

Michael A. Goodrich *Dept. of Computer Science*
Brigham Young University
Provo, USA
mike@cs.byu.edu

***Abstract*— Biological inspiration from honeybees, insects, and other animals has been used to create interesting implementations of multi-robot swarms. When the robots in a swarm are completely distributed, that is they lack any form of centralized control, the swarm acts as an agent-based model (ABM) wherein each agent implements its own controller and collective behavior emerges from the interactions between agents. Differential equation and graph-based models of some types of swarms have been used to guarantee collective behavior, but guaranteeing or predicting outcomes for *hub-based agent colonies* with finite numbers of robots remains an open problem. This paper presents a case study of designing an agent-based, hub-based swarm that solves the best-of-N problem with predictable success rates and completion times. The key innovation is modifying a tripartite graph formulation (TGF) from previous work so that it acts as a *graph schema* which abstracts an ABM into a simplified four state model, which in turn leads to a large discrete time Markov chain (DTMC) that describes how the collective state evolves over time. The DTMC can be used to compute success rates and completion times, which act as predictions for the ABM. Deviations between observed ABM outcomes and DTMC predictions lead to modifications in the ABM so that the swarm becomes more predictable.**

Best-of-N, Markov Chains, Agent-based Models, Bio-inspired Models

I. Introduction

Suppose that a multi-robot system is implemented as a distributed agent-based model (ABM), where each robot uses its own controller and where collective behavior emerges from the interactions between robots. This paper addresses how to model and tune an ABM so that the emergent behavior is predictable and satisfies performance guarantees. The paper uses a case study of the best-of-N problem in which robots based at a hub must explore an environment and find the highest quality site from N possible sites; see Fig. 1.

Agent states in the ABM must be sophisticated enough that they can be deployed on actual robots that move through a real environment, and this sophistication makes it difficult to predict or guarantee the behavior of the swarm. This paper creates a *graph schema* that abstracts the complexities from the ABM into four representative states seen in previous work [1], [2]. This four-state graph schema allows us to construct a graph-based representation of individual agents that leads to a tripartite graph formulation (TGF) of the entire swarm state [3]. Transitions between the possible configurations of the TGF encode the evolution of the collective swarm state, which can be represented by a discrete time Markov chain (DTMC). We can then exploit the properties of DTMCs to predict and provide performance guarantees for the ABM.

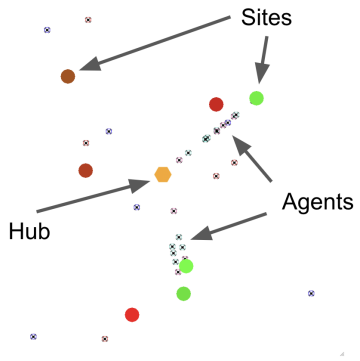

Fig. 1. Best-of-N: green (red) indicate high-quality (low-quality) sites.

The DTMC and ABM can then be used together to improve design. The transition probabilities in the DTMC must be tuned so that they reflect realistic emergent behavior for real-world robots. The "tuned" DTMC can then be used to predict the performance of the ABM under various conditions in the world (e.g., locations and qualities of the sites). Then, large differences or biases between predictions from the DTMC and observed behaviors in the ABM, can guide the robot designer to fine-tune the agent controllers so that emergent behavior is better and more predictable.

The contribution of this paper is a case study that demonstrates (a) how a DTMC can be designed given an ABM designed for the real world using a four-state graph schema, (b) how the DTMC can be tuned so that it captures the key properties of the ABM, (c) how performance can be computed using Monte Carlo simulations of the DTMC, and (d) how differences between DTMC performance and observed ABM behavior can be used to tune the ABM.

The work was supported by the US Office of Naval Research under grant N00014-21-1-2190. The work does not represent opinions of the sponsor.

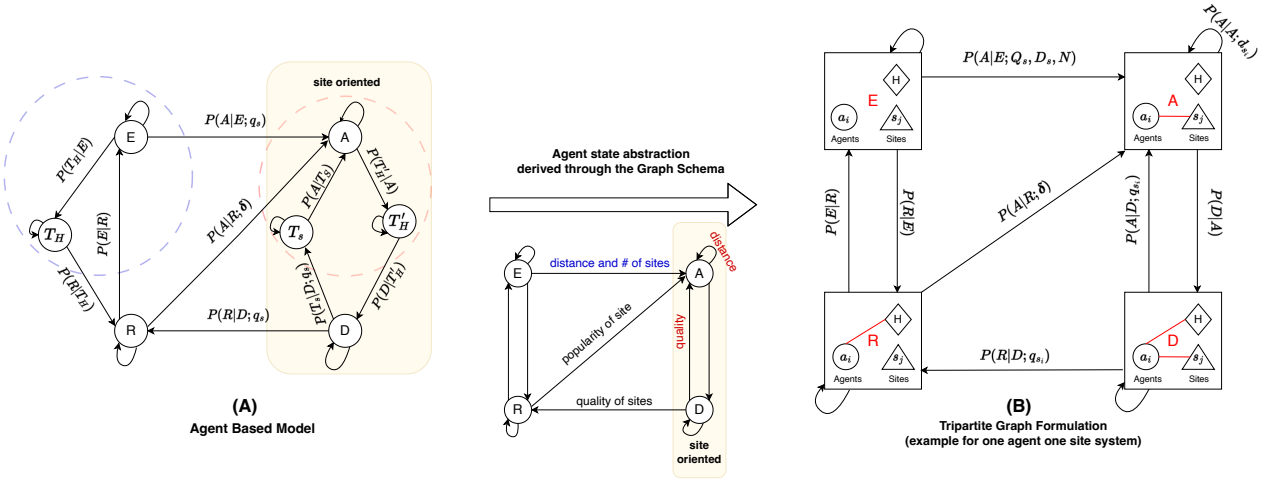

Fig. 2. Deriving a DTMC from an ABM via the graph schema.

II. Related Work

Distributed swarm systems provide a robust and adaptive solution approach to many optimization problems. Instead of using one really sophisticated agent, multiple smaller less sophisticated agents can provide the flexibility to evaluate more options and still provide a "good" answer. The agent-based model used in this paper to solve the best-of-N problem is based on Seeley's work with real honeybee nest selection and related formalisms [4]–[6]. Closely related agent-based approaches for the best-of-N problem [1] have looked at using both a majority and k-unanimity rule for well-timed quorum detection, which reduces swarm resource consumption by preventing unnecessary deliberation.

A survey of best-of-N problems is presented in [7] where they discuss different structures and solutions of this problem. Papers which talk about the best-of-N problem in terms of maximizing network influence include [8]–[10]. Genetic algorithms are also used for maximizing the influence in networks [11]. There has also been work in including time optimality in maximizing network influence [12]. Other papers which address the best-of-N problem include [4], [7], [13], [14] [15, chapter 4]. A voter-based approach is presented in [16], [17], which demonstrated the accuracy of both deterministic and stochastic models in predicting swarm consensus.

The work in this paper is a type of *global-to-local* swarm design [18], in which distributed algorithms are created that yield emergent behavior that satisfies global performance guarantees. Markov chains are used to determine the global properties of the swarm, following previous work on using Markov chains to simulate guidance [19], [20], self-organization [21], task allocation [22] and consensus [23]. Work done using Markov chains for decision models includes [23], where they express majority-rule as absorbing Markov chains, and [24] where they use semi-Markov decision models for real-time scheduling. Markov chains have also been used to show competence of best-of-N colonies [25].

III. From ABM to DTMC

A. Agent Based Model Design

The agent-based model is shown in Fig. 2(A). The model has seven states: (R)est, in which agents are at the hub observing other agents. (E)xplore, where agents follow random paths in the environment in an effort to discover sites. TravelHome (T_H), where agents give up on trying to find a site and return to the hub. (A)ssess a site, TravelHome2 (T'_H), where agents travel from a site to a hub so that they can recruit other agents to the site. (D)ance, where agents at the hub actively recruit resting agents. TravelSite (T_S), where agents who have been recruiting for a site at the hub return to the site to reassess its quality. The figure only shows assessing/dancing for a single site, even though assessing/dancing states are site-specific.

Transitions between states are represented as conditional probabilities, $P(x_{t+1}|x_t;\theta)$, where x_{t+1} is the next state, x_t is the present state, and θ is a parameter that affects the transition probabilities. Self loops encode how long an agent expects to stay in the state. The self-loops on T'_H and T_S encode the distance between site and hub. Site quality affects the transition from explore to assess because sites of terrible quality are ignored and the agent keeps exploring. Site quality affects the probability of transitioning from dancing to resting (and from dancing to traveling to site) because agents make multiple trips to reassess a site when the site quality is high (based on honeybee behavior). Finally, the number of agents dancing for different sites, denoted by $\boldsymbol{\delta}$, parameterizes the probability of transitioning from resting to assessing, which represents the fact that when more agents are dancing for a site then it is more likely that an agent at rest will assess the site.

The algorithms used to implement this state machine are shown (Algorithms 1-5). (Traveling to the site is omitted because the algorithm is trivial.) The parameters were subjectively tuned so that the collective often succeeded in choosing

the highest quality site over a range of site qualities and site distances. A quorum is reached when enough agents are dancing at the hub for the same site.

Agents in the rest state remain there unless (a) there are dancers advertising for a site or (b) agents "get bored" and explore. Given a set of dancers at the hub, resting agents select a site with probability proportional to the number of dancers for each site.

Algorithm 1 Rest

if neighbors > 0 and $\mathcal{U}(0,1) \leq (P_{RA} + 0.5)/20.0$ **then**
  for $s_j \in$ Sites **do**
    $p_j \leftarrow$ # agents at hub dancing for s_j
  Normalize p
  $j \leftarrow$ random.choice(p) ▷ Prefer popular s_j
  $a \rightarrow$ Assess(s_j) ▷ To Assess
else if $\mathcal{U}(0,1) \leq P_{RE}/10.0$ **then**
  $a \rightarrow$ Explore ▷ To Explore

Agents in the assess state stay there until a random trigger sends them to the hub.

Algorithm 2 Assess

if $\mathcal{U}(0,1) \leq P_{AD}$ **then**
  $a \rightarrow$ Travel$_{\text{toHub}}$(from_state=Assess) ▷ To Travel

Agents in the dance state remain there unless they transition to rest, which occurs with higher probability for low quality sites than for high quality sites, or they transition to traveling back to their favored site.

Algorithm 3 Dance

if $\mathcal{U}(0,1) \leq 5^{-1.0 * q_{s_i}}/4.0$ **then**
  $a \rightarrow$ Rest ▷ To Rest
else if $\mathcal{U}(0,1) \leq P_{DA}$ **then**
  $a \rightarrow$ Travel$_{\text{toSite}}$ ▷ To Travel

Agents in the explore state continue to move in a randomly chosen direction unless they encounter a site and choose to assess it (with probability proportional to site quality) or until a random trigger sends them back to the hub to rest. @ Hub in the algorithms denotes the agent is "at Hub", and similarly for sites.

Algorithm 4 Explore

if @ site s_j **then**
  likeSite $= (50.0 * q_{s_j}) * N/4.0$
  if $\mathcal{U}(0,1) \leq P_{EA} *$ likeSite **then**
    $a \rightarrow$ Assess site s_j ▷ To Assess
else if $\mathcal{U}(0,1) \leq P_{ER}$ **then**
  $a \rightarrow$Travel$_{\text{toHub}}$(from_state=Explore) ▷ To Travel

Agents travelling to the hub continue toward the hub until they reach it. Once at the hub, they transition to either the assess state or the rest state, depending on whether they are returning from assessing or exploring, respectively; see Fig. 2(A).

Algorithm 5 Travel$_{\text{toHub}}$(from_state)

if @Hub and from_state==Assess **then**
  $a \rightarrow$ Dance ▷ To Dance
else if @Hub and from_state==Explore **then**
  $a \rightarrow$ Rest ▷ To Rest

Agents traveling to the site continue traveling until they reach the site, and then they assess.

B. Graph Schema

Prior work showed how to construct a DTMC from a four-state ABM [3], but the prior work was not applicable to real-world robots because it did not sufficiently account for travel time and site quality. The four-state model has been implemented in other ABMs [1], [2] in which the effect of distances and qualities were explicitly modeled.

In this paper, we use a four-state model as a *graph schema*. The graph schema is shown in the middle of Fig. 2 and has four states: rest, explore, assess, and dance. Assess and dance are shaded in the figure to indicate that they are *site-oriented*, meaning that agents are assessing or recruiting to a specific site. In the ABM, there are four states that can be considered site-oriented (also shaded): dance, assess, and travelling between hub and site.

Since there is no representation of travel time or discovery time in the graph schema, we group travel time when exploration fails into the graph schema explore state, and we group travel between hub and site while recruiting into the graph schema assess state. The main change that this produces is that the self-loop on the graph schema asses state is now parameterized by distance since abstract assessment will need to include travel time. This choice in the graph schema will be explained further when we discuss how to create the DTMC.

C. From Graph Schema to Tripartite Graph

The graph schema is an abstract encoding of what an agent is doing. Understanding collective state requires mapping the set of graph schema, one for each agent, to a collective representation. The representation we use is a *tripartite graph*, which is a modified version of the bipartite graph used in [3].

The tripartite graph formulation (TGF) uses three types of nodes: agents, sites, and the hub. These are represented in Fig. 2(B) with circles, triangles, and a diamond, respectively. There is a circle node for each agent, a triangle node for each site, and a single diamond node for the hub. The *graph schema state* of the agent is *encoded by the edges incident* to the agent node. The following describes the mapping between (and reasoning behind) graph schema state and agent edges:

- **Explore (E)**: Exploring agents are away from the hub and haven't found sites, which is represented by

agent a_i with no incident edges; see upper left of Fig 2(B).

- **Rest (R)**: Resting agents are at the hub and not site-oriented, which is represented by agent a_i with a single edge to hub H; see lower left of Fig 2(B).
- **Assess (A)**: Assessing agents evaluate a site, away from the hub, which is represented by a node a_i with a single edge to site s_j; see upper right of Fig 2(B).
- **Dance (D)**: Dancing agents are at the hub and oriented toward a specific site, which is represented by agent a_i with an edge to site s_j and an edge to hub H; see lower right of Fig 2(B).

The TGF is constructed by aggregating all nodes into a single graph. There are three partitions: agents, sites, and hub. The presence or absence of edges between agents and sites/hub encode the collective state of the entire swarm. We refer to a specific set of edges in the TGF as a *swarm configuration* because it represents the graph schema for each agent at a particular instant of time.

D. From TGF to DTMC States

We can represent a TGF configuration at time t using a special form of an incidence matrix, $B(t)$ whose rows are indexed by agent nodes and whose columns are indexed by the hub and by the sites. For example, the incidence matrix for a configuration where all but one agent is resting at the hub and the M^{th} agent is recruiting to site s_N is given by

$$B_t = \begin{array}{c} \\ a_1 \\ a_2 \\ \vdots \\ a_M \end{array} \begin{array}{c} \begin{array}{ccccc} s_1 & s_2 & \cdots & s_N & H \end{array} \\ \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \end{array}. \quad (1)$$

The *DTMC state* is defined as the vector of the number of edges incident to the hub and to each site. The number of edges incident equals the degree of the node, which is denoted by k. Thus, the DTMC state at time t is computed as

$$\mathbf{x}_t = \mathbf{1}^T B_t = [k_{s_1}(t), k_{s_2}(t), \ldots, k_{s_N}(t), k_H(t)]^T, \quad (2)$$

where the bold $\mathbf{1}$ is a column vector of all ones. For example, the DTMC state for the incidence matrix in Eq. (1) is $[0, 0, \ldots, 1, M]^T$, which means that $M-1$ agents are resting at the hub (agent nodes attached only to the hub node) and one agent is dancing for site s_N (agent node attached to both hub node and site node s_N).

Dancing and assessing agents favor some site s_j, which is represented by a tuple $(j, q_{s_j}, d_{s_j}, \phi_{s_j})$ where j is a site identifier, $q_{s_j} \in [0, 255]$ is the site quality, and (d_{s_j}, ϕ_{s_j}) is the location of the site in polar coordinates. The hub is at location $(0, 0)$, so d_{s_j} represents the site distance. Agent i knows its degree, k_i, so that it can transition to new states. Agent i knows its position in polar coordinates (r_i, ρ_i) so that it can compute a direction vector between its location and either the hub or a site. Finally, each resting agent at the hub is aware of the number of agents dancing for each site, $\boldsymbol{\delta}_t = [\delta_{s_0}(t), \delta_{s_1}(t), \ldots, \delta s_N(t)]$.

agent i's state	agent i's observation $\mathbf{o}_t$
R	$\mathbf{o}_t = \{k_i = 1, (r_i, \rho_i), \boldsymbol{\delta}_t\}$
D	$\mathbf{o}_t = \{k_i = 2, (r_i, \rho_i), (j, q_{s_j}, d_{s_j}, \phi_{s_j})\}$
A	$\mathbf{o}_t = \{k_i = 1, (r_i, \rho_i), (j, q_{s_j}, d_{s_j}, \phi_{s_j})\}$
E	$\mathbf{o}_t = \{k_i = 0, (r_i, \rho_i)\}$

Given the set of DTMC states, we must define the transitions between the states. DTMC transitions are determined by agents *attaching* or *detaching* from a site or the hub. Recall from Eq. (2) that $\mathbf{x}_t$ denotes the state at time t and $P(\mathbf{x}_t|\mathbf{x}_{t-1}; \theta)$ represents the transition probability between the collective state at time $t-1$ and time t, which depends on parameter vector θ.

E. Monte Carlo Simulation of DTMC

The hardest part of constructing the DTMC is creating the transition probabilities between collective states. Unlike previous work which represented the DTMC transitions explicitly, this paper uses use Monte Carlo samples to derive estimates for success rate and time to completion. Monte Carlo sampling is useful because the number of possible DTMC states grows combinatorically with the number of agents and sites.

Algorithm 6 describes the Monte Carlo simulation of the DTMC model for best-of-N solution to spatial problems. Agents are polled in order (line 5), which doesn't match how real robots would make decisions but which is convenient for the Monte Carlo simulation. Transitions to new DTMC states are done by first determining whether a new edge should be attached to the agent node and then determining whether an existing edge should be detached from the agent node.

Algorithm 6 Monte Carlo Simulation of DTMC

```
1: Agents ← [a_1, a_2, ..., a_M]
2: θ ← [P_RA, P_RE, P_AD, P_DA, P_DR, P_ER, P_EA]
3: x ← [0, ..., 0, M]                    ▷ Initialize all agents at hub
4: while t < t_max do                    ▷ Simulate
5:     for a_i ∈ Agents do               ▷ Poll agents
6:         observe o for agent a_i
7:         x_t ← x_t + attach(a_i, o, θ)
8:         x_t ← x_t − detach(a_i, o, θ)
```

Attachment on line 7 of Alg. 6 represents transitions from explore to assess, explore to rest, or assess to dance states. Attachment increments values to the DTMC state via Alg. 7. Lines 6–11 randomly select a site to attach to based on the number of agents dancing at the site. This matches how agents in the ABM decide when they will assess a site based on the number of agents dancing for it in Alg. 1. Lines 14–18 randomly select a site to attach to based on the quality of the site and the distance of the site from the hub. This matches how agents in the ABM are less likely to discover distant sites and sometimes ignore low quality sites in Alg. 2.

Algorithm 7 attach(a_i, **o**, θ)

1: $r_{\text{val}} \leftarrow [0, \ldots, 0]^T$ ▷ No transition
2: **if** $k_i == 1$ and j known **then** ▷ If agent assessing
3: **if** $\mathcal{U}(0,1) \leq P_{AD} * 250.0/d_j$ **then** ▷ From A to D?
4: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1]^T$ ▷ Attach to hub
5: **else if** $k_i == 1$ and j unknown **then** ▷ If agent resting
6: **if** $\mathcal{U}(0,1) \leq P_{RA}$ **then** ▷ R to A?
7: **for** $s_j \in$ Sites **do**
8: $p_j \leftarrow$ # agents at hub dancing for s_j
9: Normalize p
10: $j \leftarrow$ random.choice(p) ▷ Attach to random s_j
11: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1, 0, \ldots, -1]$ ▷ 1 in column j and -1 in hub column
12: **else if** $k_i == 0$ **then** ▷ If agent exploring
13: **if** $\mathcal{U}(0,1) \leq P_{EA}$ **then** ▷ E to A?
14: **for** $s_j \in$ Sites **do**
15: $p_j \leftarrow 50.0 q_{s_j} + 50.0/d_{s_j}$ ▷ Match ABM
16: Normalize p
17: $j \leftarrow$ random.choice(p) ▷ Attach to random s_j
18: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1, 0, \ldots, 0]$ ▷ 1 in column j
19: **else if** $\mathcal{U}(0,1) \leq P_{ER}$ **then** ▷ E to R?
20: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1]$ ▷ Attach to hub
21: return r_{val}

Detachment on line 8 of Alg. 6 represents transitions from rest to explore, dance to rest, and dance to assess states. Detachment decrements values to the DTMC state via Alg. 8.

Algorithm 8 detach(a_i, **o**, θ)

1: $r_{\text{val}} \leftarrow [0, \ldots, 0]^T$ ▷ No transition
2: **if** $k_i == 1$ **then** ▷ If agent resting
3: **if** $\mathcal{U}(0,1) \leq P_{RE}$ **then** ▷ From R to E?
4: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1]^T$ ▷ Detach from hub
5: **else if** $k_i == 2$ **then** ▷ If agent dancing
6: **if** $\mathcal{U}(0,1) \leq 5^{-1.0 * q_{s_j}}$ **then** ▷ From D to R?
7: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1, 0, \ldots]^T$ ▷ Detach from s_j
8: **else if** $\mathcal{U}(0,1) \leq P_{DA}$ **then** ▷ From D to A?
9: $r_{\text{val}} \leftarrow [0, \ldots, 0, 1]^T$ ▷ Detach from hub
10: return r_{val}

This paper used an iterative hand-tuning design process, but future work should use machine learning to tune the transition probabilities. The hand-tuned parameter vector is

$$\begin{aligned} \theta &= [P_{RA}, P_{RE}, P_{AD}, P_{DA}, P_{DR}, P_{ER}, P_{EA}] \\ &= [0.01, 0.02, 0.05, 0.05, 0.05, 0.005, 0.05]. \end{aligned} \quad (3)$$

The order of the subscripts to P is intended to suggest a constant related to a transition from the state in the first subscript to the state in the second subscript, e.g., P_{RA} is a parameter related to transitioning from rest to assess.

Using predictions from DTMC we can tune the corresponding parameters in the ABM to provide guaranteed performance for the ABMs.

IV. Experiment Design

A simulated world was created with dimensions relative to a simulated robot size. Each simulated robot is a point with a circular sensing radius of 30 units and robot speed of 10 units per time step. Each site and the hub have a radius of 30 units, and the world has dimensions 1600x1200 units^2. The quorum threshold was kept at $0.3 * M$, that is, 30 % of the total agents.

A. Independent and Dependent Variables

The independent variables are the number of sites, number of agents, site distances, and site qualities with values:

Independent Variable	Values
M (# agents)	$50, 100, 200$
N (# sites)	$2, 3, 4$
D (site distances)	$100, 200$
Q (site qualities)	subjectively chosen

We subjectively chose site quality values to create *interesting* worlds. The first criterion for what makes a world *interesting* is that the quality of the best site, $q_{\max} = \max_{s_j} q_{s_j}$, in the world cannot be too low because in such worlds the ABM rarely reaches a decision. Simply put, we did not allow worlds that with $q_{\max} < 128$ because the ABM cannot "solve" these worlds in a reasonable time. The second criterion for creating *interesting* worlds is that the task difficulty increases if the quality of the second best site is close to the quality of the best site. Simply put, it is difficult for the ABM to "solve" the problem if the two best sites have about the same quality. These criteria lead to the following sets of qualities:

# sites	site qualities
$N = 2$	{{0, 128}, {0, 255}, {128, 255}, {50, 160}, {50, 250}, {160, 250}, {129, 128}, {255, 245}, {180, 255}, {190, 245}, {70, 133}, {110, 128}, {90, 248}, {140, 228}, {255, 255}, {100, 175}, {255, 40}}
$N = 3$	{{0, 128, 255}, {0, 245, 255}, {0, 128, 130}, {90, 128, 255}, {200, 228, 255}, {250, 253, 255}, {0, 12, 129}, {90, 128, 144}, {0, 128, 200}, {76, 109, 205}, {127, 128, 120}, {47, 59, 135}, {9, 122, 225}, {254, 128, 255}, {0, 8, 255},{120, 128, 255}, {0, 128, 195}}
$N = 4$	{{0, 128, 255, 254}, {0, 245, 255, 235}, {0, 128, 130, 134}, {90, 128, 255, 34}, {200, 228, 255, 188}, {250, 253, 255, 249}, {0, 12, 129, 255}, {90, 128, 144, 0}, {0, 128, 200, 212}, {76, 109, 205, 150}, {127, 128, 120, 129}, {47, 59, 135, 155}, {9, 122, 225, 0}, {254, 128, 255, 144}, {0, 8, 255, 44}, {120, 128, 255, 200}, {0, 128, 195, 207}}

There are 17 quality sets for each value of N. A *world condition* is a specific set of independent variables (M, N, D, Q). We generate 10 different worlds per world condition, placing sites equidistantly within worlds but at random directions from the hub. We ran 10 trials per world, yielding 100 trials for each of the $3 \times 3 \times 2 \times 17$ world conditions.

The dependent variables in the simulations were the Time-To-Converge (TTC) and Success Probability. TTC was de-

termined when the number of agents dancing for a given site (at the hub) were more than the quorum threshold, which was subjectively set to $0.3M$ (30% of the agent population). A trial was deemed *successful* if the quorum was reached for the best quality site in the environment, and was considered unsuccessful otherwise. Success probability per world condition was computed as the mean of successes for the 100 trials.

B. Discussion of ABM Behaviors

The different sets of qualities strongly affect world difficulty, and it is instructive to understand how world difficulty affects the two dependent variables. Figs 3-4 plot success rate and mean TTC, respectively, as a function of the differences between the qualities of the two best sites. We normalize Figure 4 by the maximum number of time steps (35,000) so that we can look at relative convergence for different simulations. The plots show that success rate is lower and TTC is higher when the two best sites have similar qualities.

Additionally, variability increases when the difference between the qualities of the two best sites is small. This is to be expected since success is defined as choosing the best quality site and the ABM does not always make the best choice in difficult worlds.

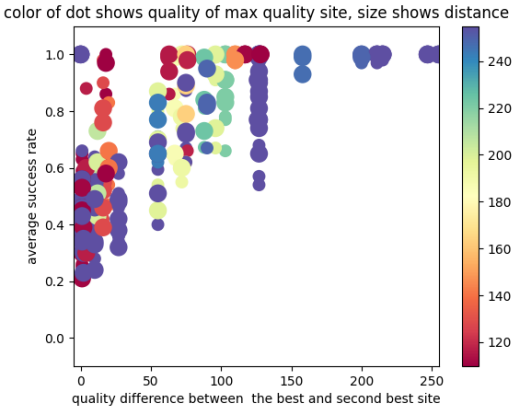

Fig. 3. Mean Success for ABM.

V. Does the DTMC Predict ABM Performance

This paper claims that an abstract DTMC can be designed that (a) models the ABM well and (b) can be used to predict the performance of the ABM since the success probability and TTC are computable for the DTMC. To compare how well the DTMC models the ABM, we use the average difference between the success and time to converge. 100 Monte Carlo samples were averaged using the same world conditions used in the ABM simluations. Figs 5–6 show the mean difference (with inter-quartile ranges) for the success probability and TTC, respectively.

Systems with high site quality differences: For simple worlds, those in which there is a large difference between the qualities of the two best sites, the ABM and DTMC have very similar success probabilities and TTCs.

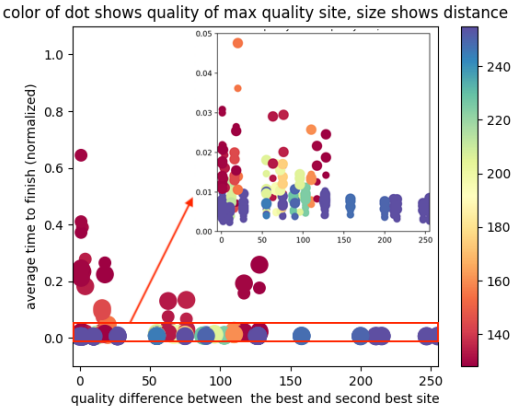

Fig. 4. Mean Time-to-Converge for ABM.

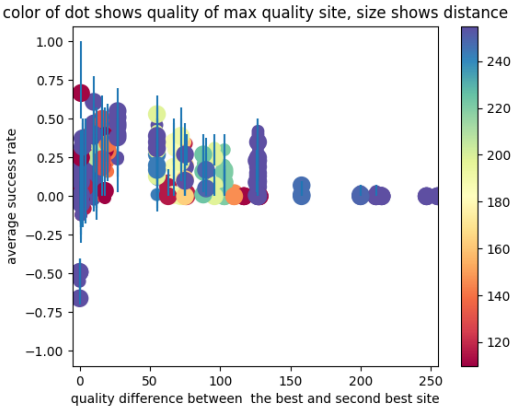

Fig. 5. Difference in Success (TGF - ABM).

Systems with low site quality differences: For difficult worlds, those with small differences between the best two sites, the performance of the ABM and DTMC have higher variability. This is expected for TTC since convergence time is higher for difficult worlds, and thus there is more room for variability. This is also expected for success probability since there is more room for variability when success rate is close to 50% than when it is close to 100%. We hypothesize that an informed Monte Carlo simulation of the DTMC will more easily compute expected performance than multiple ABM trials, but this hypothesis needs to be tested in future work.

Systems with high quality best site: Interestingly, for difficult worlds with a high quality best site (meaning the top two sites have high quality), the TTC variability is usually lower than for difficult worlds with low quality best site. By contrast, the variability in success rate is still large. This is expected since subjective observations of ABM behavior indicate that the first high quality site found is likely the site chosen by the swarm because it is easy to quickly recruit a lot of agents to a high quality site. We hypothesize that if we define success as selecting a site whose quality is within some small percentage of the best site then the DTMC will predict ABM success rates well. This is a problem for future

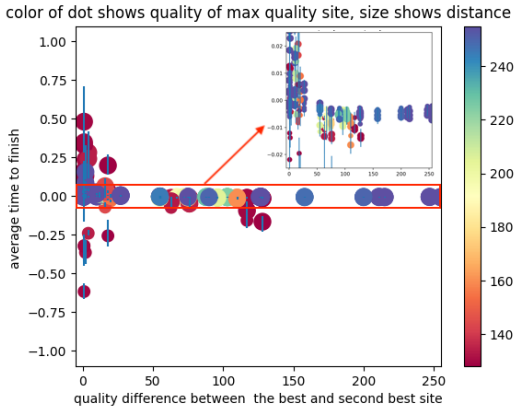

Fig. 6. Difference in Time to Converge (TGF - ABM).

work.

Systems with low quality best site: Systems with low quality best site have a hard time to converge, since the agents keep exploring for more sites and have a hard time having enough agents in the swarm go to it. This can be seen by a wider spread in success of the maroon on the left in Figure 5 and Figure 6. This is also seen in the large times taken by ABM in Figure 4 which is probably due to the repeated travel between sites and hub and from explore to rest, because the site quality isn't good enough, thus there is less attachment and more detachment, delaying the quorum.

VI. Discussion

It is clear from the algorithm descriptions that the parameters used in the DTMC are closely related to some of the parameters used in the ABM. This leads to an interesting ABM design approach, which is supported by subjective observations but not fully evaluated in this paper. The design approach is first to create an ABM that has "good" performance for the best-of-N task under a set of controlled conditions. Second, a DTMC model of the ABM is created that accurately predicts success probability and TTC well under the known conditions. Third, the DTMC parameters are tuned to produce good behavior in new or difficult conditions. Fourth, the algorithms in the ABM are modified so that they more closely align with the parameters in the DTMC. In this design pattern, an optimized DTMC becomes a template for an ABM. This could potentially be very useful for real-world robots where tuning in the real world is very costly.

The subjective observations that support the claim that this design approach can work come in two forms. First, there are biases (e.g., non-zero mean differences) in Figs 5–6. The process of hand tuning parameters demonstrated that results are very sensitive to the values of the parameters in $\text{Travel}_{\text{toSite}}$ and $\text{Travel}_{\text{toHub}}$. Indeed, hand tuning this parameter was very difficult and we found that mistuned DTMC parameters resulted in higher success rates and faster convergence. This suggests that tweaks to the ABM might improve performance, like investing in more energy to increase the speed of the agents in difficult worlds so that delays have a less significant impact on ABM performance.

The second piece of evidence to support the idea that tuning the DTMC can lead to better ABM performance is that we discovered an error in the ABM when we were exploring why the DTMC predicted success in some worlds where the ABM failed.

VII. Summary and Future Work

The paper demonstrated a process by which an abstract model could be created of a multi-robot system performing the best-of-N task. The process used a graph schema to group together some of the stages of an agent-based implementation of the multi-robot system. The graph schema was chosen because it led to a tripartite graph with agents, sites, and a hub. The edges in the tripartite graph could be projected into representation of collective state. The set of possible collective states could be formed into a discrete time Markov chain (DTMC), which was used to accurately predict agent-based performance for easy worlds, with less accuracy for difficult worlds.

The DTMC grows combinatorically in the number of agents and states, so finding efficient approximate representations of the DTMC is desirable. Future work should explore how the complexity can be reduced using node embeddings [26] or by finding inductive relationships using methods such as GraphSAGE [27]. This could enable us to generate predictions using DTMC metrics such as hitting time for larger systems without having to run Monte Carlo simulations.

Future work should also include coming up with a more flexible success metric, such as where selecting a site "close" to the highest quality site is not considered as a failure, but partial success.

References

[1] J. R. Cody and J. A. Adams, "An evaluation of quorum sensing mechanisms in collective value-sensitive site selection," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, (Los Angeles, CA), pp. 40–47, IEEE, Dec. 2017.

[2] A. Reina, G. Valentini, C. Fernández-Oto, M. Dorigo, and V. Trianni, "A design pattern for decentralised decision making," *PloS one*, vol. 10, no. 10, p. e0140950, 2015.

[3] P. Jain and M. A. Goodrich, "Processes for a colony solving the best-of-n problem using a bipartite graph representation," in *Proceedings of the 15th International Symposium on Distributed Autonomous Robotic Systems*, (Virtual Conference), 2021.

[4] T. D. Seeley and S. C. Buhrman, "Nest-site selection in honey bees: how well do swarms implement the "best-of-N" decision rule?," *Behavioral Ecology and Sociobiology*, vol. 49, no. 5, pp. 416–427, 2001.

[5] A. L. Nevai, K. M. Passino, and P. Srinivasan, "Stability of choice in the honey bee nest-site selection process," *Journal of Theoretical Biology*, vol. 263, no. 1, pp. 93–107, 2010.

[6] C. List, C. Elsholtz, and T. D. Seeley, "Independence and interdependence in collective decision making: an agent-based model of nest-site choice by honeybee swarms," *Philosophical transactions of The Royal Society B: biological sciences*, vol. 364, no. 1518, pp. 755–762, 2009.

[7] G. Valentini, E. Ferrante, and M. Dorigo, "The best-of-N problem in robot swarms: Formalization, state of the art, and novel perspectives," *Frontiers in Robotics and AI*, vol. 4, Mar. 2017.

[8] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 137–146, 2003.

[9] D.-B. Chen, H. Gao, L. Lü, and T. Zhou, "Identifying influential nodes in large-scale directed networks: the role of clustering," *PloS one*, vol. 8, no. 10, 2013.

[10] D. Wei, X. Deng, X. Zhang, Y. Deng, and S. Mahadevan, "Identifying influential nodes in weighted networks based on evidence theory," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 10, pp. 2564–2575, 2013.

[11] K. Zhang, H. Du, and M. W. Feldman, "Maximizing influence in a social network: Improved results using a genetic algorithm," *Physica A: Statistical Mechanics and its Applications*, vol. 478, pp. 20–30, 2017.

[12] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 946–957, SIAM, 2014.

[13] Y. Chow, S. Moriguti, H. Robbins, and S. Samuels, "Optimal selection based on relative rank (the "secretary problem")," *Israel Journal of Mathematics*, vol. 2, no. 2, pp. 81–90, 1964.

[14] T. S. Ferguson *et al.*, "Who solved the secretary problem?," *Statistical science*, vol. 4, no. 3, pp. 282–289, 1989.

[15] D. J. Sumpter, *Collective animal behavior*. Princeton University Press, 2010.

[16] G. Valentini, *Achieving Consensus in Robot Swarms*, vol. 706 of *Studies in Computational Intelligence*. Cham: Springer International Publishing, 2017.

[17] G. Valentini, H. Hamann, M. Dorigo, *et al.*, "Self-organized collective decision making: the weighted voter model.," in *AAMAS*, pp. 45–52, 2014.

[18] G. Valentini, H. Hamann, and M. Dorigo, *Global-to-local design for self-organized task allocation in swarms*. IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en …, 2016.

[19] B. Açıkmeşe and D. S. Bayard, "Markov chain approach to probabilistic guidance for swarms of autonomous agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[20] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm," in *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies*, 2013.

[21] I. Chattopadhyay and A. Ray, "Supervised self-organization of homogeneous swarms using ergodic projections of markov chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[22] D. K. Jha, "Algorithms for task allocation in homogeneous swarm of robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3771–3776, IEEE, 2018.

[23] G. Valentini, M. Birattari, and M. Dorigo, "Majority rule with differential latency: An absorbing markov chain to model consensus," in *Proceedings of the European Conference on Complex Systems 2012*, pp. 651–658, Springer, 2013.

[24] Y. Yih and A. Thesen, "Semi-markov decision models for real-time scheduling," *The International Journal of Production Research*, vol. 29, no. 11, pp. 2331–2346, 1991.

[25] X. Cao, P. Jain, and M. A. Goodrich, "Adapted metrics for measuring competency and resilience for autonomous robot systems in discrete time markov chains," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 71–76, IEEE, 2022.

[26] M. Wang, L. Qiu, and X. Wang, "A survey on knowledge graph embeddings for link prediction," *Symmetry*, vol. 13, no. 3, p. 485, 2021.

[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.