Memetic Comp. (2015) 7:77–91
DOI 10.1007/s12293-014-0149-3

REGULAR RESEARCH PAPER

Decentralized and dynamic group formation of reconfigurable agents

Ruby L. V. Moritz · Martin Middendorf

Received: 14 April 2014 / Accepted: 15 November 2014 / Published online: 17 December 2014

Abstract Adaptive group formation in dynamic environments performed by heterogeneous swarms of simple agents is an interesting research topic. In this paper we consider an unsupervised scenario where the individuals of the swarm have limited information about their environment as well as limited communication capabilities. The particular case of a multi-agent model with self-organized reconfigurable agents where the agents are confronted with a resource collection task, different movement, and group formation tactics are analyzed experimentally. It is shown that cooperation in groups is profitable for the group members and the optimal group size depends on environmental parameters. Moreover, a simple strategy based on the agents ability to measure their own workload results in an adaptive behavior that influences the size of the groups and increases the performance of the overall system.

Keywords Agent simulation · Group formation · Reconfigurable agents · Dynamic environments

1 Introduction

The organization of self organized decentralized multi-agent systems pose several interesting problems. One problem is the unsupervised group formation within a set of heterogeneous agents. This paper studies a dynamic version of the problem, where the agents have different, reconfigurable capabilities with respect to a resource collection task. To solve this task, the agents can dynamically form groups such that the capabilities of the agents within a group complement each other and the groups become not too large. Here we investigate simple and decentralized strategies for the group formation process of the agents.

The problem to find a partition of a set of agents into groups such that some utility function is maximized is known to be NP-hard with respect to different utility functions (see [1] or [2] for an overview). Therefore, several heuristics have been proposed for this problem. Interesting aspects of the group formation problem are, whether the decisions to form groups can be made decentralized or not and the amount and type of communication required to do so (e.g., [3,4]). To make group decisions, the agents within a group might have to communicate, in particular, when a group contains agents with different capabilities. Hence, the benefit of cooperation within a group usually depends on the cost of communication and the compromises the agents have to make.

Different from other works on group formation of agents or on reconfigurable agents (or robots) we consider a combination of the following two aspects: (i) decentralized group formation of simple moving agents and (ii) agents which can adapt their capabilities by reconfiguration. For the resource collection task it is assumed that different types of resources, e.g. food, construction material, or pollution, are distributed throughout an arena and have to be collected by the agents. The collection of the different types of resources requires different skills from the agents. The agents can move within the arena and are able to form groups with agents they meet. By forming a group the agents benefit from the diversity of their skills when they cooperatively collect the resources. By reconfiguration an agent can

Intended for the thematic issue on ‘Nature Inspired Cooperative Strategies for Optimization’.

R. L. V. Moritz (✉) · M. Middendorf
Parallel Computing and Complex Systems Group Institute of Computer Science, Universität Leipzig, Leipzig, Germany
e-mail: ruby.moritz@informatik.uni-leipzig.de

M. Middendorf
e-mail: middendorf@informatik.uni-leipzig.de

change its skills and therefore can influence its value to the group.

A restricted and static version of our agent system has been investigated before in [5]. That system had a fixed constant number of resource types and the agents were not reconfigurable. In contrast, we study here an agent system with a dynamic environment where the number of available resources changes over time. Moreover, the agents are able to change their capabilities by reconfiguring themselves. The reconfiguration process is based on a simple mutation strategy motivated by naturally occurring mutation processes. It is shown that the agent system can adapt its group formation behavior to the changing environment through the simple reconfiguration process of the agents.

Note, that this paper is an extension of our preliminary paper [6]. In the preliminary version the agent model had only one movement strategy and only one strategy for the recruitment of new agents. In order to investigate the influence of both aspects we consider alternative versions for both strategies here. It is shown that with the alternative recruitment strategy the agents can adapt their group formation behavior to the environment.

In Sect. 2 we describe related work. The multi-agent model is introduced in Sect. 3. Section 4 describes the performed experiments. The results are presented in Sect. 5. A conclusion and an outlook are given in Sect. 6.

2 Related work

Group formation (also called coalition formation) between agents and cooperative learning within groups of agents has been studied extensively in the literature under various aspects. Group formation is particularly relevant when agents have different capabilities and/or different roles. Here, we review studies on multi-agent systems which have some of the following properties in common with our agent system: (i) group formation is a self-organized and decentralized process within the agents, (ii) the agents have only simple local sensing and communication capabilities (e.g., they do not use (complex) negotiation protocols, auction mechanisms, or game playing methods), (iii) the agents only have local knowledge of their environment, (iv) the agents do not use complex strategies to decide how to adapt, learn, or reconfigure (i.e., they do not use strategies based on extensive information evaluation), iv) the agents neither follow different individual interests nor do they cheat, (v) the agents do not form a society with a complex structure (e.g., no hierarchical organization is used), (vi) the agents can move within the environment.

For simple agents with rudimentary abilities to detect environmental signals and no memory of past encounters with other agents it was shown that tag-based collaboration (a tag is a marking, display, or any other observable trait) can lead to the emergence of cooperative strategies among agents [7]. Therefore, several studies have investigated tag-based cooperation between agents. For example, in [8] agents are placed into groups that are identified by tags. In [8], the performance of the multi-agent system with tags was compared to multi-agent systems that use alternative mechanisms for coordinated learning. The system with tags proved to be suitable for coordination tasks, e.g., for cooperation building in competitive scenarios. Another, very recent study, is [9] where it was investigated how agents that are newcomers to a stable host society of agents can adapt both their tag and their strategy to decide which agents to cooperate with (matching strategy). Different payoffs for the matching strategy of the newcomers have been investigated with various combinations of evolutionary and classical learning approaches. It was shown in [9] that the newcomer agents can evolve tag based cooperation strategies that successfully integrates them (measured by a high payoff) into the host agent society. Different from our work, all these works in the literature use some evolution mechanism. The evolutionary mechanism is used to identify the tags and, typically, it is based on the outcome of games that the agents play against each other.

Decentralized strategies for the cooperation of agents that share resources to execute tasks have been investigated in [10]. The resources of an agent have been described, similarly as done in our model for the capabilities (see Sect. 3.1), by a vector $(a_1, \ldots, a_k)$ where a_r, $r \in [1 : k]$, describes the amount of resources of type i of the agent. The agents are connected by a social network and each agent has a set of tasks, each of which has a value and certain resource requirements. The social task allocation problem is to assign to each task the resources of neighbored agents such that the total value of all tasks that could successfully be assigned is maximized. The focus in [10] was to find local auction strategies where each agent uses only local information. The influence of different network structures where the number of agents was between 10 and 120 has been investigated. A similar model that enables agents to form groups for task allocation in a distributed manner, but where not only direct neighbors in the underlying social graph can form a groups, has been studied in [11]. Different from our work, the agents in these works have a fixed graph neighborhood structure on which the group formation is based and the agents are not reconfigurable/adaptable.

There are some studies on multi-agent systems with adaptable agents. A recent survey on adaptation approaches for multi-agent societies has been given in [12] (see also [13,14]). The focus of [12] is on strategies that make agents adaptable so that they can always satisfy the dynamically changing requirements of the system. All the described approaches consider relatively complex systems where the adaptation process is structured into the following four

phases: monitoring, design, selection, and evaluation. One of the more simpler approaches that has been proposed in [15] assumes that there exists a (static) social interaction network between the agents. Each agent has a single, fixed skill (out of a set of possible skills). In order to execute a task the agents have to form a team that corresponds to an induced connected subgraph of the given social interaction network. Each task is characterized by a vector that describes for each skill how many agents are required in the team with the corresponding skill. It is assumed that tasks are arriving dynamically. Different decentralized team forming strategies for systems with 25–100 agents have been investigated experimentally in [15]. Similar to our work, in [16], several strategies (centralized, random, and token based strategies) have been investigated for the adaptation of the relations between agents that can cooperate. A difference to our work is that [16] assumes the existence of a specific adaptable relation network between the agents. It is assumed in [16] that every pair of agents can calculate the utility of the possible relations between both agents and can also decide when to initiate such a calculation and to decide about changing the relation.

Distributed strategies for the group formation problem for a set of heterogeneous agents have been investigated by several authors. In [17], the computational effort of a distributed group formation algorithm was investigated in relation to the quality of the resulting groups. The capabilities of an agent have been described similarly to our model by a vector $(a_1, \ldots, a_k)$ where $a_r, r \in [1 : k]$, describes the agents talent or capability to perform an action of type r. The capability of a whole group has been defined as the sum of the capabilities of all member agents. Improved algorithms for this problem have been presented in [18]. In [19], an auction process was used for coalition formation. The aim was to make local decisions within the coalition to schedule the execution of tasks in order to reduce or avoid global communication between agents.

Coalition formation for moving agents has been studied in [20]. In that system the agents can use global information about all other existing agents, e.g., information about their capabilities. The agents have to execute a hierarchy of differently located tasks, and a task requires several agents with different capabilities for its execution. Therefore, a subset of the agents, which are located closely to a task and together have the required capabilities, can form a coalition, then move to the task, and execute it. A self-organized coalition formation scheme for agents located within an arena has been investigated in [21]. In this work, neighbored agents can form a coalition when it is profitable with respect to a payoff function. It was assumed in this model that a coalition has to pay coordination costs. The influence of different payoff functions and different coordination costs on the size of the coalitions was studied. In [22], a system with agents moving within a 2-dimensional arena in order to form clusters of cooperating agents has been studied. Trial and error is used by the agents to infer which agents to cooperate with and to solve the conflict between achieving a social optimum in the long term or an individual optimum in the short term. Different from our work, the papers cited in this paragraph focus on different payoff functions and decision functions for the agents. A specific application where the agents can form coalitions in order to traverse unsafe areas after a disaster was studied in [23].

3 Design of the model

The design of our model is supposed to be as simple as possible, reducing the number of parameters to a minimum to enable a clear and coherent analysis of the few parameters of interest. The abilities and knowledge of an agent are very limited in order to investigate a general scenario that does not depend on assumptions from specific applications.

The model has an arena F that is a two-dimensional torus of $d \times d$ fields (i.e., the last field in every row and column is adjacent to the corresponding first field in its row, respectively column). $A = \{a_1, \ldots, a_n\}$ is a set of agents where each agent is located on one field of the arena. Each agent can move by stepping to one of the eight fields that are adjacent to its current field. There exists a set $R = \{r_1, \ldots, r_k\}$ of resources of different types. The amount of a resource of type r_i on a field $f \in F$ is given by $f_i \in [0, 1]$. Every agent has the task to collect as many resources as possible.

Before the details of the different parts of the agent model are described in the next subsections, a sketch of the model is given in the following. An agent has different skills to collect the different resources, and two agents can differ in these resource collecting skills. However, the agents can form groups in order to collaborate in collecting resources. The collaboration of two agents would be especially beneficial if one agent is barely able or unable to collect resources of type $r \in R$, whereas the other agent is very skilled in the collection of resource type r. Agents are also able to adapt their skills to the need of their group by reconfiguration. All agents of a group are always located on the same field of the arena.

At each time step at most one group of agents or one single agent can be located on one field. In the following, we consider single agents as groups of size one. Dynamic scenarios are studied in the experiments where the number of available resource types can change.

3.1 Agents

The agents capabilities are described by a simple slot model providing the means of reconfiguration in a rather intuitive way. Such slot models are used by researchers in the area

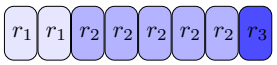

Fig. 1 Configuration of an agent a_i with $s = 8$ slots, the *brightness* indicates the state (1 = light, 2 = medium, 3 = dark), r_i with $i \in \{1, 2, 3\}$ in a *slot* denotes the resource type whose collection this slot enhances, vector (a_{i1}, a_{i2}, a_{i3}) describes the skills of a_i, i.e., a_{ij} denotes the number of slots in state j

Fig. 2 Orientation and location of an agent (*gray dot*) within a part of the arena, the agent is oriented to the field above it (indicated by the *black dot*), numbers show the eight neighbored fields

of reconfigurable hardware (examples are [24,25]). In these models, a slot (also called slice or frame) is the smallest relevant unit of reconfiguration.

In the following, the system parameters are introduced. Each agent has s slots, and each slot is in one of k possible states. The configuration of an agent assigns a state to every slot that can change during a reconfiguration operation. Initially, every slot of an agent is in a state that is chosen randomly with a uniform distribution. Each slot in state $i \in \{1, \ldots, k\}$ increases the agents capability in the collection of resources of type r_i. Figure 1 shows an exemplary configuration for an agent with eight slots in a model with three different types of resources.

Let agent $a_i \in A$ have a_{ij} of its s slots in state j. If agent a_i is located on field $f \in F$ then it can collect an amount $(a_{ij} \cdot f_j)/s$ of resource type r_j per simulation turn. The performance $P(a_i)$ of agent a_i is defined as the total amount of resources the agent can collect in one simulation turn.

$$P(a_i) = \frac{1}{s} \sum_{j=1}^{k} a_{ij} \cdot f_j \tag{1}$$

Note, that with $f_i \in [0, 1], i \in \{1, \ldots, k\}$, and $\sum_{j=1}^{k} a_{ij} = s$ the value of $P(a_i)$ is in $[0 : 1]$.

Each agent has an orientation, i.e., it faces one of the eight fields neighbored to its current location (see Fig. 2). Also, each agent has a battery to store the energy that is needed by the agent to perform its actions. Once the battery is exhausted, the agent is not active any more for a certain time to reload its battery. Thus, an agent a is reloading in a simulation turn, if its battery contains zero energy units (i.e., $a.battery = 0$) or if the agent was reloading in the previous simulation turn (i.e., $a.reload = 1$) and the energy in the battery has not yet been reached the capacity (i.e., $a.battery < capacity$):

$$a.reload := (a.battery = 0) \vee (a.reload \wedge a.battery < capacity)$$

An agent that is reloading adds $capacity/reload\ time$ many energy units per simulation turn to its battery. While an agent reloads the battery it cannot be member of a group and it can neither move nor collect resources. An active agent removes one energy unit from its battery during each simulation turn.

3.2 Groups

To increase its performance, a single agent can join a group or recruit another agent to its own group. The synergistic effect of forming a group is modeled in such a way that small groups potentially benefit more from the recruitment of a new agent than larger groups. A group can access the capabilities of its most skilled agent for the corresponding resource type. Hence, unless the capability of a recruited agent is superior to all other members of the group for at least one resource type, there is no immediate benefit for the group. However, an agent might reconfigure some of its slots to become a profitable member of the group. This is (in principle) always possible when the group has at most k members, where k is the number of resource types. An agent is called specialized for resource type $i \in \{1, \ldots, k\}$ when all its slots are in state i. When there are k agents in a group and each of them is specialized for a different resource type, additional agents cannot not provide any benefit for the group. Equation (2) gives the formal definition of the performance $P(G)$ of a group G.

$$P(G) = \frac{1}{s} \sum_{j=1}^{k} \max_{a_i \in G} a_{ij} \cdot f_j, \qquad P(G) \in [0, k] \tag{2}$$

All members of a group share their location such that the group takes just as much space—one field—in the arena as a single agent. Thus, one field is the area where the group can collect resources during one simulation turn. Accordingly, all agents of the same group move together and are oriented in the same direction. Note, that the physical size of an agent is considered here to be small compared to the size of a field. Therefore, the agent size is considered to be negligible. The assumption that at most one group of agents is located on a field is motivated by the fact that several groups on a field would hinder each other in their work or at least would have to coordinate their work.

When the battery of an agent becomes empty, the agent leaves its group (unless it is already a single agent). Then, the agent starts reloading its battery and stays on its location. Its former group is temporarily allowed to stay on the same field until the group decides to move onto another field. No other agents are allowed to step onto the field of the reloading agent. However, if a new agent has been recruited by a group (from a neighbored field as explained later), the recruited

agent immediately changes its position to the field of the group and adopts the orientation of the group.

3.3 Reconfiguration

An agent has the possibility to change the state of its slots by a reconfiguration operation. Since we assume very simple agents here, the reconfiguration strategy is based on a simple random process called *mutation*. This is motivated by similar mutation processes occurring in nature, e.g., the changes of a DNA sequences. A mutation can occur when an agent has *idle* slots. A slot in state i of an agent a_1 in group G is called *idle* if one of the following conditions holds: (1) $f_i = 0$, (2) there is another agent $a_2 \in G$ with $a_{1i} < a_{2i}$, or (3) there exists another agent $a_3 \in G$ with $a_{1i} = a_{3i}$ and a_3 joined G before a_1.

A slot that becomes idle has probability $\mu \in [0, 1]$ to mutate during that simulation turn where parameter μ is called the *mutation strength*. With each following simulation turn that this slot remains idle its mutation probability increases by μ. Hence, after x idle simulation turns, the mutation probability of the slot is $x \cdot \mu$. The mutation probability of a slot is reset to zero if (1) the slot becomes active (i.e. it is not idle) or (2) the slot mutates.

In case of a mutation, a slot changes its state randomly by choosing the new state from $\{1, \ldots k\}$ with a uniform distribution. In our model we assume that a mutation is not for free but has a fixed cost $c_{\text{reconf}} \in [0, 1]$, where parameter c_{reconf} is called *reconfiguration cost*. The reason for this assumption is that in real applications the reconfiguration operations (e.g., in hardware) take some time and the agent might not able to work normally during this time. Therefore, parameter c_{reconf} reduces the overall amount of collected resources of the agent during the simulation turn at which the mutation took place. The higher the reconfiguration costs are, the lower is the total amount of collected resources. Thus, the performance of an agent a during a simulation turn with mutation is reduced by the factor $(1 - c_{\text{reconf}})$, i.e., its performance is only $(1 - c_{\text{reconf}}) \cdot P(a)$. Note, that the reconfiguration costs are independent of the number of mutated slots during a simulation turn.

3.4 Simulation phases

An experimental simulation run consists of several simulation turns. During each simulation turn three phases occur. At first, during the *recruitment phase*, the groups have the possibility to recruit new members. Afterwards, the *working phase* is scheduled. The final phase is the *battery phase*. The agents are synchronized into these three phases and can only enter the next phase after all agents have completed the previous phase. To avoid inconsistencies, the simulation steps are performed sequentially and the agents perform their operations one after another separately in each phase. For each simulation turn the agents are called in a newly chosen random order. Figure 3 gives an overview of one simulation turn.

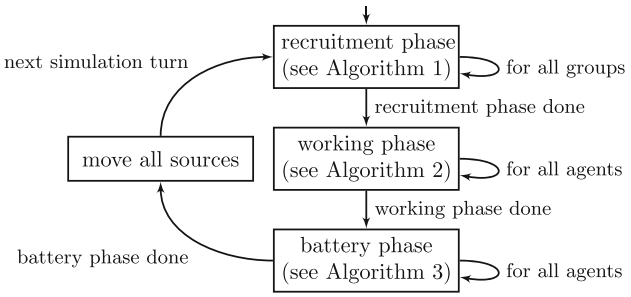

Fig. 3 Flow diagram of the three phases making up a simulation turn

Recruitment Phase. Algorithms 1 and 2 describe the course of action taking place during the recruitment phase. A group can only recruit an agent when (i) the agent belongs to a single agent group, (ii) the agent is located on the field the group is facing, and (iii) the agent is not reloading. Observe, that these rules imply that a group can grow at most linearly in time.

Algorithm 1: Recruitment Phase of the static recruitment version.

```
for each group G do
    f' is the adjacent field in the current orientation
    if there is a group of size one containing only agent a on f' then
        draw a random number φ
        if φ ≤ recruitment rate then
            recruit a to G
```

Algorithm 2: Recruitment Phase of the dynamic recruitment version.

```
1 for each group G do
2     f' is the adjacent field in the current orientation
3     if there is a group of size one containing only agent a_i on f'
      then
4         if there are no idle slots then
5             recruit a_i to G
```

There exist two strategies for recruitment: *static recruitment* and *dynamic recruitment*. Assume in the following that a group G faces a field with a single agent $a \in A$ that is not reloading.

The static recruitment strategy depends on a fixed parameter called *recruitment rate* $\in [0, 1]$. The group G choses a random number $\phi \in [0, 1]$ and recruits a if $\phi <$ *recruitment rate*.

The dynamic recruitment strategy, on the other hand, takes current information about the group into account. The agent

a is recruited if, and only if, none of the slots of the agents in the group *G* is idle. Note, that this strategy is based solely on information which the agents of *G* have already gathered for their reconfiguration processes. If the agents would base their choice on additional information they would require more intelligence, opposing our aim to keep the agents simple. The agents neither count the number of agents in their group nor the number of resource types that are available. However they do get a rough approximation of the relation between both of those values, as larger groups contain agents with idle slots during times of few available resource types, whereas smaller groups typically contain no such agents if many resource types are available.

Working phase. During the working phase (see Algorithms 3, 4) the agents perform several tasks. All reloading agents refill their battery by adding *capacity/reload time* units of energy to it. All active agents suffer a loss of energy (by subtracting one unit of energy from their battery) and perform the tasks as described in the following.

Algorithm 3: Working Phase for the random walk version.

```
1  for each agent a do
2      if a is reloading then
3          reload battery
4      else
5          remove one energy unit from battery;
6          if group G with a ∈ G did not move yet then
7              determine next field f′;
8              draw a random number φ
9              if φ < velocity and f′ is empty then
10                 move G on f′
11             else
12                 if G turned in previous simulation turn then
13                     turn G in previous direction
14                 else
15                     turn G in random direction
16         collect resources and reconfigure idle slots
```

The first step during the working phase is the groups movement. In this paper we analyze two different modes of group movement: *random walk* and *gradient walk*.

Random walking groups make a random decision to either step on the field in their orientation or to remain on their current field to change their orientation by 45°. In the latter case, the group rotates randomly to the left or right, if it has moved in the previous simulation turn, or, otherwise, rotates in the same direction as in the previous simulation turn. Note, that the latter is done to avoid inefficient back and forth rotation. The probability to move forward is defined by the *velocity* parameter. If the *velocity* is low, a group performs more rotations and remains longer on the same field. If the *velocity* is high, a group rotates less and moves forward more often. Thus, the higher the *velocity* is the higher is the average expected number of visited fields per simulation turn.

Algorithm 4: Working Phase for the gradient walk version.

```
1  for each agent a do
2      if a is reloading then
3          reload battery
4      else
5          remove one energy unit from battery;
6          if group G with a ∈ G did not move yet then
7              determine next field f′;
8              if P(G) on f ≤ P(G) on f′ and f′ is empty then
9                  move G on f′
10             else
11                 if G turned in previous simulation turn then
12                     turn G in previous direction
13                 else
14                     turn G in random direction
15         collect resources and reconfigure idle slots
```

The *gradient walk* provides a group with the possibility to intentionally leave undesirable locations and move towards more beneficial areas of the arena, i.e., areas with higher amounts of resources. For this, a group measures its performance on its current field and compares it to the performance it would have on the neighbored field the group faces. If the performance on the neighbored field is higher, the group moves there. Otherwise, the group rotates according to the same rule as introduced for the random movement.

After the group has finished its movement procedure the agents reconfigure and collect resources.

Battery phase. During the battery phase each agent checks the amount of energy in its battery and changes its state (active or reloading) in the following two cases: (i) if the battery is full, i.e., the number of energy units in the battery equals *capacity*), the agent switches into active state, and (ii) if the battery is empty the agent switches into reloading state. See Algorithm 5 as a formal description of this final phase of a simulation turn.

Algorithm 5: Battery Phase: Agents are able to switch their state of activity from reloading to active or vice versa.

```
1  for each agent a do
2      a.reload := (a.battery = 0) ∨ (a.reload ∧ a.battery <
       capacity)
```

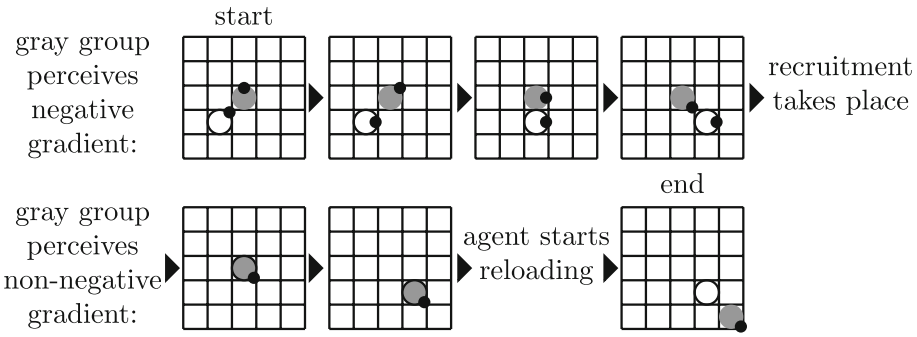

Fig. 4 Exemplary situation of a meeting between a single agent (*white*) and group of more than one agent (*gray*) with recruitment of the single agent and the loss of one agent that starts reloading its battery

Figure 4 depicts an exemplary sequence of simulation turns during which a group recruits one agent and also loses one agent that switched to reloading.

3.5 Resources

To provide the dynamic agents with a dynamic environment and give them an incentive to move during the simulation, the resources are mobile. Here we assume that the resources have a random movement (with respect to velocity and direction) as described in the following. It is assumed that each resource type has exactly one source—the place of highest concentration—that is located within the arena. The location of a source does not always lie in the center of a field and is thus not bound by the otherwise discrete properties of the arena. The location of the source of resource r_i at simulation turn t is denoted by $(r_i.x, r_i.y)(t) = (r_i.x(t), r_i.y(t))$ with $r_i.x(t), r_i.y(t) \in [0, 1]$, i.e., normalized coordinates are used. The initial location $(r_i.x, r_i.y)(0)$ of each resource is a randomly chosen location within the arena. To realize a random movement, in simulation turn $t + 1$ the location of the source of resource type r_i, $i \in \{1, \ldots, k\}$ is relocated to

$$(r_i.x)(t+1) = \begin{cases} r_i.x(t) + \rho_x, & \text{if } r_i.x(t) + \rho_x \in [0, 1] \\ r_i.x(t) + \rho_x - 1, & \text{if } r_i.x(t) + \rho_x > 1 \\ r_i.x(t) + \rho_x + 1, & \text{if } r_i.x.(t) + \rho_x < 0 \end{cases} \tag{3}$$

where $\rho_x \in [-v_{max}, v_{max}]$ is the change of the sources location along the x-axis that is in each simulation turn chosen at random. The *maximum velocity* of resources $v_{max} \in [0, 1]$ is a fixed parameter. The coordinate $r_i.y$ is changed analogously to $r_i.x$ with an independently chosen value ρ_y.

Each resource has an availability radius r_{source} that is identical for all resource types. The farther away a field $f \in F$ is from the source of a resource type r_i the smaller is the available amount f_i of resource type r_i on field f. Let $(f.x, f.y)$ be the center point of f in normalized coordinates then

$$f_i = \max\left(0, 1 - \frac{\Delta(f, r_i)}{r_{source}}\right) \tag{4a}$$

$$\Delta(f, r_i) = \sqrt{\left(\frac{1}{2} - \left||f.x - r_i.x| - \frac{1}{2}\right|\right)^2 + \left(\frac{1}{2} - \left||f.y - r_i.y| - \frac{1}{2}\right|\right)^2} \tag{4b}$$

Note, that the torus arena gives four euclidean distances between two points and that Eq. (4b) determines the minimum euclidean distance between f and the source of r_i. By dividing that distance in Eq. (4a) we either get a value < 1 when the minimum distance is smaller than the radius r_{source} or a value ≥ 1 when the minimum distance is equal to or larger than the radius r_{source}. In the latter case $f_i = 0$, because f's center lies outside of r_i's availability.

4 Experimental settings

To analyze the impact of the different behaviors of the agents, we performed extensive experiments measuring different aspects of the system to infer a good comparison.

4.1 Parameters

Table 1 shows for all model parameters the different values used for the test runs. For the experiments we set standard values for all parameters. A special parameter are the reconfiguration costs for which all three parameter values are considered standard (i.e., each system variant was tested with all three reconfiguration parameter values). Note, that the reconfiguration costs only influence the systems efficiency, but not on the walking or group formation behavior of the agents.

To analyze the robustness of the system with gradient walk and dynamic recruitment we varied the values of several parameters (the system parameters, n, agent velocity, recruitment rate, μ, resource velocity) in additional experiments. Thus, all combinations of parameter values from Table 1 have been tested where all except at most one parameter have standard values. In addition we tested also a system that has random walk and static recruitment with a *recruitment rate* of 0.25 and a *velocity* of 0.6 were used together with standard values for all other parameters.

Table 1 Model parameters applied in experiments

Parameter	Definition	Standard	Extra
System parameter			
–	Movement behavior	Gradient	Random
–	Velocity of agents (only for random)	0.6	0.1, 0.9
–	Recruitment strategy	Dynamic	Static
–	Recruitment rate of agents (only for static)	0.25	0.1, 0.9
Other parameter			
d	Size of arena in fields	50×50	
n	Number of agents	100	50, 500
s	Number of slots	10	
–	Number of simulation turns	4,000	
k	Number of resource types/slot states	10	
–	Capacity of agent batteries in simulation turns	500	
–	Reload time of empty battery in simulation turns	50	
μ	Mutation strength	0.1	0, 0.5
c_{reconf}	Reconfiguration cost	1, 0.5, 0	
v_{max}	Maximal velocity of resources	0.25	0, 1
r_{source}	Radius of resources	1	

The reconfiguration cost is the only parameter with multiple standard values because each system variant has been tested with all three values of reconfiguration costs. If an extra value is used for one parameter in a system, all other parameters have standard values. The only exception is the test of a system with static recruitment and random walk (*recruitment rate* $= 0.25$, *velocity* $= 0.6$, all other parameters had standard values)

Thus, altogether, 3×14 (1 standard values + 2 variations of v_{max} + 2 variations of μ + 2 variations of n + 3 variations of the recruitment rate + 3 variations of the velocity + 1 with random movement and static recruitment rate) experiments have been done, where an experiment corresponds to one combination of parameter values. For each experiment 50 simulation runs have been done. To increase the comparability of the test runs for the different experiments the same random resource movement is repeated in each corresponding run of the different experiments. Thus, all 50 runs of one experiment have different resource movements, but the movement of a run $i \in \{1, 2, \ldots, 50\}$ is identical in all experiments. This ensures that in all experiments the agent system is confronted with the same environmental changes. The only exception are the experiments with different maximal velocities of the sources. Their corresponding test runs, obviously, have a different movement of the resources.

The velocity 0.6 of the random walk agents is chosen such that the mean number of movements per simulation turn is close to that of the gradient walk agents. Similarly, the relation between positive and negative recruitment decisions in static recruitment and dynamic recruitment systems are nearly identical with a recruitment rate of 0.25.

For the radius of the resource types sources the value $r_{source} = 1$ was chosen. Therefore each source is available on every field to some extent. However, in order to study dynamic scenarios not all resource types are available at all times. A source can become inactive for several simulation turns. If the source of resource type r_i, $i \in \{1, \ldots, k\}$, is inactive each field $f \in F$ has an amount of zero of resource r_i, i.e., $f_i = 0$. During the first 1,500 and final 1,000 of the 4,000 simulation turns only two sources of the ten resource types are active. From simulation turn 1,501 until simulation turn 3,000 all ten sources are active.

The mutation strength of $\mu = 0.1$ ensures that an idle slot mutates within ten simulation turns. It also follows that with a probability of approximately 94 % the idle slot mutates within six simulation turns. Moreover, a slot has an expected number of $3.66 = \sum_{x=1}^{10}((1-0.9^x)\cdot x)$ idle simulation turns before it mutates.

The slots of the agents are initialized randomly such that each slot is set to a random state drawn from a uniform distribution. The batteries are initialized, such that all possible active and inactive states are evenly distributed to avoid periodical fluctuations in the number of active and inactive agents. With respect to the number of energy units in the battery there exist 500 active and 50 inactive states. A random number from a uniform distribution is initially chosen for each agent to set it randomly into one of the 550 possible states.

4.2 Measurements

For each variant of the system 50 runs have been performed to infer the special characteristics and differences between the system variants. Several measures—as described in the following—have been recorded and averaged over all runs to infer the general behavior and characteristics of the different systems.

Movement decisions It is measured how often a group rotates either because the target field is occupied, or due to their movement strategy (i.e., because they randomly decide to rotate or measure a negative gradient). This value gives insight into the mobility of the agents in the different systems.

Group size In each simulation turn an agent is either member of a group or reloading its batteries. The size of the group of each agent is measured to infer the impact of different numbers of available resource types on the group formation behavior.

Idle slots The number of idle slots of each agent during a simulation turn shows how beneficial the agents are for their group at that time during the simulation.

Collected resources The average total amount of collected resources per run per agent gives a qualitative measurement of the different systems.

5 Results

The first part of this section gives an analysis of the agent behavior. This includes their movement decisions, their recruiting decisions, and their reconfiguration incentives. In the second part of this section the focus is set to the performance of the system variants considering different reconfiguration costs. In the third part we analyze the impact of different parameters and the robustness of the system against changes in them.

5.1 Behavioral analysis

Four system variants have been investigated. The first two have agents performing a random walk and the latter two have agents that move according to the gradient walk. The first and third variant use the static recruitment, whereas the second and fourth variant use dynamic recruitment.

Movement analysis To analyze the impact of the two different modes of movement—random walk and gradient walk—the frequency of the different movement decisions of all groups were measured. Figure 5 shows the average values over the performed runs for each simulation turn. These values include the current battery state, the presence of a group on the target field, the amount of positive decisions in the random case, and the measured gradient in the gradient walk case. Figure 5 only shows the results for the systems with static recruitment. The respective results for the dynamic recruitment are too similar to see any influence of the recruitment mode on the movement decisions. Agents with the random walk strategy move 0.53 fields per simulation turn on average. This is slightly less then the value 0.6 of their velocity parameter because in some cases the agents are reloading or the groups movement is blocked. The average speed of agents using the gradient walk strategy is 0.51 fields per simulation turn on average and significantly slower (Mann-Whitney test with p-value < 0.01) than with random walk. Recall, that the velocity parameter has been set for a better comparison such that agents in both system variants have a similar average speed. About 10 % of the agents are reloading during a simulation turn and it can be seen that it rarely happens (less than 5 %) that the agents movement is blocked by the presence of other agents on the target field. It can also be seen that for both system variants the increase or decrease in the number of resource types does barely influence the average movement speed. There are no significant changes with random walk and a slight, but significant, difference with gradient walk (increase from 0.506 to 0.514 from simulation turn 1,001 to 1,500 and 2,501 to 3,000 respectively).

Hence, the results show that the 100 agents can move relatively free inside the arena without blocking each other often. This ensures that the movement of the agents is barely affected by congestion.

Group formation In the following we analyze the agents group formation behavior. To this end, the distribution of agents between groups of different sizes is depicted in Fig. 6 for the different methods of movement and recruitment.

The topmost plot shows the total average number of agents in the groups of different sizes in a system with random walking and static recruitment. Initially, all agents are placed individually on random locations into the arena with a random battery state. It can be seen that after about 500 simulation turns the group size distribution went into an equilibrium. About one tenth of the agents are reloading, about one quarter of the agents stay alone (group of size one), over one quarter of the agents stay in groups of size two, and the remaining agents—approximately one third—are in groups that have between three and seven members. The average group size for this system—once in equilibrium—is 1.9 independently of the number of available resource types.

The gradient walking agents with static recruitment show a similar group formation behavior in the second plot of Fig. 6. The average group size of 2.0 is slightly, but significantly (Mann-Whitney test with p-value < 0.01) above the one of the random walking agents. Potentially, this is because the agents in the first system are slightly slower. Slower movement results in a higher chance of meeting other agents in the arena, as slower groups rotate more often and are thus more aware of their surroundings. This increases the number of sights of other groups and situations where a group meets

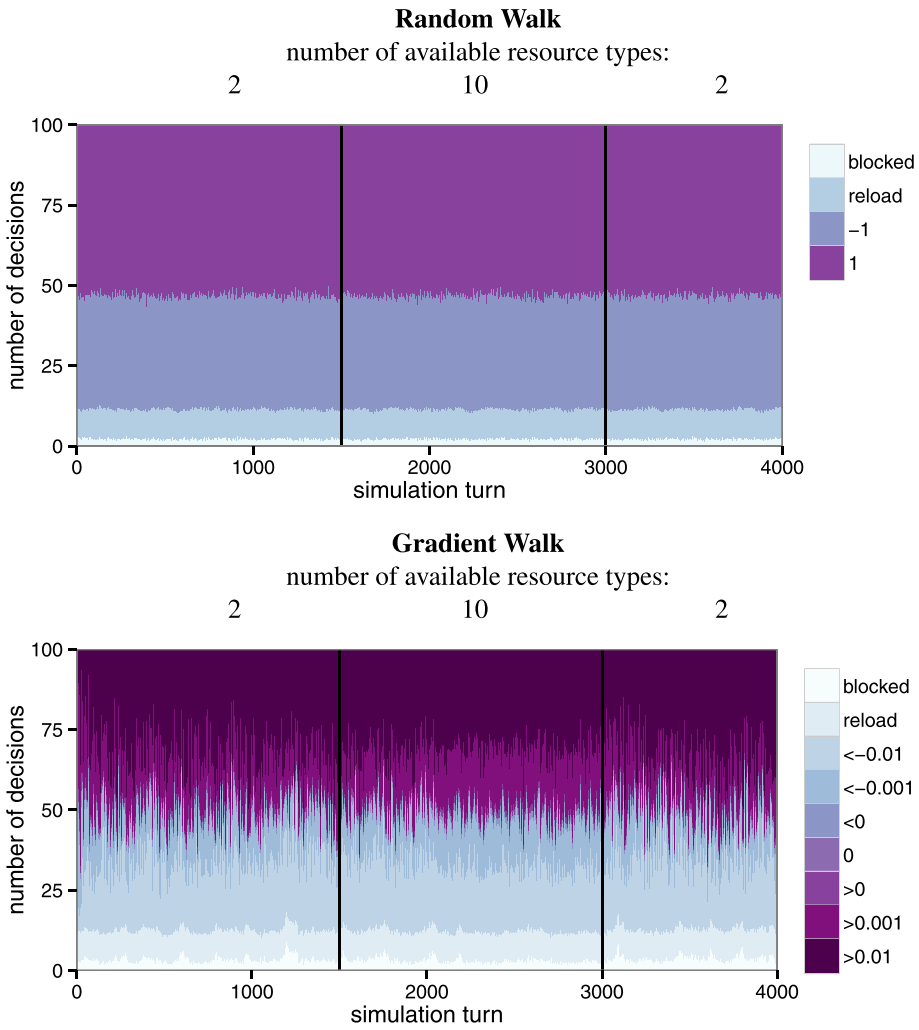

Fig. 5 Movement behavior of a system with static recruitment: random walking groups (*top*), gradient walking groups (*bottom*). Negative numbers indicate a decision to stay and rotate, *positive numbers* indicate a movement of the group. The lower plot gives margins of the measured gradients. Groups oriented towards fields containing other agents are listed as blocked. Reloading agents are listed as such

a single active agent with the possibility of recruiting that agent (number of blocked fields in Fig. 5).

In contrast, the dynamic recruitment shows a clear adaptation of the group size to the changing number of available resource types. Once the number of available resource types increased to ten, the average groups size rises from 2.0 (simulation turns 1,001 to 1,500) to 2.6 (simulation turns 2,001 to 2,500), which is a significant increase (Mann-Whitney test with p-value < 0.01). The dynamic recruiting system has smaller groups on average than the static recruitment systems during simulation turns with only two available resource types. However, the average group size in the dynamic recruitment system is higher than in the static recruitment systems with ten resource types available. In the dynamic recruitment system most agents are in groups of size two or three when there are two resources types to collect. Once the average group size has adapted to the increase of available resource types, over half of the agents are in groups of size 3 or larger. This adaptation takes approximately 250 simulation turns.

Beginning with simulation turn 3,001 the number of active resources is reduced to two again. The dynamic recruiting agents stop recruitment immediately, and the system shows a fast decrease in group size. This reaction overshoots and reduces the group size to a point below the systems steady state. The number of groups with less than three agents decreases in favor of a slightly increasing average group size before the system returns to a steady state.

Idle slots Figure 7 shows the fluctuation in the number of idle slots over the experimental runs with gradient walking agents. During the first 500 simulation turns the system slowly adapts to the two available resources with the random reconfiguration processes. About every tenth agent reloads and has ten idle slots during that time. Another 15 % of the agents have more than seven idle slots, because they are in groups with two other agents that already specialized on the available resource types. Once the number of available resource types is increased to ten, barely any agents have any idle slot (not counting reloading agents).

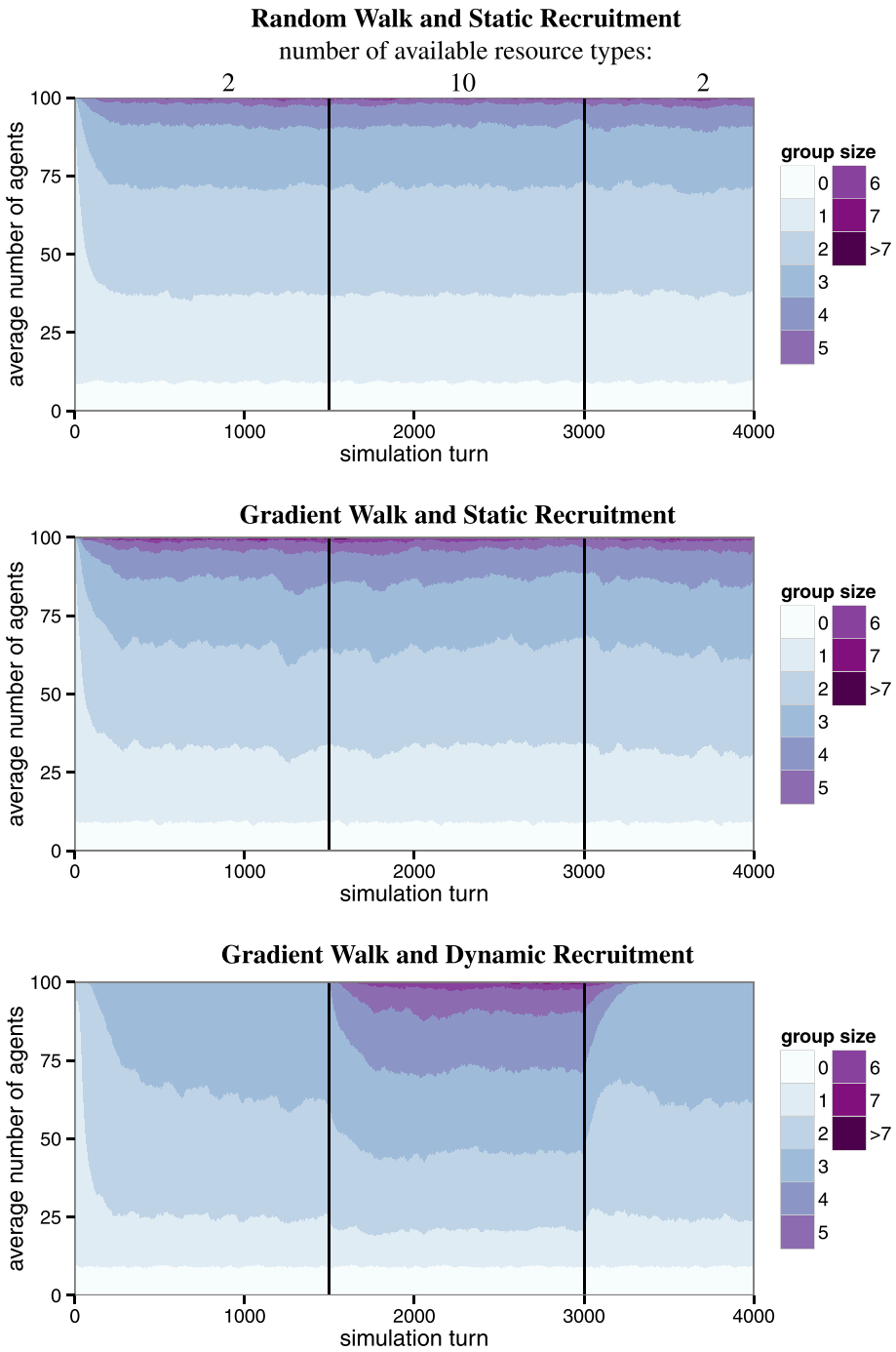

Fig. 6 The total average number of agents in groups of different sizes: random walking groups with static recruitment (*top*), gradient walking groups with static recruitment (*middle*), gradient walking groups with dynamic recruitment (*bottom*). Reloading agents are listed as agents in a group of size zero

Dynamically recruiting agents have an average of 2.5 idle slots when two resource types are available (measured from simulation turn 1,001 to simulation turn 1,500). In the system with static recruitment this number is 2.8, which is significantly higher than for the dynamic recruitment system (p-value < 0.01, Mann-Whitney test). With ten available resource types, the average number of idle slots per agent is as low as 0.99 in the dynamic and 0.97 in the static recruitment system (measured from simulation turn 2,501 to simulation turn 3,000). Though the difference between the averages is small, the number of idle slots is significantly higher (Mann-Whitney test with p-value < 0.01) in the dynamic system than in the static recruitment system. When the number of resource types is switched from ten to two the groups have a large average number of idle slots immediately after the change. However, the agents adapt the group sizes to the number of available resource types and reduce the number of idle slots once again to a lower state than that in the static recruitment system. The static recruitment system is slightly faster in doing so, because the dynamic recruitment system

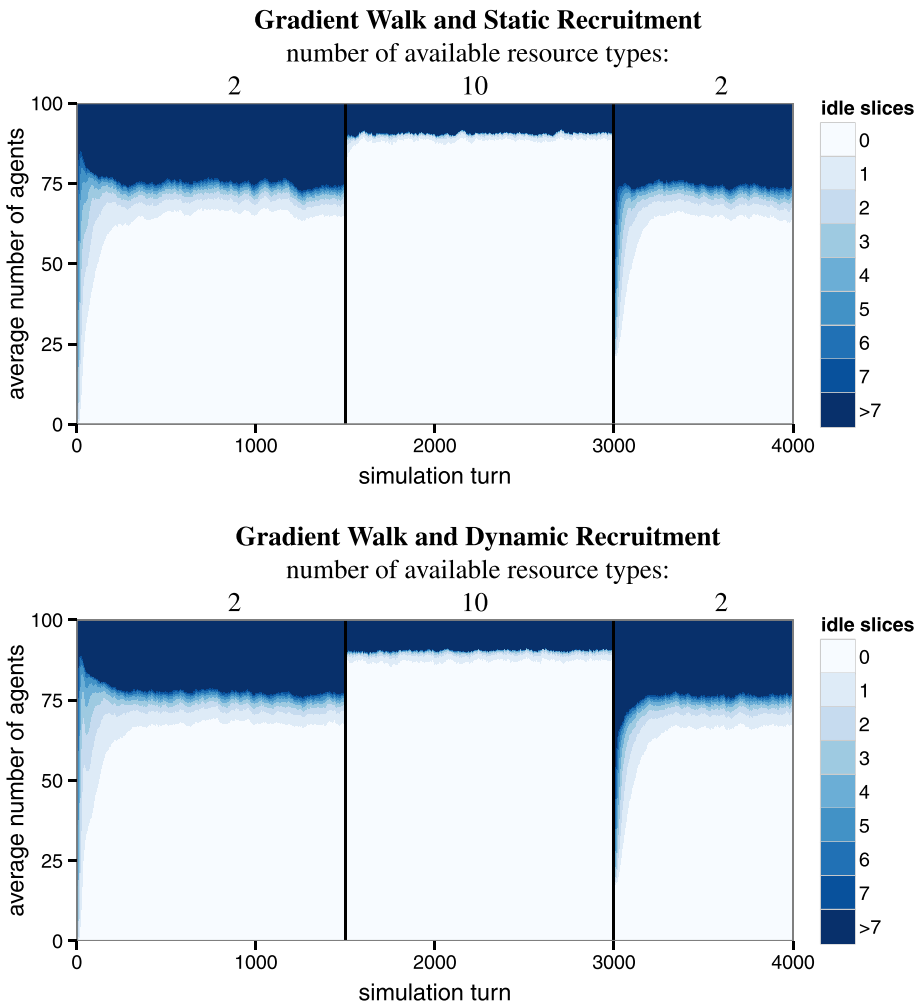

Fig. 7 Average number of agents with the specified number of idle slots: gradient walking agents with static recruitment (*top*), gradient walking agents with dynamic recruitment (*bottom*). Reloading agents are listed as agents with ten idle slots

has a higher average group size at the time of the switch. The larger groups have a higher number of idle slots. To adapt to the change, the agents in the static recruitment system only mutate their configuration into a more fitting state, but the average groups sizes do not change. The dynamic recruiting system simultaneously reduces the size of the groups. As the groups can only decrease their size if a member agent becomes inactive, this process can take a few hundred simulation turns.

5.2 Performance analysis

The performance of a group is determined by the capabilities of its members and the resource availability at their current location. All performance results are significantly different from each other (Friedman test using a Nemenyi post-hoc test with p-value < 0.01), unless stated otherwise. The actual amount of collected resources can decrease if agents have to pay for their reconfiguration. Figure 8 demonstrates how the different systems behave in case of high or no reconfiguration costs. The dynamic recruitment systems (d) and the gradient walk systems (g) perform better than their respective counterpart with static recruitment (s) and random walk (r). Irrespective of the reconfiguration cost, the system with gradient walk paired with dynamic recruitment performs best. However, the improvement achieved by gradient walk is small compared to the positive impact of dynamic recruitment.

5.3 Robustness

The model has several parameters with considerate influence on the agents behavior and performance. In the following we analyze this influence. All performance results are significantly different from each other (Friedman test using a Nemenyi post-hoc test with p-value < 0.01), unless stated otherwise.

5.3.1 Number of agents

Varying the number of agents (n) is similar to changing the size of the arena, as it sets the density of agents per field. A higher density of agents leads to significantly larger groups (pairwise Mann-Whitney tests with p-values < 0.01). The reason is that a higher density of agents leads to a higher

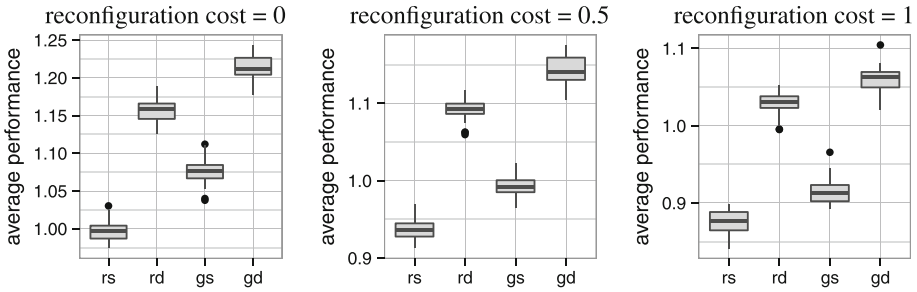

Fig. 8 Performance of the different systems, *r* random walk, *g* gradient walk, *s* static recruitment, *d* dynamic recruitment; average over all simulation turns and agents in one run

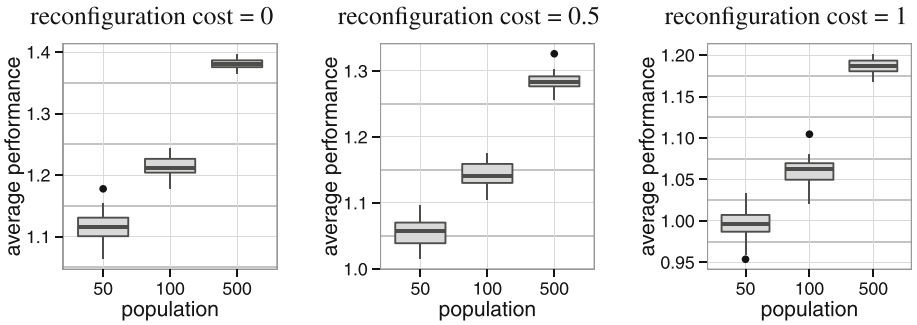

Fig. 9 Average performance of agents in the standard model with 50, 100, and 500 agents. Average over all simulation turns and agents in one run; boxplot for the 50 runs

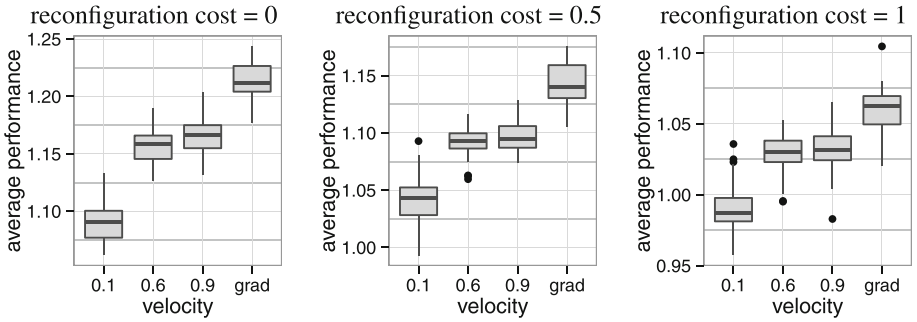

Fig. 10 Average performance of agents in the standard random walk model with different velocity values in comparison to the gradient walk system. Average over all simulation turns and agents in one run; boxplot for the 50 runs

number of collisions between them. This increases the number of possibilities to recruit agents and thus increases the average size of groups in the system. Note, however, that the group size never exceeds three whenever there are only two resource types available.

Figure 9 shows the influence of the number of agents on the systems average performance. The different reconfiguration costs reduce the average performance, but do not change the fact that systems with large populations perform better than systems with smaller populations.

5.3.2 Velocity of the agents

If the agents are unable to measure the gradient between their current and next field, i.e. in static systems, they decide at random whether to move forward or rotate on their current field. How many fields per simulation turn they visit on average is denoted as their velocity. Figure 10 shows the influence of different agent velocities on the systems performance. While higher velocities seem to improve the system, the performance of the gradient walk system has a higher performance then any of the systems with random walk. The performance of systems with a velocity of 0.6 and 0.9 are not significantly different.

5.3.3 Recruitment rate

Higher recruitment rates increase the average size of the groups. In a large group, with at least one specialized agent for each type of resource, every agent can collect the maximum amount of resources. Agents that are not better than all other agents for at least one resource type, constantly reconfigure their slots. However, this is not a problem when reconfiguration is for free. Therefore static recruitment systems with high recruitment rates have a better performance than static systems in case of no reconfiguration costs, because they form large groups.

Large groups are less beneficial with high reconfiguration costs. However, Fig. 11 shows that even with reconfiguration costs of one simulation turn the system with static recruitment still shows a performance increase with increasing recruitment rates. However, systems with dynamic recruitment are the best performing systems in case of high reconfiguration cost. They benefit from their agents ability to adapt their

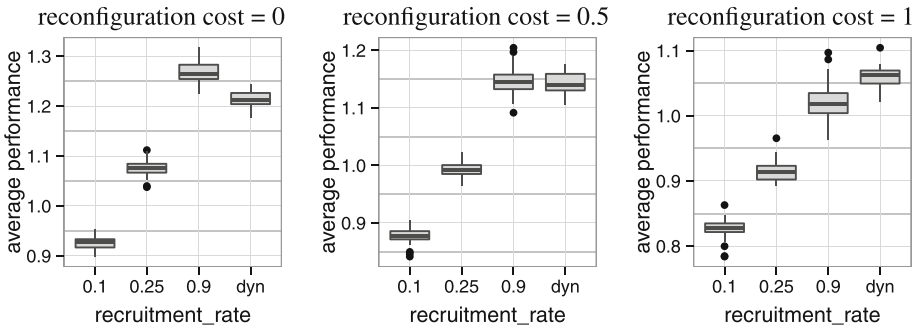

Fig. 11 Average performance of agents in the standard static recruitment model with different recruitment rates in comparison with the dynamic recruitment system. Average over all simulation turns and agents in one run; boxplot for the 50 runs

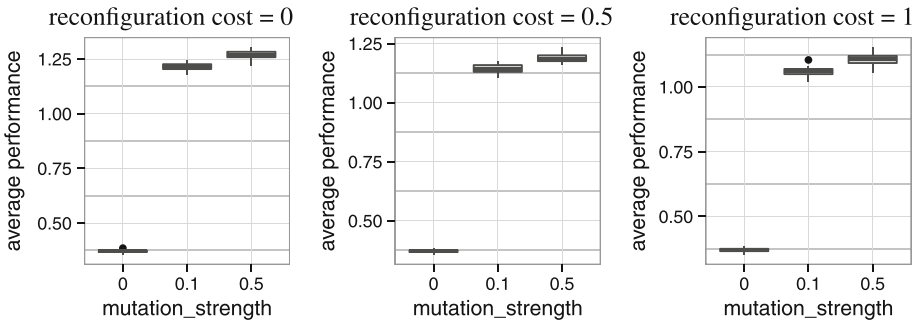

Fig. 12 Average performance of agents in the standard model with different values for the mutation strength. Average over all simulation turns and agents in one run; boxplot for the 50 runs

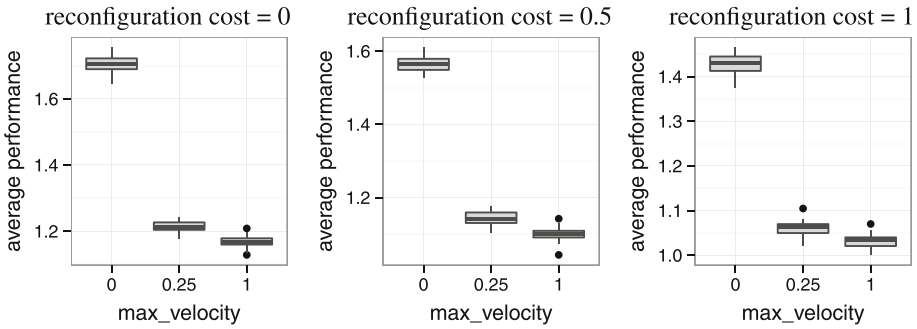

Fig. 13 Average performance of agents in the standard model with different values for the resources maximal velocity. Average over all simulation turns and agents in one run; boxplot for the 50 runs

group size. They form large groups, when many different resources are available and smaller groups, otherwise. With medium reconfiguration cost of 0.5 the performance of the best static recruitment system is not significantly different from the dynamic system.

5.3.4 Mutation strength

A higher mutation strength decreases the time an agent needs to adapt. It also increases the number of reconfiguration events for agents, that can be considered superfluous in their groups, as all other group members already cover all available resource types with their capabilities. Figure 12 shows the influence of the mutation strength on the standard system. The benefit of the fast adaption of the agents is, even with the highest reconfiguration costs, higher than the loss of performance by the superfluous agents in the groups. Increasing reconfiguration costs of several simulation turns will at some point change this phenomenon. Here, the dynamic recruitment increases its performance, if the agents reconfigure and adapt fast.

5.3.5 Maximal velocity of the resources

The velocity of the sources of the resource types influences the performance of the gradient walking system. A slow movement, or even lack thereof, makes it easier for the agents to detect profitable fields and follow the sources. The higher the fluctuation in the environment by faster moving sources, the lower is the performance of the system (see Fig. 13).

6 Conclusion and outlook

If the benefit of cooperation is bound to some environmental and dynamic influence, a dynamic decision tactic to join into a cooperation can be very profitable. In the presented model a set of simple agents was confronted with such a situation. The varying and reconfigurable set of capabilities of the agents provides a very heterogeneous and adaptive swarm of agents. By cooperation through group formation agents benefit from each others capabilities. It is every agents goal to become member of a group that, ideally, consists of one specialist

for each capability that is currently asked for by the dynamic environment.

Our simulations showed that, when agents base their decisions on their current workload, the group formation of the agents is adaptive and increases the overall performance of the system. In a considerably small amount of time the swarm of agents adapts to larger changes in their environment. The impact of a different moving behavior—in this case a gradient walk—has considerably less impact on the performance in such dynamic systems compared to randomly walking agents. However, the gradient walk is more beneficial for the agents than the random walk. Thus, the system with dynamic recruitment and gradient walk performs best, especially if there are reconfiguration costs.

Variations of the values of the model parameters showed that the system with gradient walk and dynamic recruitment is robust and behaves expectedly.

It is left to future work, to investigate whether other simple tactics of the agents can regulate the group formation adaptively. Agents which are rotating often without changing their location gather in larger groups than agents that move faster. This could be an interesting aspect for designing new adaptive group formation systems in the future. Other movement strategies—not based on random decisions or a simple weighted gradient of the multi-objective environment—are also subject of further investigations. It will also be interesting to extent the model by additional spatial aspects, e.g., let agents and groups have specific sizes.

Acknowledgments This work has been supported by the European Social Fund (ESF) and the Free State of Saxony within Nachwuchsforschergruppe „Schwarm-inspirierte Verfahren zur Optimierung, Selbstorganisation und Ressourceneffizienz“. We thank the anonymous reviewers for their comments which helped to improve the paper.

References

1. Gerkey BP, Mataric MJ (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. Int J Robot Res 23:939–954
2. Vig L, Adams JA (2006) Multi-robot coalition formation. IEEE Trans Robot 22:637–649
3. Li C, Sycara K (2004) A stable and efficient scheme for task allocation via agent coalition formation. In: Algorithms for Cooperative Systems, World Scientific. p 20
4. Procaccia AD, Rosenschein JS (2006) The communication complexity of coalition formation among autonomous agents. In: Proceedings of the 5th International Joint Conference on Autonomous agents and multiagent systems. pp 505–512
5. Moritz RL, Middendorf M (2013) Self-organized cooperation between agents that have to solve resource collection tasks. In: Proc. IEEE Swarm Intelligence Symposium. p 12
6. Moritz RL, Middendorf M (2014) Self-adaptable group formation of reconfigurable agents in dynamic environments. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2013). Springer, pp 287–301
7. Riolo MD, Cohen R, Axelrod R (2001) Evolution of cooperation without reciprocity. Nature 414:441–443
8. Chao I, Ardaiz O, Sanguesa R (2008) Tag mechanisms evaluated for coordination in open multi-agent systems. In: Artikis A, OHare GM, Stathis K, Vouros G (eds) Engineering Societies in the Agents World VIII. Volume 4995 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp 254–269
9. Yucel O, Crawford C, Sen S (2014) Evolving effective behaviors to interact with tag-based populations. In: AAMAS-14 Adaptive Learning Agents Workshop, ACM (2014) 7 pages
10. de Weerdt MM, Zhang Y, Klos T (2012) Multiagent task allocation in social networks. Auton Agents Multi-Agent Syst 25(1):46–86
11. Walaa El-A, Hu J, R L (2013) A proposed feasible coalition formation model based on agent weighted graph. J Comput Inf Syst 9:6333–6341
12. Alberola JM, Julian V, Garcia-Fornes A (2014) Challenges for adaptation in agent societies. Knowl Inf Syst 38(1):1–34
13. Mahani MN (2012) Strategic structural reorganization in multiagent systems inspired by social organization theory. Ph.D. thesis, Faculty of Electrical Engineering and Computer Science, University of Kansas
14. Ye D, Zhang M, Sutanto D (2013) Self-adaptation-based dynamic coalition formation in a distributed agent network: a mechanism and a brief survey. IEEE Trans Parallel Distrib Syst 24:1042–1051
15. Gaston ME, DesJardins M (2008) The effect of network structure on dynamic team formation in multi-agent systems. Comput Intel 24(2):122–157
16. Kota R, Gibbins N, Jennings NR (2012) Decentralized approaches for self-adaptation in agent organizations. ACM Trans Autonom Adapt Syst 7(1):28
17. Shehory O, Kraus S (1996) Methods for task allocation via agent coalition formation. Artif Intel 101:165–200
18. Rahwan T (2007) Algorithms for coalition formation in multi agent systems. University of Southhampton, Ph.D. Thesis
19. Kutanoglu E, Wu SD (2007) Coalitions in coordinated multi-agent production scheduling: a computational study. J Manuf Syst 26:12–21
20. Khalouzadeh L, Nematbakesh N, Zamanifar K (2010) A decentralized coalition formation algorithm among homogeneous agents. J Theor Appl Inf Technol 22:35–42
21. Singh VK, Husaini S, Singh A (2010) Self-organizing agent coalitions in distributed multi-agent systems. In: 2010 International Conference on Computational Intelligence and Communication Networks. pp 650–655
22. Abramson M (2008) Coalition formation of cognitive agents. In: Proc. of the Second International Conference on Computational Cultural Dynamics. p 7
23. Bölöni L, Khan MA, Turgut D (2007) Agent-based coalition formation in disaster response applications. Int J Intel Control Syst 12:107–117
24. Ou J, Prasanna VK (2009) Energy efficient hardware-software co-synthesis using reconfigurable hardware. Chapman and Hall/CRC, Boca Raton
25. Rullmann M, Merker R (2011) A cost model for partial dynamic reconfiguration. In: Transactions on high-performance embedded architectures and compilers IV, Springer, pp 370–390