Distributed Multi-Robot Task Allocation Through Vacancy Chains

Torbjørn S. Dahl[a], Maja J Matarić[b] and Gaurav S. Sukhatme[b]

[a] *Department of Computing, University of Wales, Newport, Allt-yr-yn Campus Newport NP20 5XR, United Kingdom*
torbjorn.dahl@newport.ac.uk

[b] *Center for Robotics and Embedded Systems, University of Southern California 941 West 37th Place, Los Angeles, California 90089-0781*
mataric|gaurav@usc.edu

Abstract

Existing task allocation algorithms generally do not consider the effects of task interaction, such as interference, but instead assume that tasks are independent. That assumption is often violated in multi-agent systems, e.g., in the case of cooperative mobile robots, where interaction effects can have a critical impact on performance. Modeling the effects of the interactions within a multi-agent system, the *group dynamics*, is difficult due to their complexity. The same complexity also makes it difficult to program, by hand, optimal solutions to multi-robot task allocation (MRTA) problems. We formalize the concept of group dynamics in the traditional framework of scheduling and show that task allocation in multi-agent systems with significant performance effects from the group dynamics is an $\mathcal{NP}$-complete problem. We then present a simplified model of task allocation in multi-agent systems based on *vacancy chains*. A vacancy chain is a resource distribution process commonly found in nature. Its simplicity and robustness inspired us to use it as a basis for a new model of task allocation with a related MRTA algorithm, both are presented in this article. The new algorithm is sensitive to interaction dynamics without involving the full complexity of the problem. The algorithm uses distributed reinforcement learning to make interference-sensitive estimates of task utilities and relies on stigmergy to produce optimal allocations. We validate our simplified model by demonstrating, in simulation, that the predicted allocations are produced by our algorithm. We also present experimental evidence that the algorithm can handle heterogeneous groups of agents where individual performance levels may differ.

1 Introduction

Existing task allocation algorithms generally assume *task independence*, i.e., they do not consider explicitly synergistic or interference-related effects on performance. Task independence facilitates optimal task allocation, and in some problem domains it is a reasonable assumption to make. For a large number of multi-agent systems, however, it is generally not valid to assume that tasks are independent, e.g., in groups of mobile robots, the effects of interaction among robots working on different tasks, i.e., the *group dynamics*, commonly have a critical impact on performance. Although we use MRTA as a particular problem domain in this article, the results are also relevant to all multi-agent systems where there are significant effects of interaction on performance. This is the case in all domains where there is a limitation on the number of agents that can work on the same task in parallel without interfering with each other, e.g., concurrent accesses to resources over a limited communication network would create interference effects similar to the effects produced by the limitation of available space in multi-robot systems.

Using *scheduling* [7] as a formalism for task allocation, we show that, in domains where the group dynamics have a significant effect on performance, task-allocation is $\mathcal{NP}$-hard. This complexity implies that seeking optimal solutions for on-line systems is infeasible and that we instead need to look for helpful heuristics and practical ways of restricting the problem. To put this work in context we review previous work on MRTA with a particular focus on approaches that use machine learning methods. We also review previous work on modeling group dynamics. As a way of restricting the problem, we present a model of task allocation in multi-agent systems as task allocation through *vacancy chains* (TAVC). The model simplifies the general task allocation problem by breaking down system performance into individual robot contributions. This has allowed us to use the model as a basis for a distributed, communication-free MRTA algorithm. We show that when each robot continuously estimates task utilities locally, solutions reliably emerge which are optimal according to the TAVC model. We demonstrate that the vacancy chain algorithm can improve the performance of a group of robots on a cooperative prioritized transportation problem beyond what is possible with hand-coded solutions. We also demonstrate that the vacancy chain algorithm is sensitive to different performance levels within a group, making it suitable for heterogeneous groups of robots.

2 Motivation

MRTA is a particular instance of the more general problem of task allocation in multi-agent systems. Here we show that task allocation in multi-agent systems is $\mathcal{NP}$-complete when the performance-related effects of the group dynamics are significant.

Gerkey and Matarić [12] presented a taxonomy of MRTA problems over three dimensions: single-task robots vs. multi-task robots, single-robot tasks vs.

multi-robot tasks, and instantaneous assignment vs. time-extended assignment. They also discussed several formalisms for expressing multi-robot task allocation problems. We explore further the scheduling formalism for task allocation and formalize the role of group dynamics. When task allocation in multi-agent systems is formalized as scheduling, the terms *job* and *task*, and the terms *machine* and *agent/robot*, become interchangeable. Heretofore, we will use the terms *job* and *machine* when we relate the task allocation problem to other scheduling problems and the terms *task* and *robot* when we discuss the specifics of MRTA.

2.1 Problem Complexity

In an idealized form, MRTA is the problem of optimizing the allocation of tasks to robots over time, with respect to some given criteria. MRTA can formalized within the existing framework of *scheduling* [7], a well studied problem domain. Defining MRTA problems in terms of a scheduling formalism allows us to consider what classes of algorithms are suitable.

Scheduling problems are described in terms of a set of **machines**, $M_j (j = 1, ..., m)$, that process a set of **jobs**, $J_i (i = 1, ..., n)$. Jobs may consist of a set of **operations**, $O_{i1}, ..., O_{i,n_i}$. If any job can run on any machine, the machines are called **parallel**. Parallel machines are the most general instance of a class called **multi-purpose machines**, (MPM). In MPM, a job can be processed on any machine which has the appropriate tools. If the machines in M_j are used simultaneously, the scheduling problem is called a **multiprocessor task scheduling problem**. In the simplest case where $n_i = 1$, each job, J_i, has a related **processing requirement**, p_i, and a cost function, $f_i(t)$, reflecting the cost of completing J_i at time t. The cost function, f_i, may use a **due date**, d_i, and a **weight**, w_i. Further constraints on precedence, preemptability, and batching of jobs are also common. The machine environment is also subject to a number of further formalizations. Parallel machines, for example, are divided into three classes: **identical** machines, P, **uniform** machines, Q, and **unrelated** machines, R. For identical machines, P, the processing time of a job, J_i, is the same on all machines, M_j, i.e., $p_{ij} = p_i$. For uniform machines, Q, each machine has a related speed, s_j, and the processing time of a job is dependent on the machine, i.e., $p_{ij} = p_i/s_j$. Lastly, for unrelated machines, R, the processing time may be dependent on the machine-job combination, i.e., $p_{ij} = p_i/s_{ij}$.

A common optimization criterion in scheduling is the **weighted total flow time**, denoted $\sum w_i C_i$, where C_i is the **finishing time** of each job, J_i. Other common optimization criteria are defined with respect to other values than the total flow time, e.g., job earliness, job tardiness, or deviation from deadline. Scheduling problems in general can be denoted by three main features: the class of machines, the attributes of the jobs, and the function to be optimized, e.g., $Q \mid p_i = p \mid \sum f_i$, denotes a problem with uniform machines and uniform task processing times, where the value to optimize is the sum of the cost. Some scheduling problems are solvable in polynomial time. Other scheduling problems, such as finding the minimal weighted total flow time for identical

machines, $P \parallel \sum w_i C_i$, have been shown to be $\mathcal{NP}$-hard [7].

2.2 Formalizing the Effects of Interaction

For a given static environment, a group's dynamics depend, to a large degree, on what machines the different jobs are allocated to, i.e., the *allocation*. We formally define an allocation to be a function mapping machines to jobs. For m machines and n jobs, this function can be represented as a vector $\mathcal{A}$ of size m, where vector element i indicates what job is allocated to machine i. We use $\emptyset$ to indicate a machine that has not been allocated a job. Intuitively, an allocation can be seen as a slice of a **schedule** as represented by a Gantt chart [7]. Figure 1 illustrates this for two allocations, $\mathcal{A}_1$ and $\mathcal{A}_2$. The allocations are indicated by the dashed lines through a job-oriented schedule. The corresponding allocations are presented in Equations 1 and 2.

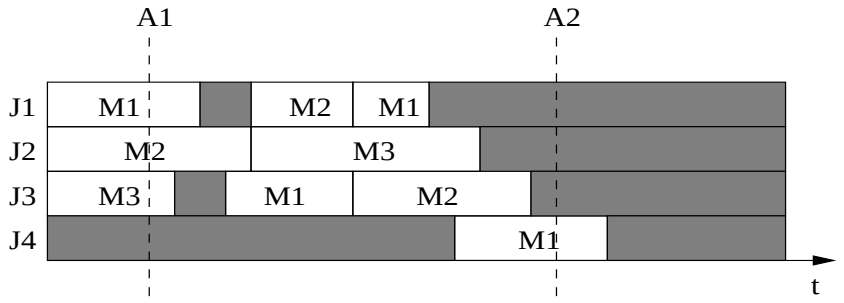

Figure 1: Job-oriented Gantt chart

$$\mathcal{A}_1 = [M1, M2, M3, \emptyset] \tag{1}$$

$$\mathcal{A}_2 = [\emptyset, \emptyset, \emptyset, M1] \tag{2}$$

2.3 Proof of Complexity

We suggest that the effects of group dynamics on job processing times i can be included in the formal scheduling framework by making the job processing time, p_i, from the time the job is started, t_s, to the time the job is finished, t_f, a function of the allocations during this time, $\mathcal{A}_{t_s,t_f}$, as shown in Equation 3.

$$p_i = g_i(\mathcal{A}_{t_s,t_f}) \tag{3}$$

We call the function g the *interaction function*. We call the class of machines whose performance is significantly affected by group dynamics *interaction-dependent machines* and denote the class using the subscript $_{ID}$. The interaction function, as a formalization of group dynamics, captures much of the complexity of group

dynamics in otherwise static environments such as transportation between stationary locations. It is worth mentioning that this is not a suitable formalization of group dynamics in more dynamic problems, e.g., problems where the processing times are also dependent on a changing spatial distribution of jobs. An example of such a problem is cooperative tracking and pursuit [18].

Theorem 1 $R_{ID} \parallel \sum w_i C_i \in \mathcal{NP}$*-hard*

Proof 1 *The optimization problem* $R \parallel \sum w_i C_i$ *describes a set of unrelated machines,* R*, a set of non-preemptable jobs,* J_i*, with related weights,* w_i*, and processing times,* p_i*. The machine speeds,* s_{ij}*, are job dependent. Unrelated machines are a sub-class of interaction-dependent machines in that the function describing the task processing time,* $p_{ij} = p_i/s_{ij}$*, is a deformed interaction function that is only dependent on the task and machine in question and not on what tasks are allocated to the other machines. We can create, in polynomial time, a corresponding problem* $R_{ID} \parallel \sum w_i C_i$ *with an identical set of machines,* R_{ID}*, an identical set of weighted jobs, but with task processing times described by the interaction function* $g_i(\mathcal{A}_{t_s,t_f})$*.*

The assumption that the machines in the original problem are unrelated, but not interaction-dependent, implies that the value of $g_i(\mathcal{A}_{t_s,t_f})$ *is only dependent on which machine* M_j*, the related job,* J_i*, is allocated to in the relevant allocations* $\mathcal{A}_{t_s,t_f}$*. For non-preemptable jobs,* M_j *will be the same for all those allocations. The interaction functions* $g_i(\mathcal{A}_{t_s,t_f})$ *can be produced in polynomial time, by finding which machine job,* J_i*, is allocated to in the initial allocation,* $\mathcal{A}_{t_s}$*. The job,* J_i*, and machine* M_j*, can then be used to find the machine speed using the given function,* p_i/s_{ij}*.*

An algorithm solving $R \parallel \sum w_i C_i$ *problems uses sets of machines, sets of jobs, and optimization criteria that are identical to the ones used by an algorithm for solving the corresponding* $S \parallel \sum w_i C_i$ *problems. The only difference between the problems is the representation of the function for job processing times. We have shown how the function* p_i/s_{ij} *can be accurately represented in the more general form* $g_i(\mathcal{A}_{t_s,t_f})$*, and that the transformation can be done in polynomial time. Hence, an optimal solution to the new problem would be an optimal solution to the original problem. This again shows that* $R \parallel \sum w_i C_i$*, which is* $\mathcal{NP} - hard$*, reduces to* $R_{ID} \parallel \sum w_i C_i$ *which,* ipso facto*, must also be* $\mathcal{NP}$*-hard.* □

3 Related Work

This work touches on two important larger areas of research; task-allocation and the modeling of group dynamics. In this section we review previous work in these two areas and relate it to the work presented in this article.

3.1 Multi-Robot Task Allocation

A number of algorithms for MRTA already exist. Here we review a selection of the most prominent of these algorithms and discuss how their different approaches to group dynamics affect their applicability in domains where the effects these dynamics have a significant impact on group performance.

Botelho and Alami's M+ algorithm [4] used a task allocation protocol based on the Contract Net protocol with formalized capabilities and task costs. The need to pre-define the capabilities and costs limits the applicability of the M+ algorithm to domains where these are known. It might, however, be possible to use the M+ algorithm with local utility estimates based on ML rather then hand-coded capabilities and costs.

Gerkey and Matarić's MURDOCH system [11], also based on the contract net protocol, used a set of metrics to locally score the suitability of the participating robots, a publish/subscribe protocol for communication, and an auction mechanism for task allocation. As with the M+ algorithm, the hand-coded suitability estimates have limited applicability, but the MURDOCH system can also use local task utility estimates based on ML.

Werger and Matarić's work on the Broadcast of Local Eligibility (BLE) algorithm for task allocation compares locally decided eligibilities to allocate tasks using Port Arbitrated Behaviors [32], an inter-robot coordination mechanism based on a fully connected communication network. Their example of Cooperative Multi-Robot Observation of Multiple Moving Targets used spatial proximity as a measure of eligibility, but BLE can also use eligibility estimates based on ML.

In the L-ALLIANCE work by Parker [28], each robot explicitly estimates its own performance and the performance of other robots on selected tasks and uses these values to reallocate tasks by taking them over or acquiescing. The L-ALLIANCE algorithm uses local utility estimates to make local allocation decision, but needs pre-programmed estimation procedures that reduce the general applicability of this algorithm.

We are not aware of any existing general MRTA algorithms that explicitly handle the effects of group dynamics. Consequently, the TAVC algorithm represents an alternative for domains where the effects of these dynamics have a significant impact on system performance.

Learning Approaches to Task Allocation One way of dealing with $\mathcal{NP}$-hard problems is to use heuristic algorithms that produce suboptimal but satisfactory solutions. Learning such heuristics in the domain of scheduling is a well-studied problem. Here we review a selection of successful uses of machine learning (ML) as a way of finding heuristics for scheduling. We also discuss the applicability of these algorithms in domain of MRTA.

Zhang and Dietterich [34] presented a RL approach to scheduling that learned domain specific heuristics for the scheduling procedure. The state space consisted of possible schedules and actions were possible changes to the schedules. The system learned what changes would quickly create feasible schedules with

maximized capacity utilization. The problem domain considered was space shuttle payload processing.

Zomaya *et al.* [35] presented another algorithm for learning scheduling heuristics. Their algorithm learned dynamic scheduling, i.e., scheduling when there is no *a priori* knowledge about the tasks. It used a back-propagation neural network and a history queue that functions like an eligibility trace to learn how to associate a set of job parameters with a set of machines.

Both above algorithm are general and applicable to all scheduling problems. However, they rely on a centralized learning mechanism, which can fail critically due to robot breakdown and also does not scale easily to large groups.

Brauer and Weiß [6] used the multi-agent learning paradigm to distribute the learning of heuristics for scheduling a set of multi-operation jobs over a set of machines. The operations were totally ordered and each operation could only be performed by a subsets of machines. Using RL, each machine estimated the efficiency of the possible predecessors and together the machines learned to improve the total production rate from an initial state where all machines were assumed to be equally efficient. Brauer and Weiß also showed that the learning allowed the system to adapt to machine breakdowns. This algorithm, however, is restricted to completely ordered jobs and this limits its applicability to general scheduling and task allocation problems.

Blum and Sampels [3] studied different pheromone representations in ant colony optimization of first order job shop scheduling. Each job traversed a set of machines by stochastically selecting machines in accordance with a given set of constraints on the job's operations. Throughout the traversal, the system updated a pheromone trace, effectively learning environmentally embedded heuristics for scheduling. The different pheromone representations allowed the system to estimate utilities for different combinations of machines, including estimating the utility of a machine in isolation and the utility of a machine depending on the last machine visited. This algorithm, however, is limited to jobs that consist of multiple operations. It is difficult to see how a similar pheromone trail could be used successfully as a general scheduling algorithm.

Tangamchit *et al.* [31] used distributed RL on a set of robots to allocate patrolling tasks. Each robot locally estimated the utility of a set of patrolling points and together learned to divide the set of point between them in an optimal manner. Tangamchit *et al.* also used local and global proximity measures as heuristics for action selection in order to speed up learning. The specificity of the spatial heuristics limits the applicability of this algorithm to general task allocation problems.

The work reviewed above demonstrates the use of ML techniques to learn successful heuristics for scheduling. The heuristics can be centralized, environmentally embedded, or distributed. However, the domains considered in all of the work reviewed above, except for the work by Tangamchit *et al.*, do not have significant effects from group dynamics. In spite of this, all the learning algorithms reviewed above would likely improve a group's performance in the presence of significant effects from group dynamics, as they in general learn where to allocate jobs according to feedback based on processing time. The

advantage of our vacancy chain algorithm over existing learning algorithms for scheduling is the combination of its general applicability and its distributed learning mechanism.

Finally, since our initial presentation of the TAVC algorithm [10], Low *et al.* [22] have presented a similar self-organizing algorithm for ant-based MRTA. The approach taken by Low *et al.*, like ours, uses a locally estimated interference indicator which they call encounter patterns. Their self-organization also works on the principle of stigmergy. The results presented by Low *et al.* support our claims we make about the reliability and dynamic properties of such algorithms. Similar results have also been reported by Labella *et al.* [19] using a swarm-based robot control.

3.2 Modeling Group Dynamics

The interaction function we defined in Section 2.1 describes the effects of all possible allocations on individual job processing times. Such a description presupposes that we can model these effects in a way that lets us identify the relevant times. This presumption, however, is too optimistic. In this section we review previous work on modeling the effects of interaction and discuss how our model fits into the traditional taxonomy.

In groups of mobile robots, we term the internal interactions between members of the group the *group dynamics*. The group dynamics can help explain group-level behavioral features created by the presence of multiple robots, e.g., interference, when the robots have to spend time avoiding each other rather than following the paths that would be optimal in single-robot systems. The presence of multiple robots very commonly has a negative effect on performance in terms of interference. Cooperation, on the other hand, produces group dynamics that have a positive effect on performance. When group dynamics have a major effect of performance, understanding and modeling these dynamics is necessary in order to predict the value of different schedules. In embodied systems such as mobile robots, there is significant uncertainty related to interaction with the real world in terms of accuracy of perception and effects of actions. A model of the dynamics within a group of mobile robots is a model of several interacting systems, each with a significant level of inherent unpredictability. This limits the possible accuracy of any formal model.

Models of group dynamics have been divided into *microscopic*, which may or may not be *simulation-based*, and *macroscopic* [20]. Microscopic models explicitly represent each agent. Simulation-based models simulate the actions taken by each of the agents so that properties of the system can be recorded as the agents interact. Macroscopic models directly describe properties to systems on the basis of abstract features such as the number and general distribution of agents. Below we review the most relevant work on producing and using explicit models of group dynamics to improve group performance.

Microscopic Models Game theory [26] is an abstract model of multiple interacting agents as players in formalized n-person stochastic games. Game theory

traditionally represents each player in terms of his/her *strategy* for playing a given game and its related *payoff matrix* for the possible moves. On the basis of these factors, game theory describes some of the resulting features of the ensuing games, e.g., their optimality and stability. A game theoretic approach to modeling group dynamics requires that we know the relevant payoff matrices. Unfortunately, this information is not known in most non-trivial multi-robot systems. To overcome unknown payoff matrices, Littman [21] presented the Minimax-Q algorithm which is guaranteed to find the equilibrium of any stochastic game. As a general payoff policy for n tasks and m robots is of size n^m, it is not feasible in general to estimate the payoffs for each state/action combination. Bowling *et al.* [5] presented experimental data demonstrating how a reinforcement learning (RL) algorithm based on policy gradient ascent and the WoLF (Win or Learn Fast) principle for adjusting the learning rate can overcome this complexity. We discuss our approach to MRTA in terms of game theory in more detail in Section 7.

Simulation-Based Models Goldberg and Matarić [14] developed Augmented Markov Models, transition probability matrices with additional temporal information, to learn statistical models of interaction in a space of abstract behaviors. They also used these models to maximize reward in a multi-robot mine collection task.

Balch *et al.* [2] studied live insect colonies and constructed three-dimensional histograms of insect presence over a discretized area. The work was a step toward a long term goal of combining spatio-temporal models with Behavior Hidden Markov Models for behavior recognition [16] in order to recognize colony behaviors.

Yan and Matarić [33] have attempted multi-level modeling of group behaviors from spatial data describing both human and robot activity. Like Balch *et al.*, they used three-dimensional histograms to recognize and describe different activity distributions as produced by underlying behaviors.

Seth [29] pointed out the distinction in biology between 'phenomenological' and 'mechanistic' models of interference, where the former identifies mathematical relationships between intake rates and agent density in empirical data while the latter constructs individual-based models with pre-specified rules for agent behavior. This distinction corresponds closely to that between macroscopic and simulation-based models used in robotics. Mechanistic models allow a derivation of the interaction between agent density and rule application with respect to intake rates. The phenomenological models assume that agents always optimize their individual intake rates. They also assume unstructured environments. These assumptions are often invalid both in biological systems and in the multi-robot task allocation domain. As an alternative to the traditional biological models, Seth presents a simulation-based model using genetic algorithms to evolve foraging behaviors for multiple agents in spatially explicit environments. The evolved systems are able to reproduce interference functions previously described in field studies of insects, but not reproduced by

phenomenological models.

Macroscopic Models Matarić [23] proposed a macroscopic model of interference as an estimate based on *group density*. Group density is defined as the ratio between the agents' *footprints* and the available *interaction space*. An agent's footprint is its sphere of influence, including factors such as geometry, motion constraints, and sensor range. If only the number and size of the agents are used, a *mean free path* can be computed and used to estimate the *number of expected collisions* for agents executing random walks.

Lerman *et al.* [20] studied three different types of models of cooperation and interference in groups of robots: sensor-based simulations, microscopic numerical models, and macroscopic numerical models. They used differential equations to model the group dynamics on a macroscopic level and showed that the three different models produced corresponding results. Unlike the sensor-based simulation and the microscopic numerical model, the macroscopic model had the advantage of being very fast and independent of the number of robots modeled. To make a macroscopic model tractable, however, many simplifying assumptions were necessary.

Macroscopic or phenomenological models are not generally sophisticated enough for optimizing specific task allocation algorithms. In particular, Matarić's model for estimating interference does not consider synergistic effects or environmental complexity. The differential equation models produced by Lerman *et al.* similarly assume uniform distributions of robots and tasks. The simulation-based models or mechanistic models produce problem-specific solutions, but the time needed to construct those renders them unsuitable for use in MRTA algorithms. Currently there are no models of the effects of group dynamics with the speed, generality, and predictive accuracy necessary for specifying the effects of interaction on task processing times in MRTA problems with the aim of constructing optimal schedules. Simulation-based models are in general too slow while macroscopic mathematical models make too many simplifying assumptions to be of predictive use.

4 Task Allocation through Vacancy Chains

To address the two main problems of task allocation in multi-agent systems where the effects of the group dynamics are significant; problem complexity, as discussed in Section 2.1, and the modeling of group dynamics, as discussed in Section 3.2, we developed the TAVC model of task allocation. We used the model TAVC model to develop the TAVC algorithm for MRTA.

4.1 Vacancy Chains

The inspiration for our adaptive task allocation algorithm is the *vacancy chain* process [8], through which resources are distributed to consumers. The typical

example is a bureaucracy where the retirement of a senior employee creates a vacancy that is filled by a less senior employee. This promotion, in turn, creates a second vacancy to be filled, and so on. The vacancies form a chain linked by the promotions. The resources distributed in this example are the positions and the consumers are the employees.

The general process of distributing resources in this way has been recognized in many different domains. It was originally reported in human populations relating to houses and apartments as well as to jobs in bureaucracies. Chase [8] proposed that major human consumer good such as cars, boats, and airplanes, also move through vacancy chains and that vacancy chains are common in other species such as the hermit crab, the octopus and different species of birds. In the case of the hermit crab, the empty gastropod shells they carry around as portable shelters are distributed through vacancy chains.

Chase lists three requirements for resource distribution through vacancy chains:

1. The resource must be reusable, discrete, and used by only one individual.
2. A vacancy is required before an individual takes a new resource unit, and individuals must need or desire new units periodically.
3. Vacant resource units must be scarce, and many individuals must occupy suboptimal units.

The vacancy chain resource distribution mechanism is both simple and powerful. It is based on *stigmergy*, unintentional communication between the consumers through their effects on the environment [17]. This makes distribution through vacancy chains robust and efficient in large groups/societies.

While distribution through vacancy chains ensures that the most attractive resources are consumed, it does not, like more sophisticated resource distribution mechanisms, take into account the quality of the consumer. As such, the vacancy chain distribution process can not exploit the possible advantages of distributing particular resources to particular consumers. The vacancy chain distribution mechanism treats all consumers as equals. In terms of task allocation, the vacancy chain distribution algorithm does not guarantee optimal allocations as it does not consider possible differences in machine speed. Our initial study of vacancy chain distribution reflects this fact by considering only homogeneous groups of robots.

4.2 The Vacancy Chain Model

Inspired by the vacancy chain process we developed a formal model describing how system performance is influenced by the allocation of tasks to machines for spatially classifiable tasks. According to the TAVC model, any number of robots can be assigned to tasks from a given class. When a j'th robot is assigned to a task from a class, i, we say that *service-slot* (i, j) is filled. The service-slots are

the resources that are distributed among the individuals. The individuals are the robots.

A particular number of homogeneous robots, j, servicing the same class of tasks, i, will have a task processing frequency, $c_{i,j}$, dependent on the degree to which the robots are able to work concurrently without interfering with each other. The difference in task processing frequency, together with the task value, w_i, define the contribution of the last robot added, or the last service-slot filled, to the total system performance. We call this contribution, which can be negative, the *slot-value*, $s_{i,j}$. The formal definition is given in Equation 4.

$$s_{i,j} = w_i(c_{i,j} - c_{i,j-1}) \tag{4}$$

When assigning an additional robot to a task leads to a decrease in the task processing frequency, the slot-value correspondingly becomes negative. When all the available service-slots have negative values, we say the task is *saturated*. If all the tasks are saturated, the system is saturated.

According to the TAVC model, an allocation is optimal when it maximizes the sum of the filled service slots. When the slot values decrease monotonically, optimizing the group performance becomes identical to optimizing the value of the individual slots. With spatially classifiable tasks and given the relevant slot-values, a group of robots can optimize the value of completed tasks over time, i.e., the *throughput*, by allocating robots to tasks or service-slots in order of decreasing slot-value. Note that this problem does not have the greedy-choice property as slot-value does not correspond to task-value and system performance is not necessarily optimized by allocating tasks from the class with the highest related value to all the robots. The total system performance or throughput value, T, for n robots, is simply the sum of the individual contributions as stated in Equation 5.

$$T_n = \sum_i \sum_j s_{i,j} \tag{5}$$

In these kinds of problems, the task allocation algorithm can be distributed by letting each robot optimize the value of the service-slot it occupies.

In a scenario where the service-slots are allocated optimally, a failure in a robot will result in an empty service-slot. If the value of the vacant slot is greater than the value of one of the other occupied service-slots, the vacant slot will have to be filled in order to restore optimal allocation. Expressed in vacancy chain terminology, a vacant service-slot is a resource to be distributed between the robots.

4.3 The Complexity of TAVC

TAVC reduces the complexity of the general MRTA problem by introduces several simplifications. The most important simplification is the restriction that jobs are *spatially classifiable*. This property is formalized below. Initially we also look only at homogeneous robots, but in Section 5.4 we relax this restriction

and report experimental results on TAVC in groups of heterogeneous robots. Both these simplification facilitates finding good solutions using distributed RL without simplifying the problem complexity.

Spatially Classifiable Jobs Such jobs can be divided into a set of classes, K_k, where the interactions between machines working on tasks in different classes have no significant effects on the groups' performance. In terms of scheduling, we denote spatially classifiable jobs with *sc*. The problem of optimizing the weighted total flow time in a system with interaction-dependent machines and spatially classifiable jobs is denoted $R_{ID} \mid sc \mid \sum w_i C_i$. One example of such problems is when classes of jobs take place in spatially separate areas. For such jobs, J_i, the processing time is a function of the effects of the interaction between the machines currently allocated to jobs in that class, $\mathcal{A}_{t_s,t_f,k}$, as expressed in Equation 6.

$$p_i = g_i(\mathcal{A}_{t_s,t_f,k}) \tag{6}$$

The allocation of robots between jobs of other classes is not relevant. By the same reasoning as with general interaction-dependent jobs, it can be shown that scheduling spatially classifiable jobs over interaction-dependent machines is an $\mathcal{NP}$-hard problem.

Homogeneous Robots Within the class of interaction-dependent machines processing spatially classifiable jobs, we restrict ourselves further to looking at problems where the machines are *identical* or *homogeneous*. In this case, the interaction function is dependent only on the *number* of robots, m_k, currently working on jobs in each class, K_k. The job processing time for this class is given in Equation 7.

$$p_i = g_i(m_k) \tag{7}$$

The problem of scheduling with two or more identical machines, $P2 \parallel \sum w_i C_i$, is known to be $\mathcal{NP}$-hard [7]. In this problem, the job processing time, p_i, is only dependent on the job, J_i. By the same reasoning as with unrelated interaction-dependent machines, it can be shown that $P2 \parallel \sum w_i C_i$ reduces to $P_{ID}2 \parallel \sum w_i C_i$, i.e., scheduling spatially classifiable jobs over two or more identical interaction-dependent machines. Hence, $P_{ID}2 \parallel \sum w_i C_i$ is also $\mathcal{NP}$-hard.

4.4 The TAVC Algorithm

Reinforcement learning (RL) provides a way to improve on hand-coded solutions. Such improvements are particularly effective for problems where it is hard for human programmers to identify optimal solutions. One such problem is MRTA, where, as discussed in Section 3.2, the group dynamics make it difficult to identify optimal allocations. We have previously studied the use of distributed RL as a way of optimizing the performance of multi-robot systems

in an interaction-sensitive manner [9]. Our TAVC algorithm [10] is a specialization of that work for MRTA. In the TAVC algorithm, an action, as known from the RL literature, corresponds to a task, as known from the MRTA literature. Each robot keeps local estimates of task utilities in form of Q-values, and choose its tasks using an ϵ-greedy action selection function.

Q-learning is not sensitive to the frequency of rewards. Hence, the estimated values of actions do not necessarily correspond to their contribution to performance over time. In order to use Q-learning to optimize performance over time, it is necessary to make the temporal aspect of performance explicit in the reward function. Such a reward function, using the last task processing time, t, and task value, w_i, is given in Equation 8.

$$r = w_i/t \tag{8}$$

This reward function promotes the actions with the highest contributions to the system performance because these will on average provide a higher reward. If a robot consistently occupies a service-slot that is suboptimal due to too much interference, the increased average traversal time will reduce the average reward for that slot below the average reward of the optimal service-slots. This change in average reward will attract the robot back to an optimal slot. Equation 9 clarifies the relationship between the definitions above by expressing the relationship between the individual throughput, T_i, the weighted differences in task completion frequencies, the weighted average task processing times and the average rewards, for j robots servicing a given task, i.

$$T_i = \sum_j s_{i,j} = w_i \sum_j (c_{i,j} - c_{i,j-1}) = w_i \sum_j \frac{1}{\bar{t}_{i,j}} = \sum_j \overline{r}_{i,j} \tag{9}$$

4.5 Controller Architecture

All the robots in our demonstration used the same behavior-based controller [24]. Importantly, however, our TAVC algorithm is independent of the underlying architecture, being defined purely in terms of distributed RL over problem states and task utilities. Based on individual experience, the robots learned to specialize but they always retained a level of exploration allowing them to find and fill new vacancies.

The State/Action Space Each controller in our experiments had a set of pre-programmed high-level behaviors. Each behavior corresponded to servicing one of the available tasks and consisted of multiple shared lower level behaviors. The low-level behaviors were as follows:

- **obstacle avoidance:** avoided obstacles detected (by laser range finder) in the desired path.
- **visible target approach:** approached the desired target when it was visible (to the laser range finder).

- **target location approach:** approached the location of the desired target when the target itself was not visible.
- **wall following:** followed the walls to the first available target when the desired target was not visible and localization (based on odometry) was deemed to be inaccurate.

The localization was deemed to be inaccurate whenever the desired target was not visible, but should have been so according to the robot's estimated position and orientation. On encountering a target, the localization estimate was reset and again deemed to be accurate.

Individual Learning The robots were homogeneous with respect to hardware configuration and control algorithms. Each robot used RL to associate the currently observed problem state with the available actions, i.e., task-related behaviors. Over time, the RL differentiated the group by specializing robots on different tasks. The robots used temporal difference Q-learning [30] to associate the different states with one of the high-level task-related behaviors. The Q-tables ware initialized with random values between -0.1 and 0.1, the learning rate α was set to 0.1, and the discount factor γ was set to 0.95.

The input- or state-space reflected which circuit the robot had used for its last traversal. This allowed the robots to learn policies that were not dedicated to one circuit. The learned policies could switch between circuits in order to construct optimal task sequences. In spite of this, the robots consistently learned a set of policies dedicated to a single circuit.

The action space corresponded to the available tasks. For the experiments validating the TAVC model we used a greedy-ϵ action selection strategy [30], where ϵ was set to 0.1. Because we wanted the system to remain adaptive to changes in the environment, we did not decrease ϵ over time, as is common.

Softmax Action Selection as a Performance Differentiator When robots keep a set of task related utilities, different functions can be used to select the next task to undertake based on these utilities [30]. With a Greedy-ϵ function, all the tasks, apart from the one with the highest utility, have equal probability, ϵ, of being explored. With a softmax function however, the probability of trying a suboptimal task is correlated with the relative estimated utility of that task. Using a Boltzmann softmax function on a task-selection level, has reliable effects on an inter-robot level, where it functions as a mechanism for allocating high-value tasks to high-performance robots. A robot that on average can service tasks in time $\overline{p}$ will have a difference in estimated task utility that correlates with the expression $\frac{w_h - w_l}{\overline{p}}$ where w_h and w_l are the values of high- and low-value tasks respectively. A fast robot with a lower average service time $\overline{p}$ will have a correspondingly higher difference between the estimated utility of high- and low-value tasks. Using a softmax action selection function, this greater difference in estimated utility theoretically translates into a greater probability of servicing high-value tasks. Such persistence will lead to the fast

robots servicing high-value tasks and may, depending on the group dynamics and the task values, lead to the slow robots servicing low-value tasks.

5 Evaluation

In order to evaluate the TAVC model and the TAVC algorithm we have conducted several experiments in the domain of prioritized transportation. First we conducted experiments to validate the TAVC model of task allocation and to verify the advantages of the TAVC algorithm. Second, we conducted an experiment which demonstrated that the TAVC algorithm is sensitive to different levels of performance among the robots and hence could be applied to heterogeneous groups of robots.

5.1 The Prioritized Transportation Problem

Cooperative transportation is a MRTA problem where group dynamics can have a critical impact on performance. It is also possible to construct spatially classifiable instances of this problem, which allows us to reduce the complexity of the group dynamics. This reduced complexity facilitates the initial study of algorithms for dealing with group dynamics in MRTA.

In the basic transportation problem, a group of robots traverse a given environment in order to transport items between the sources and the sinks. We call the time taken to traverse the environment once from a sink via a source and back to a sink the *traversal time*. To perform optimally on this task the robots must maximize the number of traversals in general. The basic transportation problem [9] is one of the sub-problems of foraging [1, 15, 27]. If the locations of sources and sinks are given, the foraging problem is reduced to a problem of transportation. Here we present the *prioritized transportation* problem, which generalizes the basic transportation problem to problems where the sources and sinks are divided into sets of different priority. Cooperative transportation is a general MRTA problem; the work on prioritized transportation presented here thus provides insights into the behavior of practical multi-robot systems in general.

When there is a source close to each sink and sinks are far apart, the optimal allocation is to have robots distributed over the local source/sink pairs or *circuits*, so as to avoid the increased traversal time implied by crossing between circuits. To optimize its performance on the prioritized transportation problem, a group of robots must strike the correct balance between different target values and different traversal times, as defined by the amount of interference on each circuit. We consider fetching and delivering one puck to be one *task*. In terms of scheduling, each transportation task corresponds to a job and the traversal times correspond to job processing times. The value of a task/job corresponds to its weight. The robots correspond to machines and, since their allocation has a significant effect on the performance of the system, the robots/machines are interaction-dependent. When the sources and sinks are distributed so as to

make the tasks/jobs spatially classifiable, the problem of optimizing throughput in prioritized transportation is an instance of either $R_{ID} \mid sc \mid \sum w_i C_i$ or $P2_{ID} \mid sc \mid \sum w_i C_i$, according to whether the robots are heterogeneous or homogeneous. In Section 2, we showed both of these problems to be $\mathcal{NP}$-hard.

5.2 The Simulated Environment

The experiments were done in simulation using the *Player* robot device server and *Stage* simulator [13] software platform. Controllers written for the *Stage* simulator have been repeatedly shown to work with little or no modification on Pioneer mobile robots. The robots in the experiments were simulated Pioneer 2DX robots with SICK laser range-finders and PTZ color cameras. Each robot wore colored markings that could be recognized using ActivMedia's Color-Tracking Software (ACTS). The prototype markings as they appear on a real Pioneer 2DX are shown in Figure 2.

Figure 2: Prototype Pioneer 2DX with color markings

The experiments took place in a simulated 12 by 8 meter environment with six robots and two sets of sources and sinks. Figure 3 shows a graphical rendering of the simulated environment in which the experiments took place, with the circuits indicated by dashed arrows and the sources and sinks labeled.

The sources and sinks were simulated laser bar-codes made from highly reflective material and recognizable by the laser range finder. We did not require actual objects to be carried. A proximity of less than 1 meter to a source or sink was interpreted as a delivery or a pick-up. The relatively high number of robots and a relatively low number of sources and sinks, or task classes, emphasizes the fact that the dominant factor in the complexity of multi-robot task allocation of spatially classifiable tasks is the number of robots, not the number of tasks.

5.3 Validating the TAVC Model

The goal of our validation experiments was two-fold: 1) to show both that the allocations produced by the TAVC algorithm satisfied the predictions of the

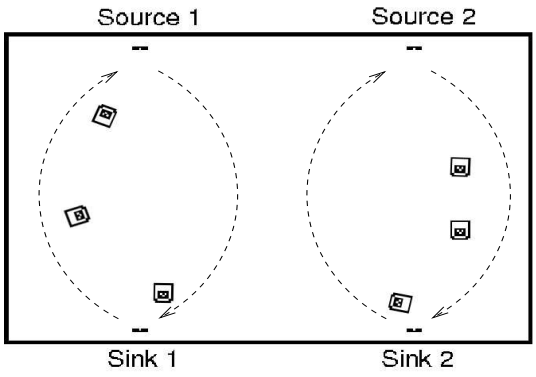

Figure 3: The simulation environment, with circuits Indicated

TAVC model and 2) that the group's performance improved on the performance of hand-coded solutions. Toward that end, we designed three experiments:

1. **Base Distribution:** The goal of this experiment was to demonstrate that the TAVC algorithm distributed six robots over two tasks of different value in an optimal way according to the TAVC model.

2. **Filling a Vacancy:** This experiment was designed to demonstrate that the TAVC algorithm could recover from a change in the optimal allocation by filling a vacancy. We created a vacancy by removing one of the robots occupying a high-value service slot. According to the TAVC model, this vacancy should be filled by one of the robots servicing a low-value service-slot in order to retain an optimal allocation. By retaining optimality, the TAVC algorithm would demonstrate that it can improve on the performance of hand-coded solutions. The TAVC algorithm is a general algorithm for retaining optimality, while hand-coded solutions must be specialized to each problem based on a model of the group dynamics. We argued in Section 3.2 that high-quality models of group dynamics are difficult to construct.

3. **Breakdown Without Vacancy:** This experiment was a control experiment showing that, in accordance with the TAVC model, the removal of a robot servicing a low-value service slot did not lead to a change in the allocation of the remaining robots.

Together, these three experiments demonstrate that the TAVC algorithm establishes and maintains optimal allocations as defined by the TAVC model. To show that the performance improvement in our experiments are due to the TAVC algorithm, we also ran the three control experiments using robots that randomly chose between the available tasks. To show how the vacancy chain algorithm outperforms hand-coded MRTA algorithms, we ran an experiment where the assumed optimal allocations for the initial setup was hand-coded in

the robot controllers. We emphasize that in general it is not possible to reliably identify this allocation and that this is the initial motivation for this work.

Reward Function In order to demonstrate the emergence of a vacancy chain structure, we designed a set of rewards that would promote the allocations predicted by the TAVC model. We call the circuit with the highest related reward the *high-value* circuit and, correspondingly, the circuit with the lowest related reward the *low-value* circuit. Specifically, in order to produce an initial allocation where three robots serviced one circuit and three robots serviced the other, it was necessary that it was less attractive to be one of four robots servicing the high-value circuit than to be one of three servicing the low-value circuit. This constraint on the reward function is presented formally in Equation 10.

$$\forall(x, y).\overline{r}_{x,4} < \overline{r}_{y,3} \tag{10}$$

In order for a vacancy in the high-value circuit to be filled, it must be more attractive to be the third robot in that circuit than to be the third robot in the low-value circuit. This is expressed formally in Equation 11, where p denotes the preferred circuit.

$$\forall(x \neq p).\overline{r}_{x,3} < \overline{r}_{p,3} \tag{11}$$

We empirically estimated the relevant average traversal times. To satisfy the constraints given these times, we chose the circuit values in Equation 8 to be $w_1 = 2200$ and $w_2 = 2000$. We emphasize that the robots do not keep any information about the task values and that the optimal allocation emerges as each robot's utility estimates converge as a result of individual experience only. The fact that the robots are oblivious to task values also allows new allocations to emerge when external factors such as the task values or the group size change.

5.4 Handling Heterogeneity

Having validated the TAVC model and task allocation algorithm, we extended the algorithm to cover heterogeneous groups of robots. This was done by introducing a softmax action selection function rather than the ϵ-greedy function used for the validation experiments.

We had two aims for the heterogeneous robot experiment. First, to test whether the modified TAVC algorithm would produce the allocation predicted to be optimal by the TAVC model, where the three fast robots serviced the high-value circuit and the three slow robots serviced the low-value circuit. Second, to demonstrate that this allocation improved the performance of the system to a level significantly above the performance level of a group where tasks were allocated randomly.

The robots were divided into two groups in order to make them heterogeneous. The first group was made to operate at a default speed of 300 mm/sec. The second group had a default speed of 200 mm/sec. The speeds were chosen to be equidistant from the speed used in the validation experiments in order to

preserve the distribution where three robots served the high-value circuit and three robots serving the low-value circuit.

Reward Functions The temperature parameter τ was set empirically to 0.005. With these values the experimentation rate was significant without being overwhelming. Because we wanted the system to remain adaptive to changes in the environment we did not decrease ϵ or τ over time, as is common.

6 Results

6.1 Validating the TAVC Model

We performed the three main experiments defined in Section 5.3, as well as a number of supporting experiments. The resulting data are presented and analyzed below. Our student-t tests for statistical significance are all done with respect to a 90% confidence level. For each experiment we defined a convergence period and a stable period according to the stability of the system performance.

Base Distribution This experiment used six robots. It consisted of 20 10-hour trials, each averaging 3000 traversals in total or 500 traversals per robot. The convergence period was 2.5 hours.

To evaluate performance, we consider the last target visited by each robot. This gives seven possible system states. We refer to each state using the notation $h : l$, where h is the number of robots whose last target was on the high-value circuit. Correspondingly, l is the number of robots whose last target was on the low-value circuit. The rows labeled $TAVC$ in Table 1 show the mean, $\hat{\mu}$, and standard deviation, s, of the time the system spent in each of the states. The values are percentages of the total stable period. The rows labeled R describe the same values for a set of 20 trials using a group of robots that randomly chose between tasks.

State		0:6	1:5	2:4	3:3	4:2	5:1	6:1
TAVC	$\hat{\mu}$	0.1	2.7	19.0	41.8	29.3	6.6	0.5
	s	0.3	1.9	7.7	6.6	9.0	3.2	0.5
R	$\hat{\mu}$	2.0	7.2	22.6	34.5	24.7	8.2	0.7
	s	0.6	2.9	3.4	3.1	3.3	3.1	0.4
T		1.6	9.4	23.4	31.2	23.4	9.4	1.6

Table 1: State-time distributions for six robots

The row labeled T lists the number of different ways to choose a sample of size n from a population of m, as percentage of all possible samples, according to Equation 12. It is worth noticing that the time distribution produced by random allocation is closely aligned with this theoretical estimate, though the differences are statistically significant.

$$T = \frac{m!}{n!(m-n!)2^m} \tag{12}$$

The two time distributions given in Table 1 are presented as histograms in Figure 4 with the standard deviation indicated by the error bars for each state.

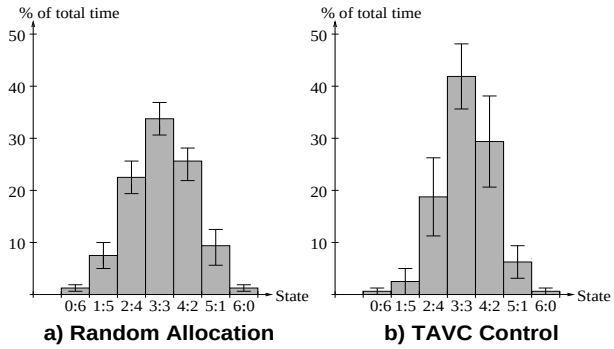

Figure 4: The state-time distributions of six randomly allocated robots and six TAVC-controlled robots

Over 20 experiments, the difference in time spent in state 3 : 3 is statistically significant. The time the TAVC-controlled group spent in state 3 : 3 is also statistically higher than the time spent in any of the other states. This confirms that the group's action-selection policies have converged to promote the state defined as optimal by the TAVC model, given the estimated task-processing times.

Figure 5 presents the average performance of a group of robots controlled by the vacancy chain algorithm over both the convergence period and the stable period. This group's performance is indicated by the thick, solid line. The average performance of random allocation is indicated by the dashed line. The performance is calculated as the sum of the delivery frequencies for each circuit weighted by the value of the task.

The performance data show that a group of robots controlled by the TAVC algorithm performs significantly higher than six randomly allocated robots. Together, the time distribution data and the performance data show that the adaptive controllers improve the group's performance by adopting a dedicated service structure that conforms to the predictions of the TAVC model.

Filling a Vacancy This experiment used five robots. We removed randomly one of the three robots that were servicing the high-value circuit at the end of the previous experiment, thus creating a vacancy on that circuit according to the TAVC model. The experiment consisted of 20 10-hour trials. The convergence period was 2.5 hours.

The converged TAVC algorithm kept the system in state 3 : 2 for a significantly larger amount of time than a group of five randomly allocated robots. The

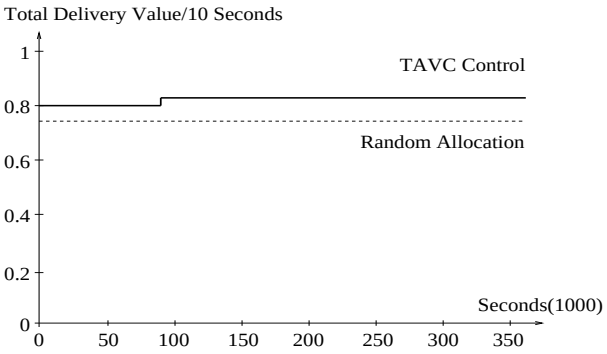

Figure 5: The performance of TAVC and random allocation for six robots

State		0:5	1:4	2:3	3:2	4:1	5:0
TAVC	$\hat{\mu}$	0.8	7.8	31.1	40.7	17.7	1.8
	s	1.1	5.1	6.3	5.3	5.9	1.6
R	$\hat{\mu}$	2.6	12.6	32.4	34.8	15.5	2.1
	s	0.9	2.2	4.0	2.3	3.7	0.4
C		3.1	15.6	31.3	31.3	15.6	3.1

Table 2: State-time distributions of five robots after a breakdown created a vacancy on the high-value circuit

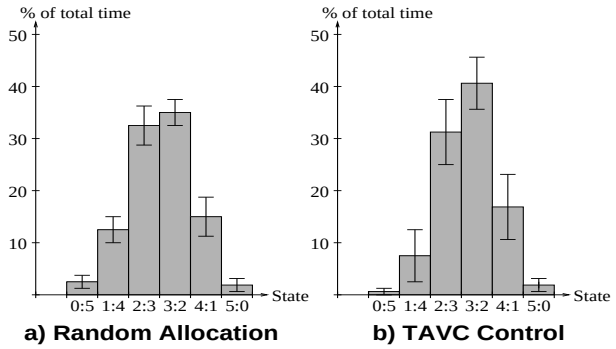

Figure 6: The state-time distributions of five randomly allocated and five TAVC-controlled robots after a breakdown created a vacancy on the high-value circuit

time distributions are given in Table 2 and a graphical presentation if provided inn Figure 6. This showed that the group had adapted its structure from one that promoted the $3:3$ state to one that promoted the $3:2$ state. The change demonstrated that a robot from the low-value circuit had filled the vacancy we had created in the high-value circuit.

The performance data presented in Figure 7 show that the removal of a robot from the high-value circuit caused the performance to drop sharply. After the re-convergence period, the performance rose again to a level that was significantly higher than the performance of random allocation and also significantly higher than the mean performance, over 20 trials, of a group of robots controlled by a hand-coded allocation algorithm optimized for six robots. The average performance of that group is indicated by the thin solid line.

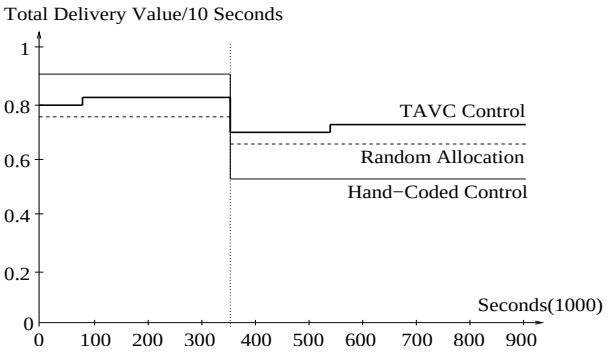

Figure 7: The performance of TAVC, random allocation, and hand-coded control before and after a breakdown creating a vacancy on the high-value circuit

Breakdown Without Vacancy This experiments used five robots from the end of the first experiment. We removed randomly one robot from the low-value circuit. According to the TAVC model, this did not create a vacancy and, hence, the system was expected to remain in the $3:2$ state. The experiment consisted of 20 10-hour trials, and the convergence time was 2.5 hours.

The state-time distribution during the stable period of this experiment, presented in Table 3, was not significantly different from the distribution produced during the second experiment presented in Section 6.1.

State	0:5	1:4	2:3	3:2	4:1	5:0
$\hat{\mu}$	0.3	6.7	34.6	47.1	10.5	0.7
s	0.3	3.7	9.2	9.4	3.8	0.4

Table 3: State-time distributions after a breakdown that did not create a vacancy

As shown in Figure 8, performance fell significantly when the robot was re-

moved, but remained significantly higher that the performance of five random controllers. There was no significant difference in the performance during the stable period of this experiment and the stable period of the second experiment where a vacancy was created. Also, there was no significant difference in performance between the convergence and stable periods of this, third, experiment. This consistency in the performance reflects the fact that the group structure remained unchanged.

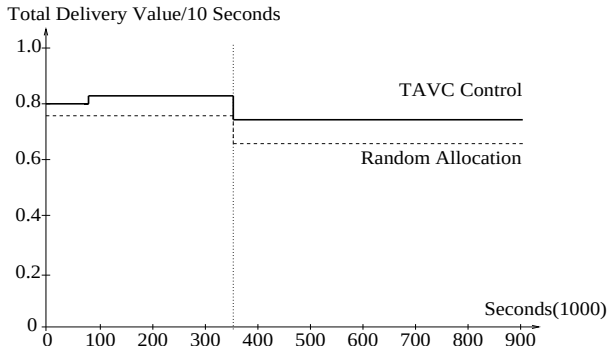

Figure 8: The performance of TAVC and random allocation before and after a breakdown that did not create a vacancy

The result demonstrates that the TAVC algorithm produces the allocations defined as optimal by the TAVC model, independently of which robot it is that breaks down.

6.2 Handling Heterogeneity

We defined a convergence period of 15 hours based on the stability of the system performance. The current allocations was identified by looking at which of the robots visited the high-value circuit last. We used three fast and three slow robots, yielding fifteen possible system states. We refer to each state using the notation $f : s$, where f is the number of fast robots whose last target was on the high-value circuit. Correspondingly, s is the number of slow robots whose last target was on the high-value circuit. The columns labeled $\hat{\mu}_a$ and s_a in Table 4 show the mean and standard deviation of the time the system spent in each of the states while running the modified TAVC algorithm. The values are percentages of the total stable period. The columns labeled $\hat{\mu}_r$ and s_r give the mean and standard deviation of the time the system spent in each of the states for a set of 15 trials using a group of robots that randomly chose between tasks.

The column labeled T lists the combinatorial probability of choosing a sample of size f from a population of $g = 3$ fast robots as well as choosing a sample of size s from a population of $h = 3$ slow robots. This probability is given in Equation 13. It is worth noticing that the time distribution produced by random allocation is closely aligned with the theoretical estimate, though the

$f:s$	$\hat{\mu}_a$	s_a	$\hat{\mu}_r$	s_r	T	$\hat{\mu}_a - \hat{\mu}_r$	$\frac{\hat{\mu}_a - \hat{\mu}_r}{\hat{\mu}_r}$
0:0	0.2	0.2	1.4	0.5	1.5	-1.3	-0.88
0:1	1.5	1.2	3.4	1.7	4.7	-3.2	-0.67
0:2	2.0	2.6	5.0	1.4	4.7	-2.7	-0.58
0:3	0.5	0.7	2.0	1.1	1.5	-1.1	-0.70
1:0	2.9	1.6	3.6	1.4	4.7	-1.8	-0.38
1:1	12.1	4.9	13.5	3.3	14.1	-2.0	-0.14
1:2	14.4	6.1	14.4	2.4	14.1	0.3	0.02
1:3	4.1	2.7	4.0	1.4	4.7	-0.6	-0.13
2:0	7.0	5.0	5.1	1.2	4.7	2.3	0.48
2:1	19.4	7.4	15.7	2.3	14.1	5.30	0.37
2:2	18.5	5.0	14.7	3.0	14.1	4.4	0.3
2:3	5.7	3.6	4.4	2.4	4.7	1.0	0.20
3:0	2.7	4.1	1.7	0.6	1.5	1.2	0.78
3:1	5.2	4.6	5.2	1.6	4.7	0.5	0.10
3:2	3.3	2.6	4.2	1.2	4.7	-1.4	-0.29
3:3	0.7	0.7	1.5	0.8	1.5	-0.9	-0.56

Table 4: State-Time Dist. for Heterogeneous Robots

differences are statistically significant.

$$T = \frac{100g!h!}{f!(g-f)!s!(h-s)!2^g 2^h} \tag{13}$$

The difference between the state-time distribution produced by the modified TAVC algorithm and the distribution produced by random allocation is presented in the column labeled $\hat{\mu}_a - \hat{\mu}_r$. This difference is presented as a percentage of the mean times from the random distribution, $\hat{\mu}_r$, for each state in the last column, labeled $\frac{\hat{\mu}_a - \hat{\mu}_r}{\hat{\mu}_r}$. The difference between the distributions produced by the adaptive controllers and the random controllers, i.e., the last two columns, are also presented graphically by the histograms in Figure 9.

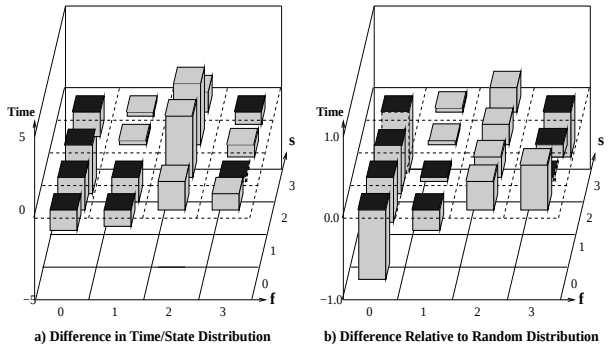

Figure 9: Difference in Distributions

Over these 15 experiments, the increase in time spent in state $0:3$ is statis-

tically significant. In Figure 9 b) the optimal state, 0 : 3, stands out as the state with the highest relative increase in time. This confirms that the group's set of Q-tables have converged to promote the state defined as optimal according to the TAVC model. The performance data also show that the performance of a group of robots controlled by the TAVC algorithm, 0.081 of target value per 10 seconds, is significantly higher than the performance of random allocation, 0.078 of target value per 10 seconds. Together, the state-time distribution data and the performance data show that the modified TAVC algorithm improve the group's performance by adopting a dedicated service structure that conforms to the predictions of the TAVC model. The learned Q-tables show that, as predicted, the fast robots, on average, have higher estimated utilities for high-value tasks and higher average differences between the estimated utilities of high- and low-value tasks.

7 Conclusions

Our experiments showed that the TAVC algorithm is an efficient and robust task allocation algorithm that is sensitive to differences in individual agent performance. The fact that the TAVC algorithm needs only a minimal amount of information about any particular problem domain makes it potentially applicable to a large class of problems. We demonstrated that in spite of the difficulties related to modeling group dynamics and the general complexity of scheduling, it is possible, under certain restrictions, to use the interference sensitive TAVC algorithm to improve on random allocation and on hand-coded solutions. It would have been informative to compare the performance of the TAVC algorithm to other known ML-based algorithms such as the progress estimation-based algorithm used by the L-ALLIANCE system [28] or a more general algorithm based on, e.g., Markov games [21]. Such algorithms may or may not outperform the TAVC algorithm. However, the fundamental difference between such solution and the TAVC algorithm is that the TAVC algorithm works without any explicit communication, while other algorithms imply communication overheads. For example, in L-ALLIANCE, each robot explicitly estimates the progress of other robots, and for Markov games, the global state has to be communicated in order to satisfy the Markov property.

Our work on the TAVC algorithm has brought up several issues that do not relate directly to the experimental results. Below we discuss the most important of those in some detail.

7.1 Applicability

The inspiration for the vacancy chain algorithm and the experimental results presented here are all taken from the multi-robot domain. As a general scheduling algorithm it is interesting to evaluate the applicability of the vacancy chain algorithm outside that domain. All domains where cooperation and/or competition provide significant effects from group dynamics are possible candidate

problems for the vacancy chain algorithm. One example is network load minimization for distributed cooperating processes. In this domain, different distributions, or allocations, imply different communication loads. A group of migrating processes, using the TAVC algorithm, could, in this domain, converge on distributions that would minimize the network load.

In general, in order to optimize an allocation problem using the TAVC algorithm, the jobs must be spatially classifiable, i.e., the effects of interaction must be restricted to work within classes of jobs. Also, the machines must be able to choose between the jobs, and the quality of each job solved must also be available to each machine to provide feedback to the local utility estimates.

7.2 Optimality and Stability

A group of non-communicating, adaptive, greedy robots can be seen as participants in a multi-player game and as such, the group behavior can be analyzed using game theory [26]. To converge to a stable solution, the TAVC algorithm depends on finding allocations where no single robot can benefit from unilaterally changing tasks. In game theory, such allocations are called *Nash equilibria*. Another important result from game theory is that some problems have *suboptimal* Nash equilibria, and some problems *do not have* Nash equilibria, unless we consider mixed strategies. Mixed strategies are not practical as we have placed a limiting factor on switching between classes of tasks in our definition of spatially classifiable tasks in Section 2.1. Problems with suboptimal Nash equilibria, as discussed below, would lead to suboptimal allocations. On problems without Nash equilibria, the TAVC algorithm would not converge.

Optimality Consider two robots, m_1 and m_2, and two task classes, k_1 and k_2, with identical values. Assuming identical traversal times and identical start and finishing times, the interaction function defined in Equation 14 will lead to convergence on a suboptimal allocation.

$$g_i(\mathcal{A}) = \begin{cases} 3 & \text{if } \mathcal{A} = [k_1, k_1] \\ 2 & \text{if only } m_i \text{ serves } k_1 \\ 5 & \text{if only } m_i \text{ serves } k_2 \\ 4 & \text{if } \mathcal{A} = [k_2, k_2] \end{cases} \tag{14}$$

This interaction function assumes that some allocations produce synergistic effects. One example of a synergistic effect is when the robots need to continuously repair a degrading path, e.g., by removing obstacles that appear on a regular basis. In this case, the average traversal time for two robots can be higher than for one robot.

The average processing time matrix corresponding to the interaction function presented in Equation 14 is presented in Table 5. This matrix shows the expected average processing time, $g_1(\mathcal{A})/g_2(\mathcal{A})$ for m_1 and m_2 servicing task classes k_1 and k_2.

The optimal allocation, with the minimal processing times for the interaction functions given above, is $[k_1, k_1]$, with the minimal total average processing time

		m_2	
		k_1	k_2
m_1	k_1	3/3	5/2
	k_2	2/5	4/4

Table 5: Processing time matrix leading to sub-optimal allocation

4. The individual reward, due to the lower processing time, for a robot when unilaterally changing to task k_2 or *defecting* will likely be reflected in both the robots' task utility estimates. With an exploration rate, ϵ, below 0.5, the majority of the exploratory changes will, on average, be unilateral. This is likely to lead to one of the robots eventually changing tasks. In game theory, a problem with this structure is called a *prisoner's dilemma*, and a unilateral change away from an optimal solution for individual gain is called a *defection*. The defection of one robot will increase the remaining robot's processing time on k_1 task and eventually force it to defect as well. Intuitively, k_2 tasks will look more attractive due to the lower repair load, but turns out to be worse due to high levels of interference, e.g., due to less available space.

An exploration rate of less than 0.5 will inhibit the robots from returning to the $[k_1, k_1]$ allocation, as most exploratory changes are unilaterally and will yield a reward lower than the one for the $[k_2, k_2]$ allocation.

Instability It is possible, with heterogeneous robots, to construct processing time matrices that imply an ever-changing allocation. In game theory these are games without Nash equilibria. One such processing time matrix is given in Table 6.

		m_2	
		k_1	k_2
m_1	k_1	2/2	4/1
	k_2	4/1	2/2

Table 6: Unstable processing time matrix

With the job processing time matrix presented in Table 6, there will always be an incentive for m_1 to change in order to be working alone on a task. For m_2, on the other hand, it will always be advantageous to change in order to work on the same task as m_1. This will leave the system in a constant state of change, not settling on any of the optimal allocations.

7.3 Commitment vs Opportunism

Matarić, Sukhatme and Østergaard [25] have empirically studied the role of commitment and opportunism in task allocation, defining a parameter space over which different degrees of these are preferable. The exploration rate, ϵ, and learning rate, α, together with the reward function, decide how easily our

robots switch between tasks. By adjusting these parameters, a wide spectrum of commitment levels is available.

8 Future Work

The prioritized transportation problem has a very restricted interaction function which reduces the scheduling complexity. We hope to explore problems with more complex interaction functions in order to assess the general applicability of the vacancy chain algorithm.

Vacancy chain distribution only partially describes the distribution mechanism used to distribute shells among hermit crabs. It is also common for crabs to fight over shells [8]. Inspired by such negotiated resource exchanges, it might be possible to produce algorithms that go further in allocating high resources to high quality consumers.

Acknowledgments

This work was supported in part by the Department of Energy (DOE) Robotics and Intelligent Machines (RIM) Grant DE-FG03-01ER45905 and in part by the Office of Naval Research (ONR) Defense University Research Instrumentation Program (DURIP) Grant 00014-00-1-0638.

References

[1] Tucker R. Balch. The impact of diversity on performance in multi-robot foraging. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *The proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 92–99, Seattle, Washington, May 1 - 5 1999. ACM Press.

[2] Tucker R. Balch, Zia Khan, and Manuela M. Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01)*, pages 521–528, Montreal, Canada, May 31 - June 4 2001. ACM Press.

[3] Christian Blum and Michael Sampels. Ant colony optimization for fop shop scheduling: A case study on different pheromone representations. In *Proceedings of the 2002 Congress on Evolutionary Computing (CEC'02)*, pages 1558–1563, Honolulu, Hawaii, May 12 - 17 2002. IEEE Press.

[4] Silvia Botelho and Rachid Alami. M+ : a scheme for multi-robot cooperation through negotiated task allocation and achievemen. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*

(ICRA'99), pages 1234–1239, Detroit, Michigan, May 10-15 1999. IEEE Press.

[5] Michael Bowling and Manuela M. Veloso. Rational and convergent learning in stochastic games. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1021–1026, Seattle, Washington, August 4 - 10 2001. Morgan Kaufmann.

[6] Wilfried Brauer and Gerhard Weiß. Multi-machine scheduling - a multi-agent learning approach. In *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 42–48, Paris, France, July 4 -7 1998. IEEE Press.

[7] Peter Brucker. *Scheduling Algorithms*. Springer Verlag, Berlin, Germany, second edition, 1998.

[8] Ivan D. Chase, Marc Weissburg, and Theodore H. Dewitt. The vacancy chain process: a new mechanism of resource distribution in animals with application to hermit crabs. *Animal Behavior*, 36:1265–1274, 1988.

[9] Torbjørn S. Dahl, Maja J. Matarić, and Gaurav S. Sukhatme. Adaptive spatio-temporal organization in groups of robots. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pages 1044–1049, Lausanne, Switzerland, September 30 - October 4 2002. IEEE Press.

[10] Torbjørn S. Dahl, Maja J. Matarić, and Gaurav S. Sukhatme. Multi-robot task-allocation through vacancy chains. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03)*, pages 2293–2298, Taipei, Taiwan, September 9 - 14 2003. IEEE Press.

[11] Brian P. Gerkey and Maja J Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, October 2002.

[12] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939 – 954, 2004.

[13] Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, Gaurav S. Sukhatme, and Maja J Matarić. Most valuable player: A robot device server for distributed control. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, pages 1226–1231, Wailea, Hawaii, October 29 - November 3 2001. IEEE Press.

[14] Dani Goldberg and Maja J Matarić. Learning multiple models for reward maximization. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 319–326, Stanford, California, June 29 - July 2 2000. Morgan Kaufmann.

[15] Dani Goldberg and Maja J Matarić. Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In Tucker R. Balch and Lynne E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 315–244. A K Peters Ltd, 2001.

[16] Kwun Han and Manuela M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the 9th International Symposium on Robotics Research (ISRR'99)*, pages 199–204, Snowbird, Utah, October 9 - 12 1999.

[17] Owen Holland and Chris Melhuish. Stigmergy, self-organization and sorting in collective robotics. *Artificial Life*, 5(2):173–202, 1999.

[18] Boyoon Jung and Gaurav S. Sukhatme. Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous Robots*, 13(3):191–205, 2002.

[19] Thomas H. Labella, Marco Dorigo, and Jean-Louis Deneubourg. Self-organised task allocation in a group of robots. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS'04)*, pages 371–380, Toulouse, France, June 23 - 25 2004.

[20] Kristina Lerman, Asram. Galstyan, Alcherio Martinoli, and Auke J. Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7(4):375–393, 2001.

[21] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, pages 157–193, New Brunswick, New Jersey, July 10 - 13 1994. Morgan Kaufmann.

[22] Kian H. Low, Wee K. Leow, and Marcello H. Ang. Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In Deborah L. McGuinness and George Ferguson, editors, *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 28–33, San Jose, California, July 25-29 2004. AAAI Press/MIT Press.

[23] Maja J. Matarić. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1994.

[24] Maja J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[25] Maja J. Matarić, Gaurav S. Sukhatme, and Esben Østergaard. Multi-robot task allocation in uncertain environments. *Autonomous Robots*, 14(2-3):255–263, 2003.

[26] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, Massachusetts, 1991.

[27] Esben Østergaard, Gaurav S. Sukhatme, and Maja J. Matarić. Emergent Bucket Brigading - A simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the 5th International Conference on Autonomous Agents (Agents'01)*, pages 29–30, Montreal, Canada, May 28 - June 1 2001. ACM Press.

[28] Lynne E. Parker. L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behaviour-Based Systems. *Advanced Robotics, Special Issue on Selected Papers from IROS'96*, 11(4):305–322, 1997.

[29] Anil K. Seth. Modelling group foraging: Individual suboptimality, interference, and a kind of matching. *Adaptive Behavior*, 9(2):67–91, 2001.

[30] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[31] Poj Tangamchit, John M. Dolan, and Pradeep K. Khosla. Learning-based task allocation in decentralized multirobot systems. In Lynne E. Parker, George Bekey, and Jacob Barhen, editors, *Distributed Autonomous Robotic Systems 4, Proceedings of the 5th International Symposium in Distributed Autonomous Robotic Systems (DARS'00)*, pages 381–390, Knoxville, Tennessee, October 4 - 6 2000. Springer.

[32] Barry B. Werger and Maja J Matarić. Broadcast of local eligibility for multi-target observation. In Lynne E. Parker, George Bekey, and Jacob Barhen, editors, *Distributed Autonomous Robotic Systems 4, Proceedings of the 5th International Symposium on Distributed, Autonomous Robotic Systems (DARS'00)*, pages 347–356, Knoxville, Tennessee, October 4-6 2000. Springer.

[33] Helen Yan and Maja J Matarić. General spatial features for analysis of multi-robot and human activities from raw position data. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pages 2770–2775, Lausanne, Switzerland, September 30 - October 4 2002. IEEE Press.

[34] Wei Zhang and Thomas G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1114–1120, Montréal, Canada, August 20 - 25 1995. Morgan Kaufmann.

[35] Albert Y. Zomaya, Mathew Clements, and Stephan Olariu. A framework for reinforcement-based scheduling in parallel processor systems. *IEEE Transactions On Parallel and Distributed Systems*, 9(3):249–259, March 1998.