On the Effects of Team Size and Communication Load on the Performance in Exploration Games

Chris Rozemuller[1], Mark Neerincx[1,2] and Koen V. Hindriks[1]
[1]*Intelligent Systems, Delft University of Technology, Mekelweg 4, Delft, The Netherlands*
[2]*TNO, Postbus 23, 3769ZG Soesterberg, The Netherlands*

Keywords: Performance, Exploration Game, Communication, Team Size, Topology, Resource Redundancy, Task Size.

Abstract: Exploration games are games where agents (or robots) need to search resources and retrieve these resources. In principle, performance in such games can be improved either by adding more agents or by exchanging more messages. However, both measures are not free of cost and it is important to be able to assess the trade-off between these costs and the potential performance gain. The focus of this paper is on improving our understanding of the performance gain that can be achieved either by adding more agents or by increasing the communication load. Performance gain moreover is studied by taking several other important factors into account such as environment topology and size, resource-redundancy, and task size. Our results suggest that there does not exist a decision function that dominates all other decision functions, i.e. is optimal for all conditions. Instead we find that (i) for different team sizes and communication strategies different agent decision functions perform optimal, and that (ii) optimality of decision functions also depends on environment and task parameters. We also find that it pays off to optimize for environment topologies.

1 INTRODUCTION

Exploration games are games where agents (or robots) need to search for resources and retrieve these resources (Hindriks and Dix, 2014). Many real-life applications are instances of such games including, for example, package delivery problems (which sometimes only require minimal search) to search and rescue missions (where search typically takes most of the time). A task in an exploration game is defined by a specific finite (sub)set of all available resources that need to be located and retrieved in a particular order (a task is defined as a sequence of resource types). The order imposed on the items to be retrieved is a key difference with typical foraging tasks. We assume that the map (i.e., topology of the environment) that needs to be explored is finite and known but that the initial distribution of resources is unknown. In this paper, performance in exploration games is measured by the time to complete a given task.

Task performance in exploration games can be improved by adding more agents because, in principle, they can perform tasks in paralelle. This is true even if agents do not communicate with each other. If resources are sufficiently available and agents act rationally, it is possible to solve an exploration game without any communication. The “only” condition that agents that do not communicate need to satisfy is that they do not waste resources (they need to ensure that resource consumption is necessary to complete the task). The performance gain of adding one more agent, however, decreases relative to the number of agents that are already deployed. Even worse, if physical size of robots and the space they occupy is also taken into account, there typically is a point where adding more robots will decrease performance again as robots become obstacles blocking each other’s movement (Rosenfeld et al., 2006). But even if we abstract from such ‘navigational issues’, as we will do in this paper, and we can safely assume that adding more agents will not decrease performance, we cannot assume that adding more agents to the mix will increase performance. Finite tasks that can be completed can only require at most a finite amount of effort, which means that there must be a point at which adding another agent will not yield any performance gain any more.

Besides by adding more agents, performance can usually also be improved by adding communication between agents. Communication, for example, can be used to avoid duplication of effort. If agents inform each other about the locations they have visited, for example, agents can avoid exploring that lo-

Rozemuller, C., Neerincx, M. and Hindriks, K.
On the Effects of Team Size and Communication Load on the Performance in Exploration Games.
DOI: 10.5220/0006661002210230
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 2*, pages 221-230
ISBN: 978-989-758-275-2

cation twice. Similarly, by communicating about the targets agents set themselves (their goals) agents can avoid retrieving the same type of resource twice. It is known from the literature that even a limited exchange of messages can already have a huge impact on performance, see e.g., (Farinelli et al., 2004). Given that exploration games are finite, it also is clear that there is some point at which more communication does not lead to any performance gains any more. A team of agents will only be able to perform better if they can further improve the coordination of their actions by communicating more. We aim to increase our understanding of how much communication can contribute to the performance of an agent team in exploration games.

Like foraging games, optimal solutions are unknown for exploration games in general and therefore agent-based simulation approaches are used to empirically establish the performance of a coordination strategy (Zedadra et al., 2017). The results from this empirical research can then be used to design better and more efficient coordination strategies for different types of exploration games. Our work is motivated by this and aims to provide guidelines that can inform this design. As a designer, it is particularly useful to understand how much performance can be improved by either adding another agent to the mix or by increasing the communication load for a given number of agents. The number of agents (robots) and number of messages exchanged between agents on average can be viewed as a budget that is available to a designer. It is useful for a designer to better understand the return on investment of adding another agent or increased the communication load.

In general, from a design perspective, it is simpler to add another agent to a system than to increase the messages that agents exchange. Exchanging more messages typically requires a more complicated coordination strategy to be effective and thus complicates system design because interdependencies between agents are increased. A more complicated coordination strategy, moreover, comes at the cost of higher processing power, additional requirements on hardware, and higher risks of failure. Additional design complexity, however, may be justified when communication can yield dramatic performance gains. This is sometimes the case, as we noted above, but to understand when requires an insight into when such performance gains are to be expected. Providing this kind of insight is one of the aims of this paper.

The main contributions of our paper are (i) that we provide convincing evidence that there is not a single coordination strategy that is optimal for all cases and (ii) show which type of coordination strategies are best suited for optimizing performance for specific map topologies of an exploration game. We also show how the performance of different strategies depends on team size and communication load, and how performance is influenced by additional factors such as map and task size, and resource redundancy.

The remainder of the paper is organised as follows. Section 2 discusses related work. In Section 3 we introduce our approach and discuss the agent decision functions used in our simulations. Section 4 presents the experimental set-up we have used to study performance gains. In Section 5 we discuss our results. Section 6 concludes the paper.

2 RELATED WORK

The effects of coordination and communication on performance have extensively been studied empirically for real robot systems; (Farinelli et al., 2004) provides a good survey. This survey presents a detailed overview of coordination mechanisms that have been proposed and concludes that to obtain reasonable performance in most cases little communication is required. However, none of the reported studies provides a detailed study of the trade-off between communication, team-size, and performance.

(Pitonakova et al., 2016) demonstrates the value of coordination by showing that both social and non-social coordination mechanisms, i.e. with and without communication, can improve a robot team's efficiency. (Pini et al., 2013) studies coordination mechanisms in relation to how tasks are partitioned. They conclude that communication is beneficial to avoid duplication of effort, but has the drawback of biasing the exploration, even slowing it down in some cases as a result. These works focus on issues such as avoiding collisions and path finding, whereas our focus is more on task related coordination issues, such as avoiding duplication of effort and efficient destination allocation. Our results, moreover, go beyond these studies by providing a more detailed overview of which performance gains can be achieved by means of increasing communication and team-size, while taking the influence of various environment factors into account.

(Liemhetcharat et al., 2015) uses a set-up similar to ours, but studies heterogeneous teams instead of homogeneous teams as we do, and a setting where resources are replenished instead of consumed as is the case in our work. Our focus, moreover, is on communication load and we take more environmental factors into account.

Several simulation-based studies that also investigate the relation between team-size, communication, and performance have used the Blocks World for Teams (BW4T) simulation environment. BW4T is a testbed for exploration games (Johnson et al., 2009) with blocks of different colors as resources and has been specifically designed for analysing and evaluating the ability to cooperate in multi-agent teams. Both (Harbers et al., 2012) and (Wei et al., 2014) use BW4T to investigate the impact of different types of communication on team performance. These works examined agents that use four different communication protocols: (i) agents that do not communicate, (ii) agents that only exchange information about the knowledge they have about the environment, i.e. the location of resources, (iii) agents that only communicate their intentions, i.e. what they plan to do, and (iv) agents that both communicate about their knowledge and intentions. (Harbers et al., 2012) concludes that it is more effective to communicate about intentions than about knowledge. (Wei et al., 2014) shows, moreover, that interference between robots can diminish the positive effects of communication. Running simulations using BW4T takes time, however, which has limited the size of the experiments that could be run to less than 10 agents and small environments. In contrast, we vary the size and topologies of environments in our experiments, systematically explore the impact of various other environment parameters, and vary team sizes from 1 to a 100 agents.

3 SIMULATION APPROACH

We use a discrete simulation model to empirically investigate performance of various agent decision functions for exploration games. In our approach we systematically vary parameters that define exploration games, including map topology and size, distribution of different types of resources, task size, and number of agents (team size) deployed for completing the task. We also systematically vary basic tactics that agents use for exploration and coordination.

3.1 Simulator

We use a very fast multi-agent system simulator for exploration games developed in MATLAB called MEG (short for Matlab Exploration Game). The map topology of an exploration game is modelled by a (symmetric) distance matrix that consists of distances between each pair of locations on the map. We assume that a unique location is singled out on the map as the target location where resources need to be delivered called the *drop zone*. Even though a single drop zone somewhat simplifies the task, we believe this constraint is reasonable for the purposes of this paper and also limits the number of topologies that we need to simulate to a feasible number. The simulator keeps track of which resources are being retrieved and moved to other locations. Each time an exploration game is loaded and initialized the simulator randomly distributes a pre-defined amount of resources on locations on the map. The initialization of the task sequence is based on the redundancy of resources and on task length parameters. All agents initially are located in the drop zone.

MEG is a discrete event simulator where all agents perform actions simultaneously at a discrete point in time T and the effects of these actions on the environment are computed at the end of each time step. After each step the global clock T is increased to $T+1$. Agents maintain a model of the environment (their beliefs) and set targets (goals to go somewhere or retrieve a known resource). The simulator automatically updates the beliefs of agents based on the resources perceived at the location an agent is at and the messages received from other agents at the end of each time step. The performance of the simulator thus effectively only depends on the time needed to compute a decision on which action to perform next for each agent and to compute the effects of performing these actions simultaneously on the environment. Our simulator is fast enough to run millions of simulations in a reasonable time (compared with, for example, a real-time simulator such as the BW4T we gain a speed increase of roughly a factor $25,000$).

Figure 1 shows the control flow that is executed by the simulator. Choices are in blue, actions are in gray, and for each time step T the green block repeats the same cycle again by executing all agents for as long as the task set has not been completed yet. For each agent a, the simulator keeps track of two important parameters: the target location $R(a)$ and $N(a)$ the time that agent a is estimated to arrive at and occupy that location. As long as $N(a) > T$, at a time step an agent performs one step towards its target location lowering the distance that still needs to be travelled. Once an agent reaches its target location, i.e. $N(a) = T$, it gets access to this location if it is not occupied by another agent. If the agent gets access, it occupies the location and the boolean mapping *occupied* is updated to model this fact. We note that agent movement is not obstructed by locations that are already occupied but can move freely past such locations. An agent that *occupies a target location* either drops a resource that it retrieved previously, retrieves a resource that is available at the location if it believes that resource still

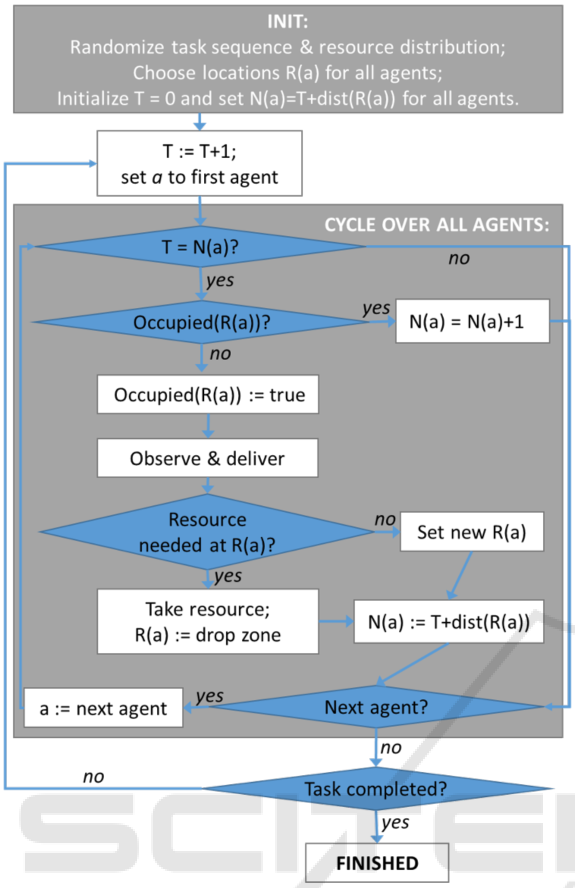


Figure 1: Control Flow of MEG Simulator.

needs to be delivered at the drop zone, or otherwise selects a (different) target location. Selection of a target location depends on the agent's exploration strategy. If the agent cannot access the location yet, the estimated time of arrival $N(a)$ is increased with one and at the next time step it is checked again whether the agent can access the location.

At a target location agents can observe which resources are available at that location. At the beginning of each turn agents are updated on task progress, i.e. on the resource types that still need to be delivered. The simulator also allows agents to communicate updates on perceived resources (belief updates) as well as changes to the targets that they set for themselves (their goals) *to all other agents*. That is, agents can broadcast their belief updates and goals. Whether agents do so depends on the decision function they use. Messages sent to an agent are available to that agent at the beginning of the next turn.

The output of a simulation run consists of the *number of turns*, i.e. T, that were needed to retrieve and deliver all resources required to complete the task and the *average number of messages each agent sent*.

3.2 Environment and Task Parameters

The simulation model allows for varying a number of parameters, including the following:

- **Size of the map**, i.e. the number of locations;
- **Map topology**, or structure of the map, i.e. whether locations are connected and, if so, what the distance between these locations is;
- **Task size**, or the length of the task sequence, i.e. the number and order of the resource types that need to be located and retrieved;
- **Resource redundancy**, i.e. a multiplier r that ensures that a the number of items of a particular resource type available on the map is r times the number that is actually needed to complete the task; $r = 1$ means that the resources available exactly match what is needed.

The map size and topology parameters determine the cost of travelling to a location. By varying the distances between locations we aim to establish when a random exploration tactic will perform better than a greedy tactic. If, for example, the distance between any two locations is the same, then randomly selecting a room cannot result in increasing the travelling costs. If, on the other hand, some locations are much more distant than other locations, then at some point it may become more efficient to apply a greedy tactic and select the closest location to avoid having to travel long distances.

By increasing the task size, we aim to establish which tactics will increase the efficiency of larger teams more because they facilitate multiple agents to perform more subtasks effectively in parallel. Finally, we aim to verify whether a higher resource redundancy factor r will favour greedy and communication tactics because resources can be assumed to be available closer to an agent's current location.

3.3 Agent Decision Functions

In order to complete a task, agents need to explore and coordinate their efforts to locate and retrieve resources. We therefore specify basic tactics for exploration and coordination which can be combined to obtain different types of agent decision functions. In this work we assume agent teams are homogeneous, i.e. all agents use the same strategy, and do not consider heterogeneous teams.

Exploration Tactics The basic tactics that we consider for exploration are a greedy and a random tactic. We assume that agents are always greedy, i.e. select the closest location, if they know a location where a resource can be found that is needed next. (Recall that we assume the map is known.) If an agent uses the *greedy exploration tactic* it will also select the closest location that has not yet been visited. By default, we assume that an agent is a greedy explorer. The random tactic instead selects a random location from those that have not been visited yet. We further differentiate by considering when the random exploration tactic is applied: at the beginning of a game (*random start tactic*) and thereafter during the rest of the game (*random exploration tactic*). Of course, an agent may both select a target location randomly at the start as well as during the game, thus effectively applying the combination of both tactics. Furthermore, we assume that our agents are persistent: They will keep trying to get access to a location until it becomes available and will not select a new target location before they have gained access to their current target location. We thus have two tactics that an agent can use instead of the default greedy exploration tactic:

- **Random Start Tactic.** At $T = 0$, agents randomly select a location that has not yet been visited to go to instead of going to the closest location first.
- **Random Exploration Tactic.** At $T > 0$, agents randomly choose a target location that has not been visited to explore next.

The effect of randomly selecting an (initial) target location is that agents will more evenly distribute over the map. This usually will reduce duplication of effort as fewer agents will try to visit the same room. The downside of an agent that applies a random tactic is that on average it will increase the distance travelled compared to an agent that uses a greedy tactic. The random start tactic will only initially give rise to a more even distribution on the map whereas the random exploration tactic will ensure exploration of all parts of the map more evenly later on in the game.

Coordination Tactics. Agents that are careful not to waste resources do not need communication to complete a task in an exploration game. By default, we therefore assume that agents do *not communicate*. Agents, however, can coordinate their efforts better when they exchange information. We consider two communication tactics. First, agents that use the *updates communication tactic* exchange updates on the locations that they visit: They inform other agents about which locations they have visited and share the information about resources found at those locations with other agents. Other agents use this information to not (re)visit a location already visited by another agent and to (greedily) select locations where resources needed can be retrieved by using the information about resources they thus obtain. In contract with exploration tactics, which only affect the agent's *own* behaviour, it is important to realize that communication tactics have an effect on the behaviour of *other* agents. Second, agents that use the *target communication tactic* share with other agents which resource they are delivering when they retrieve that resource at a location. Other agents use this information to not also then target the delivery of that same resource type but instead will focus on delivering the resource needed next.

- **Updates Communication Tactic.** Agents communicate about which locations they have visited, and about which resources are (no longer) available at a location. Other agents will not consider exploring locations that have been explored already and will retrieve resources based on information received from other agents.
- **Target Communication Tactic.** Agents communicate about which resource they are delivering when they retrieve a resource at a location that is required next to complete the task sequence. Other agents will anticipate and not deliver the same resource.[1]

The cost of communication is based on the total count of all messages that are sent from one agent to another agent. That is, all individual messages are counted instead of counting the single broadcast action that sends a message to all other agents as a single message. We do so because each individual message demands resources and requires establishing the reliability of the transmission of a message from one agent to another agent. Generally speaking, the target communication tactic results in fewer messages being sent than the updates communication tactic. The former tactic only requires one message to be sent for each resource needed to complete the task whereas for implementing the latter tactic a message needs to be sent for every resource discovered and each room that is visited for the first time.

Admittedly, our tactics are quite basic and can be refined to obtain more sophisticated variants of these tactics. We believe, however, that for our purposes these tactics are useful as they provide for basic but fundamentally different strategies to complete tasks

[1]Since communication is only available at the beginning of the next turn this introduces a new issue: Multiple agents can decide to deliver the same resource. Therefore, at the drop zone, agents also check if their resource is still required eventually. If not, they will abandon it and continue with the remainder of the task.

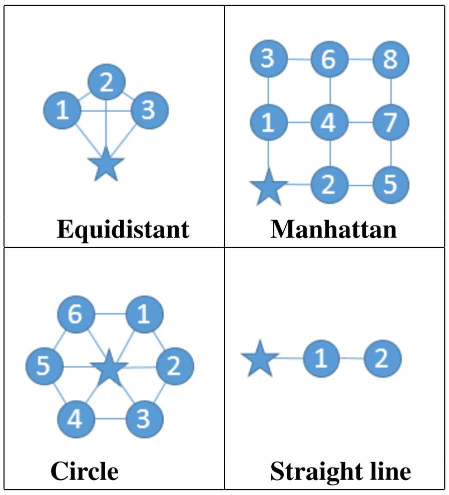
in an exploration game. They allow us to establish the effects of almost completely opposite tactics, i.e. those of greedy versus random exploration tactics and of no communication versus the communication of updates and chosen target locations. For exploration, for example, we thus can establish whether focus (on nearby locations) versus spread (aiming for visiting as many different locations as possible) improves performance in an exploration game. This allows us to investigate the efficiency of tactics that are generally applicable to robots that are tasked with exploration rather than to focus on details of very specific instances of exploration games.

Table 1: Labels of agent decision functions.

	Random Start	Random Exploration	Updates Comm.	Target Comm.	Optimal (Sometimes)
A					*Yes*
B			✓		*Yes*
C		✓			No
D		✓	✓		No
E				✓	*Yes*
F			✓	✓	*Yes*
G		✓		✓	No
H		✓	✓	✓	No
I	✓				*Yes*
J	✓		✓		No
K	✓	✓			*Yes*
L	✓	✓	✓		No
M	✓			✓	*Yes*
N	✓		✓	✓	*Yes*
O	✓	✓		✓	*Yes*
P	✓	✓	✓	✓	No

Decision Functions. The four tactics introduced above can be used to create variations of the default greedy exploration agent that does not communicate at all. This gives rise to 16 different agent decision functions. Table 1 introduces labels (single letters) used to reference these decision functions in the remainder of the paper. For example, agents that use decision function *G* initially choose to visit the location closest to it and then randomly visit unexplored locations until a resource that is needed is located; upon retrieving a resource they communicate to all other agents that they will deliver this resource but do not update other agents about locations visited and resources discovered. The last column in the table indicates whether a decision function is optimal at least some of the time, i.e. dominates other decision functions for a specific experimental condition. Decision functions that are never optimal are always outperformed by another decision function.

Table 2: Topologies used in our simulations.



4 EXPERIMENTAL SETUP

In our simulation experiments we varied the parameters discussed in Section 3.2. For map size, we varied the number of locations and used **10, 20 and 40 locations**. We defined **4 different topologies** illustrated in Table 2. Edges that connect locations in Table 2 have a distance of one.[2] In the *equidistant topology* the distance to every other location is the same. In the *Manhattan topology* all locations are placed on a grid with the drop zone located in a corner of the grid. In the *circle topology* all locations are placed in a circle around the drop zone. Finally, in the *straight line topology* all locations are placed on a straight line with the drop zone located at either end of that line.

We varied task size and used **sequences with length 3, 6 or 12 resources**. In our simulations we used 7 resource types. A task sequence of 7 resources thus could require agents to retrieve resource items each of a different type in a particular order. The constraint that resources need to be delivered to the drop zone in a particular order complicates the task if multiple agents work on it in parallel as coordination may be required to avoid duplication of effort. Resources needed are randomly distributed over available locations with a **redundancy factor of 1, 2 or 4**. If, for example, the task requires 2 resource items of type τ and the redundancy factor is 4, then 8 resource items of type τ wil be made available on the map. The choice of resources that need to be collected to complete a task and the distribution of resources in the environ-

[2]This may not always result in a feasible geometry in 3d space using only straight lines; we use these structures to demonstrate coordination issues related to connectivity and distance.

ment, moreover, was randomized for each simulation run. Finally, we varied team size and performed simulations with **1, 2, 3, 5, 10, 20, 30, 50, and 100 agents** and used all **16 agent decision functions** specified in Section 3.3.

Combined, these variations of parameters define $3*4*3*3*9*16 = 15,552$ conditions. Each type of simulation condition is repeated a 100 times to average out variation, thus giving a total of $1,555,200$ different simulation runs that were performed.

The parameter settings that we have chosen for our simulations allow us to evaluate the efficiency of a range of coordination strategies and tactics in the smallest possible environments that are still interesting as well as in very complex environments. We have also included team sizes that are more than twice as large as the largest number of locations which ensures we will reach a saturation point in which all locations will be permanently occupied and adding another agent can at best only have a marginal effect.

4.1 Performance Normalization

The main performance measure of a simulation run is the number of time steps T it takes to finish a task. The value T for a specific run depends on environment and task parameters, tactics, and team size and therefore these values can differ widely for different simulation runs. To be able to compare the performance for different parameter sets we normalize T by scaling it to a value between 0 and 1. We do so by means of the theoretically minimal time needed to complete a task T_{opt}, i.e. the optimal lower bound to complete the task possible, and use the average time T_{single} to complete a task by a single greedy agent as an upper bound. It is reasonable to assume that multiple agents should be able to outperform a single greedy agent and would need fewer than T_{single} steps.

We can compute the lower bound T_{opt} analytically by assuming that the locations of all resources are known and the agents available perform sub-tasks in parallel. Of course, without prior knowledge of resource locations it is very hard to get close to this theoretical optimum as it requires each agent to travel directly to the right locations. We determine the upper bound T_{single} empirically by running simulations for each set of environment and task parameters also for a single agent. We use the greedy agent A (see Table 1) as our results show that it is most efficient if we use only a single agent to complete a task.

The normalized performance measure T_{norm} then is computed for each team size consisting of a specific type of agent as follows:

$$T_{norm} = \frac{T - T_{opt}}{T_{Single} - T_{Opt}} \quad (1)$$

Note that we must always have $T_{opt} < T_{single}$ for tasks of length > 1 as multiple agents can perform subtasks in parallel. The value T_{norm} is independent from environment and task parameters and therefore provides an indication of the ability of the agents to coordinate their efforts.

5 RESULTS AND DISCUSSION

We computed the average performance measure T_{norm} and the communication load of agents (i.e., the number of messages exchanged per agent) for each simulation condition that was run a 100 times. We thus obtained performance measures for different types of budgets (i.e., different investments of number of agents and communication load). We plotted these outcomes in a 3D point cloud with coordinates performance, team size, and communication load. For each configuration of agent type, task size, and environment type (map size, topology, resource distribution), an optimal performance point can be found in this cloud for any team size and communication load coordinate. We averaged the best performance for each of these coordinates and linearly interpolated between the data points to obtain the 2D heat map of Figure 2, assuming that performance never degrades when more resources are invested. The colours in this map show the best performance T_{norm} that we found for each type of budget. Blue corresponds with a lower T_{norm} value, i.e. better performance, and red with a higher T_{norm} value. The color of each point in this map thus indicates the best possible performance for a given budget of agents and messages.

Figure 2 shows different bands of the same colour, e.g., green for $T_{norm} = 0.5$, where performance is the same. These iso-performance frontiers or *performance levels* visualize the trade-off between the investment of more agents or communication resources that designers of a multi-robot system can make to obtain a desired performance. For example, the figure shows that a performance of .4 can be achieved with about 20 agents but also with only 7 agents that communicate with each other. Interestingly, all of these frontiers show a more or less sharp edge around 20 messages per agent. This suggests that agents with on average that kind of communication load are a good choice to obtain a desired performance level. The performance gain of investing more agents that communicate less is small and increasing the communication load would still require almost the same number of robots to achieve a similar or better performance. Our results thus suggest that providing agents with basic communication skills can significantly reduce the

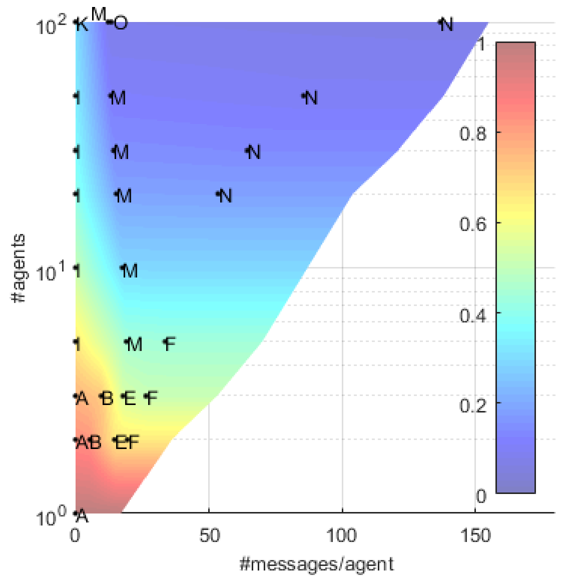


Figure 2: Performance heatmap with optimal agent types.

number of robots that are needed to achieve a specific performance level.

5.1 Performance and Tactics

Figure 2 also identifies for a given budget which agent types (see Table 1) are able to perform best and at what performance level. It is interesting to observe that for different budgets different agent types perform best. In other words, there is no single agent type that dominates all others and performs bests for arbitrary budgets. For example, for budgets with less than 10 agents, agent types A, B, E, and F perform best but performance is rather low with a normalized performance $T_{norm} > .6$, whereas for budgets with more than 10 agents types K, M, N, and O perform nearly-optimal. For small budgets agent type E, a greedy agent that communicates about target delivery, seems a particularly attractive choice. For larger budgets M, a greedy agent with random start and target communication tactics, is a particularly attractive choice.

If we average over environment and task parameters, moreover, we can conclude that 7 of our 16 agent types are dominated by some other agent type. Generally speaking, we find that an initial random distribution of agents becomes more important when team size becomes larger and that target communication makes a larger team more effective. Somewhat remarkably the updates communication tactic is only useful for small budgets; this appears to be the case because the target communication tactic provides sufficient information for coordinating larger agent teams. Note, however, that if we do not average over all parameters

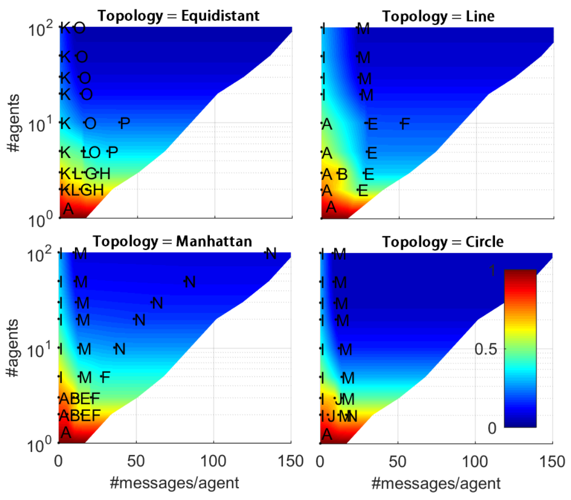


Figure 3: Heatmaps for different topologies.

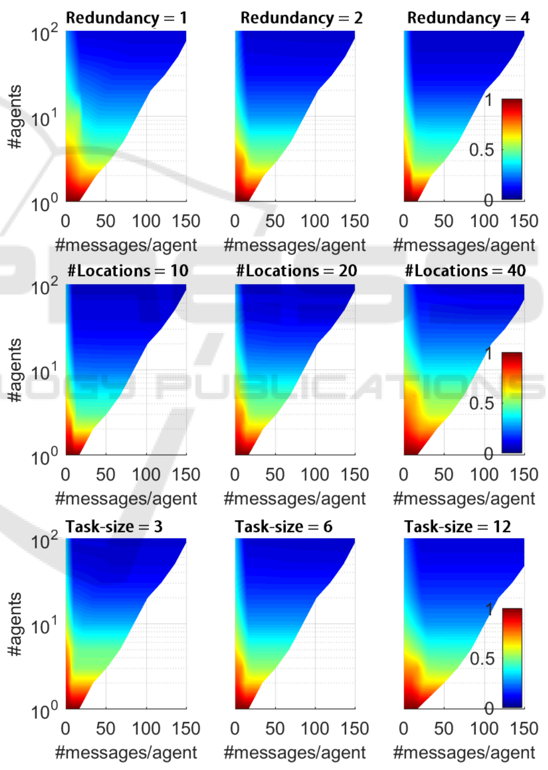


Figure 4: Heatmaps for environment parameters.

other tactics may become relevant again. For example, for specific topologies such as e.g. the equidistant topology and for small budgets the random exploration tactic (agent type G, see Figure 3) can outperform other tactics.

5.2 Environment Influence

Figures 3 and 4 show how different environment and task parameters influence the performance for different budgets. These figures show best performance for specific values of a parameter while averaging over all other parameter settings. For example, the left-upper heatmap in Figure 3 shows the performance heat map for the equidistant topology.

By comparing heat maps for different redundancy factors, we can conclude that impact of redundancy on performance is rather small; there is less need for communication for higher redundancy factors. The impact of map size on performance is as one would expect: larger maps require more agents to achieve similar performance levels. In contrast, if the task size is increased, more communication between agents is needed to achieve similar performance levels.

Finally, we find that the type of topology has a rather large effect on the shape of the performance levels that are visible in the heat maps (see Figure 3). Most notably, whereas for most parameters the agent types that perform best match those of Figure 2, this turns out to be not the case for different topologies. The heat map for the Manhattan topology matches best with the heat map of Figure 2 averaging over all parameters. But for other topologies the heat maps are quite different. For example, we find that on a line and equidistant topology the target communication tactic can significantly increase efficiency (agent I versus M and K versus O), but the update communication tactic only yields significant performance gains on a Manhattan topology. We conclude that it is particularly interesting to fine-tune and optimize an agent decision function for a specific topology.

6 CONCLUSIONS

This paper investigates what the best performance is that can be achieved with a given budget, i.e. an investment of a specific number of agents and communication load per agent. We use a simulation approach and a discrete event simulator for exploration games to empirically obtain insights in how performance depends on different tactics used for composing a strategy for deciding what to do next. Several exploration tactics including greedy and random exploration tactics and several communication tactics are evaluated. We find that there does not exist one dominant strategy but that for different budgets different sets of tactics perform best.

Our results can inform designers of multi-agent systems for exploration game type applications. First, our results can inform the choice of budget itself and can be used to make a trade-off between budgets and performance. Moreover, we found that certain combinations of tactics are outperformed by other strategies and thus are best avoided. Finally, we have shown that for different environment and task parameters different strategies perform best. In particular, we found that fine-tuning of agent coordination strategies is particularly useful if agents only have to handle a specific type of environment topology.

In future work we plan to refine and evaluate the agent tactics used in this paper and to study particular mechanisms for optimizing performance in specific types of environment topologies.

ACKNOWLEDGEMENTS

This work was supported by European Union's Seventh Framework Programme for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763.

REFERENCES

Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multirobot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015–2028.

Harbers, M., Jonker, C., and Van Riemsdijk, B. (2012). Enhancing team performance through effective communication. In *Proceedings of the 4th Annual Human-Agent-Robot Teamwork Workshop*, pages 1–2.

Hindriks, K. V. and Dix, J. (2014). GOAL: A multi-agent programming language applied to an exploration game. In *Agent-Oriented Software Engineering*, pages 235–258. Springer.

Johnson, M., Jonker, C., van Riemsdijk, B., Feltovich, P. J., and Bradshaw, J. M. (2009). *Joint Activity Testbed: Blocks World for Teams (BW4T)*, pages 254–256. Springer Berlin Heidelberg, Berlin, Heidelberg.

Liemhetcharat, S., Yan, R., and Tee, K. P. (2015). Continuous foraging and information gathering in a multi-agent team. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1325–1333.

Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., and Birattari, M. (2013). Task partitioning in a robot swarm: a study on the effect of communication. *Swarm Intelligence*, 7(2):173–199.

Pitonakova, L., Crowder, R., and Bullock, S. (2016). Task allocation in foraging robot swarms: The role of information sharing. In *Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE XV)*, pages 306–313. MIT Press.

Rosenfeld, A., Kaminka, G. A., and Kraus, S. (2006). *A Study of Scalability Properties in Robotic Teams*, pages 27–51. Springer US, Boston, MA.

Wei, C., Hindriks, K. V., and Jonker, C. M. (2014). The role of communication in coordination protocols for cooperative robot teams. In *ICAART (2)*, pages 28–39.

Zedadra, O., Jouandeau, N., Seridi, H., and Fortino, G. (2017). Multi-agent foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling*, 5(1):3.