Human Factors in Design, Engineering, and Computing, Vol. 238, 2026, 1651–1660
https://doi.org/10.54941/ahfe1008393

Behavior-Aware Arbitration for Scalable Robot Swarms in Cooperative Search-and-Retrieval Tasks

Bradley Killen[1], Kenna Henkel[1], Cindy L. Bethel[1,2], Daniel W. Carruth[1], and Victor J. Paul[3]

[1]Center for Advanced Vehicular Systems, Mississippi State, MS 39759, USA
[2]University of Alabama, Tuscaloosa, AL 35487, USA
[3]U.S. Army DEVCOM-Ground Vehicle Systems, Warren, MI 48397-5000, USA

ABSTRACT

Adding agents to a team can improve search speed and task coverage, but larger teams can also increase coordination load, resource contention, and task interference. This paper studies robot-only swarm behavior as a step toward arbitration support for human-autonomy teaming. We developed a maze-based simulation in which agent teams collect keys, open doors, respond to coordination requests, and attempt to reach a shared objective. Agent behaviors include normal task execution as well as error-prone conditions that produce confused key handling, request overcommitment, task abandonment, or adversarial interference. Structured event logs were parsed into run-level, agent-level, request-level, and decision-level summaries. Experiments across multiple map sizes and team sizes show nonlinear performance scaling. Small maps were nearly always solved across team sizes, while larger maps were more often solved by mid-sized teams and showed degraded performance for larger swarm sizes. Request expirations and timeout-related events increased in higher-load conditions, suggesting that coordination overhead can reduce the value of additional agents. We also trained a contextual candidate-action arbitration model that converts event-stream features into supervisory recommendations, including door-progress prioritization, timeout-risk reassignment, continued exploration, key escort, and reduced reliance on agents associated with disruptive behaviors. The model is intended to centralize swarm activity and highlight when task progress, reassignment, or additional supervision may be needed. These findings support behavior-aware arbitration as a decision-support layer for identifying when larger autonomous teams are improving task progress and when they are increasing coordination burden.

Keywords: Robot swarms, Adaptive arbitration, Multi-agent systems, Autonomous systems, Human-autonomy teaming

INTRODUCTION

Adding more agents to a team does not always make the team better, especially when the added agents increase coordination and supervisory demands (Kolling et al., 2015). A larger group can explore the maze more quickly, find keys sooner, and work on multiple doors or requests at the same time. However, those same agents can also interfere with one another, hold

Received September 25, 2026; Revised October 27, 2026; Accepted November 6, 2026; Available online December 1, 2026

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

resources needed by other agents, or create more coordination requests than the team can resolve. This matters for human-autonomy teaming because a larger swarm may appear more capable while becoming harder to understand, manage, or redirect.

This paper studies robot-only swarm behavior as a step toward behavior-aware arbitration for human-swarm teaming. The goal is to understand how the autonomous team performs before adding human supervisors or collaborators. Specifically, the study examines how team size changes performance, how coordination load grows, and which event patterns indicate whether the swarm is making progress or losing efficiency through unresolved requests, timeouts, and task interference. This robot-only baseline can inform support systems that help a human supervisor decide when to intervene, which agents to trust with a task, or when a task should be reassigned.

The simulation environment supports a cooperative search-and-retrieval task in which agents move through maze environments, collect keys, open doors, make and respond to coordination requests, and attempt to reach a shared objective (Aldridge and Bethel, 2023). Agents operate under normal and error-prone behavior conditions, including confusing keys, overcommitting to perform requests, abandoning tasks, or disrupting team progress. The simulation records structured event logs, which are converted into summaries for swarm-scaling analysis and decision-level records for arbitration.

The analysis shows that swarm performance is nonlinear across map sizes and team sizes. Small maps are solved reliably across the tested team sizes, while larger maps show stronger tradeoffs between coverage and coordination load. Moderate-sized teams can improve task coverage, but larger teams can also show lower success when request expirations, timeouts, and unresolved coordination events increase. The paper contributes: (1) a log-processing pipeline for analyzing cooperative agent behavior in a key-door maze task, (2) an analysis of how mission success and request failures change across map sizes and team sizes, and (3) a behavior-aware arbitration model that converts event-stream features into interpretable supervisory recommendations.

BACKGROUND AND MOTIVATION

Human-swarm teaming introduces supervisory demands that differ from controlling a single agent. As the number of autonomous agents increases, a human operator must monitor not only individual actions but also coordination, resource use, and emerging team behaviors. The human-factors challenge is therefore not simply whether the swarm completes its mission, but whether its behavior remains interpretable enough to support appropriate trust and timely intervention (Kolling et al., 2015).

A larger autonomous team can be useful because it can uncover the environment more quickly and work on different parts of the task in parallel. However, increasing team size can also increase the frequency of coordination problems. Agents may compete for the same key, duplicate

work, block paths, accept tasks they do not complete, or create requests that remain unresolved. These problems can occur in any multi-agent team, but larger teams create more opportunities for them to occur and more events for a supervisor to interpret. In those cases, the swarm may produce more activity without producing more useful progress.

In a small team, a supervisor may be able to follow which agent has a key, which door is blocked, and which requests are active. As the number of agents increases, those relationships become harder to track, increasing the operator's coordination and supervisory burden (Crandall and Goodrich, 2005; Wang and Lewis, 2007). Timeout-related events are one example of this problem. A request that receives no response, a request that is accepted but not completed, and a task that times out during execution represent different coordination failures. For a supervisor, those distinctions matter because each failure type may call for a different response.

Reliance judgments also change over time. An agent that repeatedly completes useful actions may be a better candidate for future work than one that is only nearby (Hancock et al., 2011). An agent that abandons tasks, mishandles keys, hoards resources, or closes doors may be a poor candidate even when it appears available. In larger teams, these behavior patterns are difficult for a person to track manually (Aldridge, Buck, Anderson, Hudson, Smink, Paul, et al., 2025; Hoff and Bashir, 2015).

In this paper, arbitration means using the swarm's event stream to recommend what kind of supervisory action would be useful at a given point in the task. The arbitration model does not replace the human supervisor. It reduces the amount of raw swarm activity the supervisor must interpret by converting event patterns into higher-level action recommendations. These actions include continuing exploration, prioritizing door progress, reassigning a timeout-risk task, supporting an active request, escorting a key toward a door, or avoiding a disruptive agent.

SIMULATION ENVIRONMENT

The simulation builds on prior shared-space human-agent teaming work and a virtual testbed developed for studying interaction, reliance, and shared information in human-autonomy teams (Aldridge, Buck, Anderson, Hudson, Smink, Paul, et al., 2025). A run takes place in a grid-based maze containing a start location, a configurable number of keys and locked doors, and a trophy. Each key corresponds to a door that blocks access to part of the maze. Both the human player and the agents operate within the same environment and share the same overall objective of progressing through the maze to reach the trophy. They also share information about the map, revealing the locations of doors and keys discovered as they explore. Keys may be found and held by different agents, so progressing through the maze might require one agent to request help from another. Mazes can be procedurally generated or created by hand for experiments that require specific layouts. Procedural mazes use a seed, so the same base layout can be recreated consistently across experimental conditions. Maze dimensions, key and door count, starting positions, time limits, and agent composition are also configurable for each run.

Agents move through the maze by planning paths across the cell grid and periodically reassessing their current task to adhere to a predetermined behavior tree. Their general task behavior follows the same ordered set of actions, including delivering a held key, exploring unknown areas to reveal the map, retrieving needed keys, opening doors, and continuing to search and provide information when no task with higher priority is available. Incoming requests will take precedence over the general task behavior.

The simulation currently includes four assignable behavior archetypes representing different agent states. An agent assigned the Normal archetype evaluates requests based on its current situation and completes tasks that it accepts. An agent assigned the Confused archetype accepts requests but, to emulate a system with a camera or visual-processing error, cannot identify the keys in its inventory; as a result, the agent may deliver an incorrect key and still mark the task as complete. An agent assigned the Overachiever archetype accepts requests as they arrive, interrupting its current work to respond to each new request. This behavior can make the agent appear highly responsive and team-oriented; however, it results from a priority-assignment error in which every new request is treated as more important than the agent's existing tasks. An agent assigned the Adversarial archetype represents a failure in cooperative behavior. It refuses incoming requests and may interfere with other agents by closing doors, retaining keys, or moving keys away from useful locations. These behaviors allow different forms of cooperative success and failure to be observed while the agents continue to use the same underlying task-request system, and they extend the testbed's use for studying how information, autonomy, and agent behavior affect human-agent teaming (Aldridge, Buck, Anderson, Hudson, Smink, Paul, et al., 2025; Aldridge, Hudson, Smink, Buck, Anderson, Paul, et al., 2025).

LOG PROCESSING AND FEATURE EXTRACTION

Each simulation produced a structured trace of agent activity during the maze task. The logs recorded task progress, coordination activity, and failure-related behavior, including key collection, door opening, request creation, request expiration, response timeout, execution timeout, task abandonment, disruptive behavior, trophy discovery, and mission completion. Each run was reconstructed from the event stream so that task progress, coordination load, and agent behavior could be measured over time.

The parsed data were summarized at the run, agent, request, and decision levels. At the run level, the parser recorded map size, team size, mission outcome, keys collected, doors opened, requests created, requests expired, and related measures of task progress and coordination load. At the agent level, the parser counted individual contributions, including key collection, door opening, request handling, and failure-related events. This separated overall team performance from uneven agent-level participation.

Request-level summaries were generated because coordination requests were central to the scaling problem. For each request, the processed data recorded whether it was accepted, completed, expired, timed out, or abandoned. This distinction mattered because a larger number of requests did

not always mean better coordination. In larger teams, request volume could also indicate that the swarm was producing more coordination demand than it could resolve.

The same event stream was also converted into decision-level records for arbitration. At selected event points, the parser created a snapshot of the current swarm state. These snapshots included task-progress features, such as keys collected, doors opened, keys remaining, and doors remaining; coordination features, such as requests created, request failures, and recent failures within a time window; and behavior-related features, such as the acting agent's behavior condition and disruptive-event history. Each decision point was then expanded into a candidate-action table by pairing the same state with each possible arbitration action.

SWARM SCALING ANALYSIS

Prior human-robot teaming work has shown that adding agents can increase supervisory and coordination demands, so the first analysis examined how swarm performance changed as the number of agents increased across different maze sizes (Wang and Lewis, 2007; Velagapudi et al., 2008). Each simulation was grouped by map size and configured team size. The main measures were mission success, keys collected, doors opened, coordination requests created, and request failures. Request failures included expired requests and timeout-related outcomes, which were treated as signs that the swarm was generating coordination demand that it was not resolving.

Figure 1 shows the success-rate trend by team size. The small maze served as the high-solvability baseline. Across 200 experimental runs in the small map, the swarm completed 196 missions, for a 98.0% success rate. All tested team sizes performed well, with success rates between 94.0% and 100.0%. In this environment, the task saturated quickly. Once the team had enough agents to cover the maze and complete the key-door sequence, adding more agents did not change the final outcome. The 64-agent condition was not used for the small maze because the map geometry produced excessive collision errors at that density.

The mid-size maze showed a different pattern. Across 716 experimental runs, the swarm completed 472 missions, for a 65.92% success rate. Performance was strong for the 4-agent and 8-agent teams, but success declined as team size increased. The 16-agent condition dropped to 70.75%, while the 32-agent condition fell to 28.95%. The 64-agent condition recovered slightly to 34.55%, but it remained much lower than the smaller-team conditions. The request data explain much of this drop. Mean requests per run increased from 12.43 with 4 agents to 156.47 with 64 agents, while mean expired requests increased from 11.08 to 152.82. In this map, more agents increased activity, but that activity did not consistently become useful task progress.

The big maze produced the clearest nonlinear result. Across 1,116 experimental runs, the swarm completed 603 missions, for a 54.03% success rate. The 4-agent condition performed poorly, with a 35.92% success rate,

suggesting that the smallest team did not provide enough coverage for the larger environment. Performance improved sharply with moderate team sizes. The 8-agent condition reached 96.21% success, and the 16-agent condition reached 92.11%. After that point, performance declined again. The 32-agent condition dropped to 59.10%, and the 64-agent condition dropped to 18.54%. The big maze therefore had an effective middle range. Too few agents limited coverage, while too many agents created enough coordination load, interference, or resource contention to reduce performance.

Across the three map sizes, team size interacted strongly with environment scale. The small maze saturated quickly, the mid-size maze degraded as request failures accumulated, and the big maze showed an effective middle range where 8- and 16-agent teams performed best.

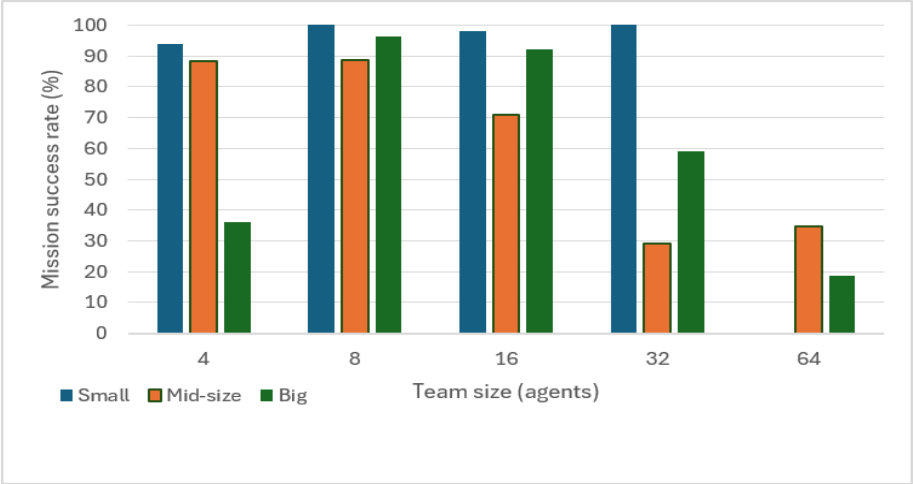


Figure 1: Mission success rate by team size and map size.

These results motivate the arbitration model, which uses event-stream features to identify whether the swarm appears to need exploration, task progress, or recovery from timeout-heavy states.

CANDIDATE-ACTION ARBITRATION MODEL

Final success or failure is not enough to describe how the swarm is behaving; process measures can show where coordination begins to break down before the run ends (Lewis et al., 2011). A run may eventually fail, but the event stream can show earlier signs of trouble, such as repeated request failures, timeout clusters, stalled door progress, or disruptive agent behavior. The arbitration model was built to use those event patterns while the task is still active. Its role is to centralize swarm events, analyze agent behavior, and convert low-level activity into higher-level recommendations that a supervisor could inspect.

The model was trained from decision-level records produced by the log-processing pipeline. At selected event points, the current swarm state was paired with a set of possible arbitration actions. Table 1 summarizes the

candidate actions used by the arbitration model. For each decision point, the model estimated the value of each candidate action and selected the highest-scoring action that was applicable to the current context. This setup follows a contextual candidate-action structure: the same swarm state is paired with possible actions, and the model scores which action is most appropriate for that context. This is related to contextual bandit formulations, where actions are selected based on state information (Gerkey and Matarić, 2004; Li et al., 2010).

Table 1: Candidate arbitration actions used by the behavior-aware model.

Arbitration Action	Purpose
Exploration	Search unresolved areas of the maze or key-door sequence
Door progress	Prioritize actions that move the team toward opening remaining doors
Timeout-risk reassignment	Redirect unresolved or repeatedly timed-out requests
Request support	Assist an active coordination request
Key escort	Move a key or key-carrying agent toward a needed door
Disruptive-agent avoidance	Reduce reliance on unreliable or disruptive agents
Observation	Continue monitoring when no stronger action is needed

The arbitration model only used information available during a live simulation run, including current task progress, recent request failures, behavior condition, and disruptive-event history. This prevented the model from relying on post-run outcomes and made the recommendations suitable for live decision support.

The offline policy did not reduce to one default recommendation. Door progress and timeout-risk reassignment were the two most common actions, while exploration appeared mostly in earlier task states when many keys and doors were still unresolved. This pattern matched the task structure. Once the swarm had found keys and started opening doors, the model often favored door progress. When the event stream showed timeouts, abandoned tasks, or unresolved requests, the model shifted toward reassignment.

The policy also changed with the type of event that triggered the decision. Accepted coordination events more often led to exploration or door-progress recommendations, depending on how much of the task remained. Timeout-heavy events shifted the policy toward reassignment, especially when recent failures were accumulating. This does not mean that every timeout should be handled the same way. In some states, reassignment was the clearer response. In other states, especially when the swarm was closer to completing the key-door sequence, the model still favored door progress.

The trained model was also connected to Unreal for live testing. During these runs, Unreal sent event information to a local arbitration server while the simulation was running. The server returned a recommended supervisory action, and that recommendation was written to the simulation log. The recommendation was not used to change agent behavior during these runs; it was recorded so the model's live outputs could be checked before using it as

an active controller. In this role, the arbiter acted as an intermediary analysis layer, consistent with mixed-initiative and variable-autonomy approaches in which autonomy can support human decision-making without fully replacing the operator (Sellner et al., 2006; Baker and Yanco, 2004). It centralized the event stream, identified the current coordination state, and produced recommendations that could later be shown to a human operator or passed to an active task-allocation policy. Table 2 summarizes the live shadow-mode validation batch. The live recommendations were distributed across timeout-risk reassignment, door progress, and exploration rather than collapsing to one default output.

Table 2: Live shadow-mode arbitration validation summary.

Measure	Value
Shadow-mode simulation runs	100
Runs reaching trophy	87
Runs without trophy collection	13
Recommendations logged	629
Door-progress recommendations	210
Timeout-reassignment recommendations	237
Exploration recommendations	182

Taken together, the arbitration results support the use of the model as a behavior-aware analysis layer for swarm supervision. The model receives event-stream features, scores possible supervisory actions, and produces interpretable recommendations while the task is running. In this role, arbitration serves as a centralized view of what the swarm is doing, which agents may be creating coordination problems, and which task-level response may be worth a supervisor's attention.

DISCUSSION

The results show that swarm size is not a simple performance setting. Additional agents improved coverage only when the environment and task structure could support the added coordination. Small maps saturated quickly, while larger maps showed stronger tradeoffs between coverage and unresolved requests. This matters for human-swarm teaming because a larger swarm can appear more capable yet produce more activity for a supervisor to interpret.

The arbitration model addresses this problem as an event-stream analysis layer. It reduces low-level swarm activity into supervisory cues such as exploration, door progress, timeout-risk reassignment, request support, key escort, disruptive-agent avoidance, or observation. A supervisor does not need every raw event. The supervisor needs to know which situations are becoming difficult to manage and which agents or requests may need attention (Goodrich and Olsen, 2003).

These findings connect the scaling results to the larger human-autonomy problem. The robot-only experiments identify where the swarm succeeds and where coordination begins to break down. The live arbitration layer then provides a way to organize those patterns into recommendations that could support later human supervision or active task allocation.

CONCLUSION AND FUTURE WORK

This paper studied behavior-aware arbitration for simulated swarms in a cooperative search-and-retrieval task. The scaling analysis showed that adding agents did not always improve performance. Small maps saturated quickly, while larger maps showed a tradeoff between coverage and coordination load. Moderate team sizes could improve performance, but oversized teams produced more unresolved requests, timeouts, and inefficient activity.

The candidate-action model converted event-stream behavior into supervisory recommendations for exploration, door progress, timeout-risk reassignment, request support, key escort, disruptive-agent avoidance, and observation. In live shadow mode, the arbiter functioned as a centralized analysis layer that received event-stream features and produced recommendations during simulation runs.

Future work should tune the reward model with a larger validation set, evaluate whether the timeout-heavy guard should remain hand-coded or be learned, and test the arbiter as an active task-allocation component. Human-in-the-loop studies should then measure mission success, workload, trust calibration, intervention timing, and whether arbitration recommendations help supervisors by supporting effective management that surpasses the results in the baseline studies.

ACKNOWLEDGMENT

The authors wish to acknowledge the technical and financial support of the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-24-2-0001 U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC) Warren, MI.

DISTRIBUTION STATEMENT

Approved for public release; distribution is unlimited. OPSEC11097.

REFERENCES

Aldridge, A.L., Bethel, C.L. (2023) “M-OAT shared meta-model framework for effective collaborative human-autonomy teaming”, in: Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. pp. 663–666.

Aldridge, A.L., Buck, A.R., Anderson, D.T., Hudson, C., Smink, K., Paul, V., et al. (2025) “Exploring trust and autonomy: how information affects human-agent teaming performance”, in: Unmanned Systems Technology XXVII, Vol. 13477. pp. 10–26. SPIE.

Aldridge, A.L., Hudson, C., Smink, K., Buck, A.R., Anderson, D.T., Paul, V., et al. (2025) "A Virtual Testbed for the Multidisciplinary Evaluation of Human-Agent Teaming Dynamics", in: International Conference on Human-Computer Interaction. pp. 165–183. Cham: Springer Nature Switzerland.

Barca, J.C., Sekercioglu, Y.A. (2013) "Swarm robotics reviewed", Robotica, Vol. 31, No. 3. pp. 345–359.

Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M. (2013) "Swarm robotics: A review from the swarm engineering perspective", Swarm Intelligence, Vol. 7, No. 1. pp. 1–41.

Crandall, J.W., Goodrich, M.A., Olsen, D.R., Nielsen, C.W. (2005) "Validating human-robot interaction schemes in multitasking environments", IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 35, No. 4. pp. 438–449.

Gerkey, B.P., Mataric, M.J. (2003) "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures", in: 2003 IEEE International Conference on Robotics and Automation, Vol. 3. pp. 3862–3868. IEEE.

Goodrich, M.A., Olsen, D.R. (2003) "Seven principles of efficient human robot interaction", in: SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance, Vol. 4. pp. 3942–3948. IEEE.

Hancock, P.A., Billings, D.R., Schaefer, K.E., Chen, J.Y.C., de Visser, E.J., Parasuraman, R. (2011) "A meta-analysis of factors affecting trust in human-robot interaction", Human Factors, Vol. 53, No. 5. pp. 517–527.

Hoff, K.A., Bashir, M. (2015) "Trust in automation: Integrating empirical evidence on factors that influence trust", Human Factors, Vol. 57, No. 3. pp. 407–434.

Kolling, A., Walker, P., Chakraborty, N., Sycara, K., Lewis, M. (2015) "Human interaction with robot swarms: A survey", IEEE Transactions on Human-Machine Systems, Vol. 46, No. 1. pp. 9–26.

Lewis, M., Wang, H., Chien, S.Y., Velagapudi, P., Scerri, P., Sycara, K. (2011) "Process and performance in human-robot teams", Journal of Cognitive Engineering and Decision Making, Vol. 5, No. 2. pp. 186–208.

Mooers, B., Aldridge, A.L., Buck, A., Bethel, C.L., Anderson, D.T. (2023) "Human-robot teaming for a cooperative game in a shared partially observable space", in: Geospatial Informatics XIII, Vol. 12525. pp. 94–109. SPIE.

Şahin, E. (2004) "Swarm robotics: From sources of inspiration to domains of application", in: International Workshop on Swarm Robotics. pp. 10–20. Berlin, Heidelberg: Springer Berlin Heidelberg.

Sellner, B., Heger, F.W., Hiatt, L.M., Simmons, R., Singh, S. (2006) "Coordinated multiagent teams and sliding autonomy for large-scale assembly", Proceedings of the IEEE, Vol. 94, No. 7. pp. 1425–1444.

Velagapudi, P., Scerri, P., Sycara, K., Wang, H., Lewis, M., Wang, J. (2008) "Scaling effects in multi-robot control", in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2121–2126.

Wang, J., Lewis, M. (2007) "Human control for cooperating robot teams", in: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction. pp. 9–16.