Scheduling with Group Dynamics: a Multi-Robot Task Allocation Algorithm based on Vacancy Chains

Torbjørn S. Dahl, Maja J. Matarić, and Gaurav S. Sukhatme
tdahl|mataric|gaurav@usc.edu
Robotics Research Laboratory, Center for Robotics and Embedded Systems
Department of Computer Science, University of Southern California
Henry Salvatori Computer Science Center
941 West 37th Place, SAL 214, Los Angeles, California 90089-0781

Abstract

Existing task allocation and scheduling algorithms, including task-allocation algorithms for multi-robot systems, generally assume that tasks are independent. This assumption is often violated in groups of cooperative mobile robots, where the group dynamics can have a critical impact on performance. We present a multi-robot task allocation algorithm that is sensitive to group dynamics. Our algorithm is based on *vacancy chains*, a resource distribution process common in human and animal societies. We study the problem of cooperative transportation in simulation. We demonstrate through experiments in simulation that if robots keep local task utility estimates, and follow a greedy task selection policy, the interactions in the group cause the collection of learned policies to converge toward an optimal allocation pattern as defined by the vacancy chain framework. As the robots are continuously updating their individual utility estimates, the vacancy chain algorithm has the additional property of adapting automatically to changes in the environment, e.g., robot breakdowns or changes in task values. Our experiments show that in the case of such changes, the vacancy chain algorithm consistently outperforms random and static task allocation algorithms. Finally, the vacancy chain algorithm uses no communication or unique roles, and as a result it is more likely to scale to large groups and will degrade gracefully in response to individual breakdowns.

1 Introduction

Existing task allocation and scheduling algorithms, including task-allocation algorithms for multi-robot systems, generally assume *task independence*, i.e., they do not consider synergistic or interference-related aspects of group dynamics. Task independence is an important assumption in that it facilitates the production of optimal schedules by general scheduling algorithms. In static domains such as job-shop scheduling, task independence is a fair assumption, but it is not always valid in groups of cooperative mobile robots. In such systems group dynamics can have a critical impact on performance. This forces task allocation algorithms to be sensitive to such dynamics if they are to produce optimal schedules. However, as we will demonstrate, the problem of scheduling with group dynamics is $\mathcal{NP}$-hard, and modeling group dynamics can in itself be difficult. This implies that general algorithms for identifying optimal schedules in multi-robot systems are infeasible. It is possible to circumvent the modeling problem by estimating the task utilities under different allocation patterns without an explicit model of the underlying dynamics. However, estimating the task utilities for all possible allocation patterns can only be done when the number of robots and tasks is small, as the related number of possible allocation patterns grows exponentially. We present a task allocation algorithm that handles the complexity of task allocation by using local task utility estimates and by relying on the ability of groups of adaptive robots to produce specialized individuals. Using Q-learning with an ϵ-greedy task selection function, the interaction between the robots produces a collection of learned policies that converge toward an optimal allocation pattern as defined by the *vacancy chain* formalism. We do not initially handle the full complexity of general task utility estimation for multi-robot systems, but study at a restricted class of multi-robot task allocation problems.

In Section 2 we review the formal framework used to study traditional scheduling as well as some common results with regard to the complexity of scheduling problems. This framework is later used to formalize the problem of multi-robot task allocation and relate it to other well-studied scheduling problems in terms of relevant properties and problem complexity. In Section 3 we review previous work on modeling group dynamics. Group dynamics set multi-robot task allocation problems apart from traditional scheduling problems. Problems in modeling such dynamics limit the soundness of the predictions that can be made about group performance and job processing times. This affects the feasibility of finding optimal schedules in the multi-robot task allocation domain. Section 4 presents the problem of multi-robot task allocation in detail and formalizes the problem in the scheduling-theoretic framework reviewed in Section 2. We demonstrate that multi-robot task allocation is an $\mathcal{NP}$-hard problem by showing that a different scheduling problem known to be $\mathcal{NP}$-hard, reduces to the multi-robot task allocation problem. We conclude that optimal solutions in terms of scheduling are infeasible for multi-robot task allocation problems and suggest the use of machine learning as a way to estimate the effects of the group dynamics and to provide heuristics for a heuristics-based scheduling algorithm. Section 4 also reviews previous work on multi-robot task allocation and previous work using machine learning methods to overcome the problems of complexity in scheduling. In Section 5 we define a sub-problem of the general multi-robot task allocation problem where jobs are *spatially classifiable*. We study this sub-problem as a way of examining the issues of group dynamics in scheduling without embracing the full complexity of general multi-robot task allocation. Section 5 also reviews *vacancy chains*, a resource distribution process common in human and animal societies, and presents an algorithm for multi-robot task allocation for spatially classifiable jobs based on the vacancy chain process. Section 6 presents a concrete multi-robot task allocation problem, *prioritized transportation*, and describes how some instances of this problem belong in the restricted problem class we suggest to study. In Section 7 we present the robot controllers implementing the vacancy chain task allocation algorithm. Section 8 presents three experiments we designed to evaluate the vacancy chain task allocation algorithm and Section 9 presents the results. In Section 10 we discuss the experimental results along with a number of other issues related to our vacancy chain task allocation algorithm and multi-robot task allocation in general. Finally, Section 11 outlines promising avenues for future work.

2 Scheduling

In an idealized form, multi-robot task allocation is the problem of optimizing, over time, with respect to some given criteria, the allocation of tasks to robots. This problem can be formalized within the existing framework of scheduling. Scheduling (Brucker, 1998) has been well studied and the complexity of many scheduling problems has previously been established. These results suggest what classes of algorithms may be applicable to different scheduling problems. If the multi-robot task allocation problem is defined in terms of a scheduling formalism, those results can also indicate what classes of algorithms are suitable for that problem.

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. $p_{ij} = p_i/s_{ij}$.

2.1 Optimality and Complexity

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., $Q \mid p_i = 1 \mid \sum f_i$, 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 (Brucker, 1998).

2.2 On-Line Scheduling

Scheduling without a priori knowledge of the start-times of the tasks is called on-line scheduling. On-line scheduling algorithms are said to be optimal if they can *achieve scheduling*, i.e., if they find a schedule that does not break any deadlines, whenever any other algorithm can do so. Dertouzos and Mok (Dertouzos & Mok, 1989) showed that without a priori knowledge of task start-time, it is impossible to guarantee optimal scheduling.

3 Modeling Group Dynamics

Scheduling problems assume task independence, i.e., they do not consider synergistic or interference-related aspects of group dynamics. In multi-robot task allocation however, the effects of group dynamics can have a critical impact on system performance. When group dynamics have a major effect of performance, understanding and modeling these dynamics is necessary to predict the value of different schedules. Accurate models of the effects of group dynamics however, are hard to construct. The effects of group dynamics on a group's performance may correlate non-linearly with the number of robots working on a task. Effects such as synergy and interference may contribute concurrently to group performance. Models of group dynamics have been divided into *microscopic* or *simulation-based* and *macroscopic* (Lerman, Galstyan, Martinoli, & Ijspeert, 2001). Simulation-based models explicitly represent each agent in the model and the properties of the system can be recorded as the agents interact. Macroscopic models directly describe properties of systems in terms of abstract parameters 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.

Simulation-Based Models Goldberg and Matarić (Goldberg & Matarić, 2000) developed Augmented Markov Models, or 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.* (Balch, Khan, & Veloso, 2001) studied live insect colonies and construct 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 (Han & Veloso, 1999) in order to recognize colony behaviors.

Yan and Matarić (Yan & Matarić, 2002) 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 (Seth, 2001) 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 the distinction 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 (Matarić, 1994) 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.* (Lerman et al., 2001) have 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.

Generally macroscopic or phenomenological models are not sophisticated enough for optimizing specific task allocation problems. 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 models, but the time needed to construct these models makes them unsuitable for task allocation 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 task allocation 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 in order to be of predictive use. Our task allocation algorithm uses reinforcement learning as a way of handling the unknown group dynamics by continuously estimating the effects the group dynamics have on relevant task properties.

4 Multi-Robot Task Allocation

Multi-robot task allocation is concerned with allocating tasks to robots. This problem can also be framed in the more general scheduling framework. When this is done, the terms *job* and *task* as well as the terms *machine* and *robot* become interchangeable. We will use the terms *job* and *machine* when we relate the multi-robot task allocation problem to other scheduling problems and the terms *task* and *robot* when we discuss the specifics of multi-robot task allocation.

In groups of mobile robots, the group dynamics can have a critical impact on the job processing times, p_i. As these dynamics are difficult to model, and as a result, their effect on task processing times is difficult to predict. Within a given environment, the group dynamics may depend on which machines/robots the different jobs are allocated to, i.e., the *allocation pattern*. We formally define an allocation pattern to be a function from 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 pattern can be seen as a slice of a **schedule** as represented by a Gantt chart (Brucker, 1998). Figure 1 illustrates this for two allocation patterns $\mathcal{A}_1$ and $\mathcal{A}_2$. The patterns are indicated by the dashed lines through a job-oriented schedule presented as a Gantt chart. The corresponding allocation patterns are presented in Equations 1 and 2.

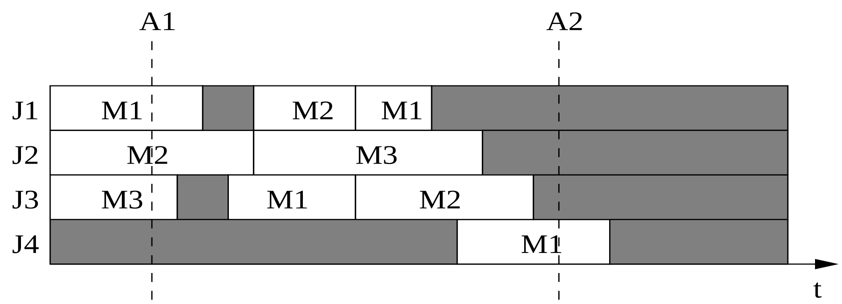

Figure 1: Job-Oriented Gantt chart

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

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

We suggest that when the effects of group dynamics on job processing times are mainly dependent on allocation patterns, these dynamics 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 allocation patterns 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 with this kind of dynamics *interaction dependent machines* and denote the class using the letter S.

4.1 Computational Complexity

The 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 subclass 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 $S \parallel \sum w_i C_i$ with an identical set of machines, S, 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 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 allocation patterns $\mathcal{A}_{t_s,t_f}$. For non-preemptable jobs, M_j will be the same for all those patterns. 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 pattern, $\mathcal{A}_{t_s}$. 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 truthfully 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 $S \parallel \sum w_i C_i$ which *ipso facto*, must also be $\mathcal{NP}$-hard.

4.2 Previous Work on Multi-Robot Task Allocation

A number of algorithms for multi-robot task allocation already exist. Below we review a selection of the most prominent of these and discuss how their different approaches to task utility effect their applicability in domains where group dynamics have significant effects on group performance.

Botelho and Alami's M+ algorithm (Botelho & Alami, 1999) 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.

Gerkey and Matarić's Murdoch system (Gerkey & Matarić, 2002) used a set of metrics to locally score the suitability of the participating robots and a publish/subscribe communication paradigm with an auction mechanism for task allocation. Murdoch also used unique task monitors or 'watchers'. Frequent re-allocations however, makes Murdoch robust to individual robot failures.

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 (Werger & Matarić, 2000), 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 model free eligibility estimates. Werger and Matarić also described a dynamic leader selection mechanism where a unique leader monitored the environment for new tasks and relinquished the leadership role for high-priority tasks. The available position as leader was then taken by a robot currently assigned to a low-priority tasks, in a mechanism that closely resembles a vacancy chain. This mechanism however would fail critically if the current leader broke down.

Both the Murdoch and the BLE algorithm implicitly ignore group dynamics by delegating the responsibility of utility estimation to the individual robots and using these estimates to allocate tasks from a global perspective. Since the interaction function depends on global allocation patterns, it is not possible for robots to independently estimate the global task utility.

In the L-ALLIANCE work by Parker (Parker, 1997), 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 multi-robot task allocation algorithms that are sensitive to the effects of group dynamics. As such, our vacancy chain algorithm represents an attractive alternative for domains where these dynamics have significant effects on the system performance.

4.3 Learning Approaches to Scheduling and Task Allocation

As we have shown, the multi-robot task allocation problem is, like many scheduling problems, $\mathcal{NP}$-hard. One way of dealing with $\mathcal{NP}$-hard problems is to use polynomial time heuristic algorithms that produce sub-optimal but satisfactory solutions. Learning such heuristics in the domain of scheduling, is a well studied problem. Below we present a selection of successful uses of machine learning as a way of finding heuristics for scheduling and discuss the applicability of these algorithms in domains with significant effects from group dynamics.

Zhang and Dietterich (Zhang & Dietterich, 1995) presented a reinforcement learning 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.* (Zomaya, Clements, & Olariu, 1998) 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 the algorithm by Zhang and Dietterich and the algorithm by Zomaya *et al.* are general algorithms applicable to all scheduling problems. However, it relies on a centralized learning mechanism. Such mechanisms can fail critically on a robot breakdown and does not scale easily to large groups.

Brauer and Weiß (Brauer & Weiß, 1998) 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 reinforcement learning, 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 is restricted to completely ordered jobs and this limits its applicability to general scheduling and task-allocation problems.

Blum and Sampels (Blum & Sampels, 2002) 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 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.* (Tangamchit, Dolan, & Khosla, 2000) used distributed reinforcement learning 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 machine learning 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 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.

5 Interference-Sensitive Task Allocation

There are two main reasons why it is difficult to find optimal solutions to multi-robot task allocation problems when the group dynamics have a significant effect on performance.

First, the problem has been shown to be $\mathcal{NP}$-hard. This indicates that for large numbers of robots and jobs, it is not feasible to find an optimal solution within a reasonable time-frame. In such problems, finding an optimal solution might also not be worth the trouble. Corkill and Lesser (Corkill & Lesser, 1983) demonstrated that in the domain of coordination of distributed problem solving networks, the computational and communication cost of finding an optimal solution could outweigh the benefits such a solution brought over solutions produced using heuristic methods.

Second, there is currently no way of accurately modeling group dynamics so as to support on-line task allocation. As long as this is the case, task allocation algorithms must rely on approximate models or estimates of the interaction function. The number of possible allocation patterns increases exponentially with the number of robots and tasks. This indicates that in general it is not feasible to even estimate accurately the complete interaction function.

5.1 Spatially Classifiable Jobs

As a way of reducing the complexity of estimating the interaction function, we focus initially on a particular class of problems where the machines are homogeneous and the jobs are *spatially classifiable*. Spatially classifiable jobs can be divided into 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, sc. The problem of optimizing the weighted total flow time in a system with interaction dependent machines and spatially classifiable jobs is denoted $S \mid sc \mid \sum w_i C_i$. One example of such problems is when classes of jobs take place in spatially separate areas. For such jobs, the speed of a machine, M_j, with respect to a given job, J_i, in class K_k, is a function of the allocation of the machines that are currently working on jobs in class, K_k, only. The allocation of robots between jobs of other classes is not relevant. Within this class we restrict ourselves to looking at problems where the interaction function is dependent only on the *number* of robots, m_k, currently working on jobs in class K_k. The job processing time function or interaction function for this class is given in Equation 4.

$$p_{ij} = g_i(m_j) \tag{4}$$

In this problem class, an estimate of the interaction function is tractable. While these restrictions limit the applicability of our algorithms, they allow us to do initial studies of group dynamics in task allocation without being swamped by the complexity of the problem in its most general form.

5.2 The Vacancy Chain Distribution Process

The inspiration for our adaptive task allocation algorithm is the vacancy chain process. The vacancy chain process is a process 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. Thus the resources that are 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 (Chase, Weissburg, & Dewitt, 1988) 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 sub-optimal ones.

The vacancy chain resource distribution mechanism is both simple and powerful. It is based on *stigmergy*, unintentional communication between the robots through their effects on the environment (Holland & Melhuish, 1999). 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 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 allocation patterns as it does not consider possible differences in machine speed. This initial study of vacancy chain distribution reflect this by only implementing it in groups of homogeneous robots.

5.3 Task Allocation through Vacancy Chains

Inspired by the vacancy chain process we developed a formal framework describing how the allocation of tasks to machines influences system performance for spatially classifiable tasks. Our model also has the important property of breaking the group performance down to individual machine contributions. This property allows us to use distributed control algorithms.

According to our vacancy chain formalism, 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.

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 getting in each other's way. The difference in task processing frequency together with the class task value, v_i, define the contribution of the last robot added or the last service-slot filled. We call this contribution, which can be negative, the *slot-value*, $s_{i,j}$. The formal definition is given in Equation 5.

$$s_{i,j} = (c_{i,j} - c_{i,j-1})v_i \tag{5}$$

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.

In this framework, 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, the *throughput value*, by allocating robots to tasks or service-slots in order of decreasing slot-value. In this case, the throughput value, T, for n robots is simply the sum of the individual contributions as stated in Equation 6.

$$T_n = \sum_n s_n \tag{6}$$

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. Distributed algorithms are generally more robust than centralized algorithms. They also scale more easily to large numbers of robots.

In a scenario where the service-slots are allocated optimally, a failure in a robot servicing a high-value task will result in an empty high-value service-slot that must be re-allocated for optimal system performance. If the value of the vacant slot is greater than the value of one or more of the other occupied service-slots, the vacant slot will have to be filled in order to restore optimal performance. Expressed in the vacancy chain framework, a vacant, high-value service-slot is a resource to be distributed between the robots.

5.4 The Vacancy Chain Task Allocation Algorithm

We have previously studied the use of distributed reinforcement learning as a way of optimizing the performance of multi-robot systems in an interaction sensitive manner (Dahl, Matarić, & Sukhatme, 2002). In our vacancy chain task allocation algorithm, a task corresponds to an action, and each robot keeps a local estimate of task utilities and choses its next task using an ϵ-greedy selection function.

Q-learning is not sensitive to the frequency of rewards. Hence, the estimated values of actions do not necessarily correspond to the action's 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 7.

$$r = w_i/t \tag{7}$$

This reward function promotes the tasks with the highest value because these will on average provide a higher reward. However, if a robot consistently occupies a service-slot that is sub-optimal 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. For j robots servicing a task, i, the formal relationship between slot-value, average reward, $\overline{r}_{i,j}$, and average task processing time $\overline{t}_{i,j}$ is given in Equation 8.

$$S_{i,j} = v_i \sum_j (c_j - c_{j-1}) = \sum_j \overline{r}_{i,j} = v_i \sum_j \frac{1}{\overline{t}_{i,j}} \tag{8}$$

6 The Prioritized Transportation Problem

Cooperative transportation is a multi-robot task allocation problem where group dynamics can have a critical impact on performance. It is also possible to construct spatially classifiable instances of this problem which lets us reduce the complexity of the group dynamics found in the general transportation problem. This reduced complexity again facilitates the initial study of algorithms for dealing with group dynamics in multi-robot task allocation.

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 the this task the robots must maximize the number of traversals in general. The basic transportation problem (Dahl et al., 2002) is one of the sub-problem of foraging (Balch, 1999; Goldberg & Matarić, 2001; Østergaard, Sukhatme, & Matarić, 2001). If the locations of sources and sinks are given, the foraging problem is reduced to a problem of transportation. The prioritized transportation problem extends the basic transportation problem by dividing the sinks into sets of different priority.

When there is a source close to each of the sinks and the sinks are far apart, the optimal solution is to have the 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 different interference rates 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 $S \mid sc \mid \sum w_i C_i$. According to our vacancy chain formalism, this problem can be solved optimally by distributing the robots over the optimal service-slots according to the vacancy chain formalism. We studied one concrete instance of the prioritized transportation problem and demonstrated through a series of experiments, that our algorithm consistently produced dedicated optimal allocation patterns in accordance with the definition of optimality provided by the vacancy chain framework.

7 Controller Architecture

All the robots in our demonstration used the same adaptive, behavior-based controller (Matarić, 1997). However, our vacancy chain task allocation algorithm is independent of the underlying architecture, being defined purely in terms of distributed reinforcement learning of task utilities.

Based on individual experience our robots learned to specialize. However, they always retained a certain level of exploration that allowed them to find and fill new vacancies.

7.1 The State/Action Space

Each controller in our experiments had a set of pre-programmed high-level *approach* behaviors. Each behavior corresponded to servicing one of the available tasks. Each of the high-level behaviors consisted of multiple shared lower level behaviors:

- **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.

7.2 Adaptation

The individual robots were homogeneous with respect to hardware configuration and control algorithms. Each robot used reinforcement learning with a randomly initialized Q-table. Over time, the reinforcement learning differentiated the group by letting the individual robots specialize on certain tasks. The robots used temporal difference Q-learning (Sutton & Barto, 1998) to associate the different input states with one of the high level approach 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 tasks in order to construct optimal task sequences. In spite of this, the robots consistently learned a set of policies dedicated to single tasks.

The action space corresponded to the available tasks. For action selection we used a greedy-ϵ strategy (Sutton & Barto, 1998), 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.

8 Experimental Design

The goal of our experiments was to show that a group of robots could optimize its performance by distributing tasks through a vacancy chain structure. First we aimed to show that the group structure and individual robot actions satisfied the requirements for resource distribution through vacancy chains. Secondly we wanted to showed that the group's performance was significantly better than the performance of groups of robots using random or static task allocation algorithms.

The experiments were done in simulation on the *Player/Stage*(Gerkey, Vaughan, Støy, Howard, Sukhatme, & Matarić, 2001) software platform[1]. From experience, controllers written for the *Stage* simulator work with little or no modification on real Pioneers. 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 ActiveMedia's Color-Tracking Software (ACTS). The prototype markings as they appear on a real Pioneer 2DX are shown in Figure 2.

The experiments took place in a simulated twelve by eight meter environment with two sets of sources and sinks. Figure 3 shows a graphical rendering of the simulated environment in which the experiments took place, with the sources and sinks labeled.

The sources and sinks were simulated laser-beacons, effectively bar-codes made from high-reflection material and recognizable by the laser range finder. We did not require actual objects to be carried. A minimum proximity to a source or sink was interpreted as a delivery or a pick-up.

[1] *Player* is a server and protocol that connects robots, sensors and control programs across a network. *Stage* simulates a population of Player devices, allowing off-line development of control algorithms. *Player* and *Stage* were developed jointly at the USC Robotics Research Labs and HRL Labs and are freely available under the GNU Public License from *http://playerstage.sourceforge.net*

Figure 2: Prototype Pioneer 2DX with Color Markings

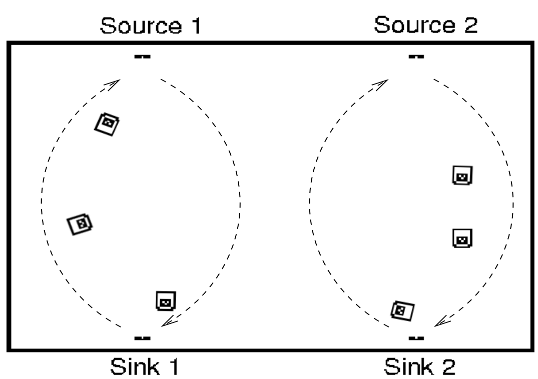

Figure 3: The Simulated Environment with Circuits Indicated

To demonstrate that the structure of the group conformed to the definition of resource distribution through vacancy chains we designed three experiments as follows:

1. **Base Distribution:** The goal of this experiment was to show that the vacancy chain framework could distribute six robots over two tasks of different value in the pattern it describes as optimal.

2. **Filling a Vacancy:** This experiment was designed to demonstrate the filling of a vacancy. We created a vacancy by removing one of the robots occupying a high-value service slot. According to the vacancy chain framework, this vacancy should be filled by one of the robots servicing a low-value service-slot.

3. **Breakdown without Vacancy:** This experiment was a control experiment showing that, in accordance with the vacancy chain framework, the removal of a robot servicing a low-value service slot did not lead to robot migration.

Together these three experiments demonstrate that the vacancy chain algorithm can establish and maintain optimal allocation as defined by the vacancy chain framework.

8.1 Reward Function

In order to demonstrate that resource distribution through vacancy chains could emerge from a collection of adaptive policies of highly abstracted behaviors we designed a set of rewards that would promote this behavior in the robots. We call the circuit with the highest related reward the *high-value* circuit and correspondingly, the the circuit with the lowest related reward is called the *low-value* circuit.

Specifically, in order to produce an initial task allocation where three robots serviced circuit one and three robots serviced circuit two, it was necessary that it was less attractive to the robots 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 Equations 9.

$$\forall(x, y).\overline{r}_{x,4} < \overline{r}_{y,3} \tag{9}$$

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 10, where p denotes the preferred circuit.

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

We empirically estimated the relevant average traversal times. To satisfy the given constraints we chose the circuit values, as defined in Equation 7, to be $w_1 = 2200$ and $w_2 = 2000$.

9 Results

We performed the three main experiments defined in Section 8, for different configurations of robots and tasks as well as a number of supporting empirical experiments. The resulting data are presented and analyzed below.

9.1 Base Distribution

For each experiment we defined a convergence period and a stable period according to the stability of the system performance. This experiment started with six robots that had randomly initialized Q-tables. We performed 20 10-hour long individual experiments, each averaging 3000 traversals in total or 500 traversals per robot. The convergence period was 2.5 hours.

To examine the 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 A in Table 1 shows the mean, $\overline{s}$ 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
A	μ	0.1	2.7	19.0	41.8	29.3	6.6	0.5
	σ	0.3	1.9	7.7	6.6	9.0	3.2	0.5
R	μ	2.0	7.2	22.6	34.5	24.7	8.2	0.7
	σ	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: Empirical end Theoretical 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 11. It is worth noticing that the time distribution produced by the six random controllers is closely aligned with the theoretical estimate, though the differences are statistically significant.

$$T = \frac{m!}{n!(m-n!)2^m} \tag{11}$$

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.

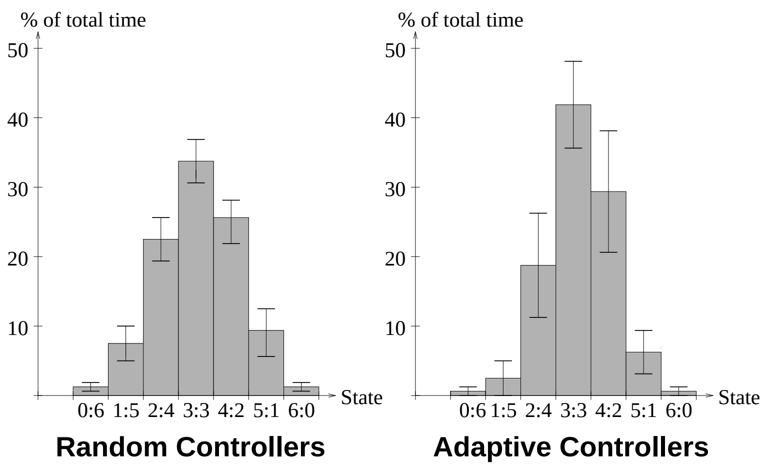

Figure 4: Time-State Distribution for Six Random and Six Adaptive Robots

Over 20 experiments, the difference in time spent in state 3 : 3 is statistically significant. The time the adaptive group spends in state 3 : 3 is also statistically higher than the time spent in any of the other states. This confirms that the group's set of task-selection policies have converged to promote the state defined as optimal according to the vacancy chain framework, the rewards and 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 a group of six robots controlled by an algorithm that choses randomly between the high-level approach behaviors 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.

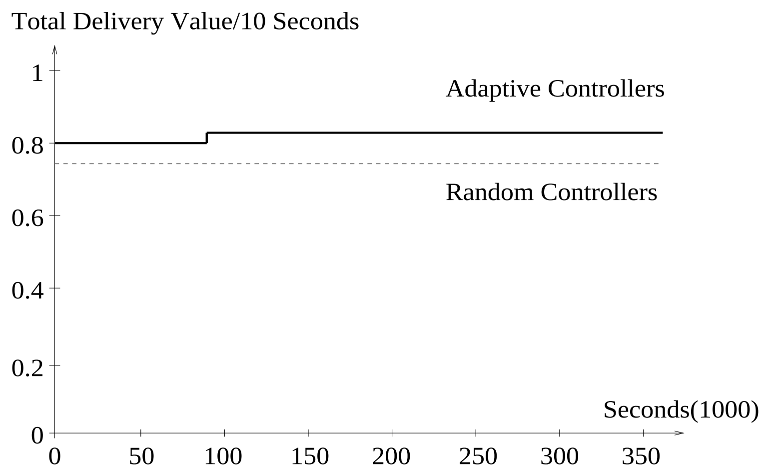

Figure 5: The Performance of Six Adaptive Robot Controllers

The performance data show that the performance of a group of robots controlled by the vacancy chain algorithm is significantly higher than the performance of a group of robots controlled by a random choice algorithm. 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 rules of the vacancy chain framework.

9.2 Filling a Vacancy

In order to demonstrate filling of a vacancy, we performed a second experiment, with five robots. The robots used Q-tables from the end of the stable period of the initial convergence experiment. We removed randomly one of the three robots that were servicing the high-value circuit, thus creating a vacancy on that task. We performed 20, 10 hour experiments. The convergence period again was 2.5 hours. The data presented here are taken from the stable period.

The converged controllers kept the system in state $3:2$ for a significantly larger amount of time than a group of five random controllers. The values of the time distributions are given in Table 2 and a graphical presentation if provided in 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. This change showed that a robot from the low-value circuit had filled the vacancy we

State		0:5	1:4	2:3	3:2	4:1	5:0
A	μ	0.8	7.8	31.1	40.7	17.7	1.8
	σ	1.1	5.1	6.3	5.3	5.9	1.6
R	μ	2.6	12.6	32.4	34.8	15.5	2.1
	σ	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: Imperial and Theoretical Distributions after a Breakdown creating a Vacancy

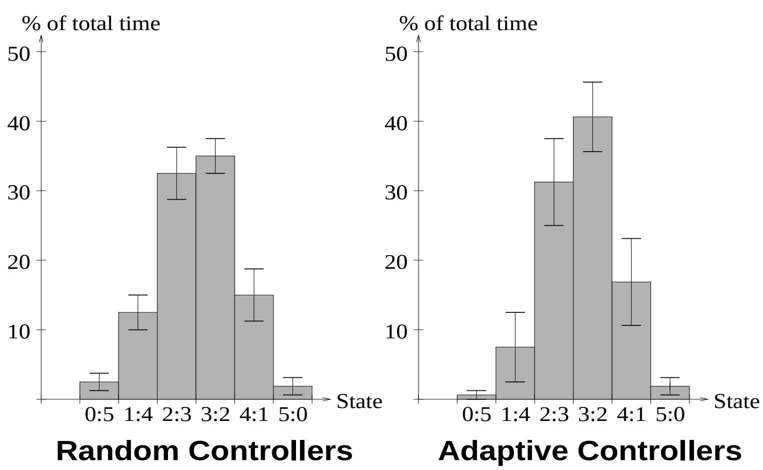

Figure 6: State-Time Distributions after a Breakdown creating a Vacancy

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 causes 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 five random controllers and also significantly higher than the mean performance, over 20 trials, of a group of robots controlled by a static task allocation algorithm. The average performance of the static group is indicated by the thin solid line.

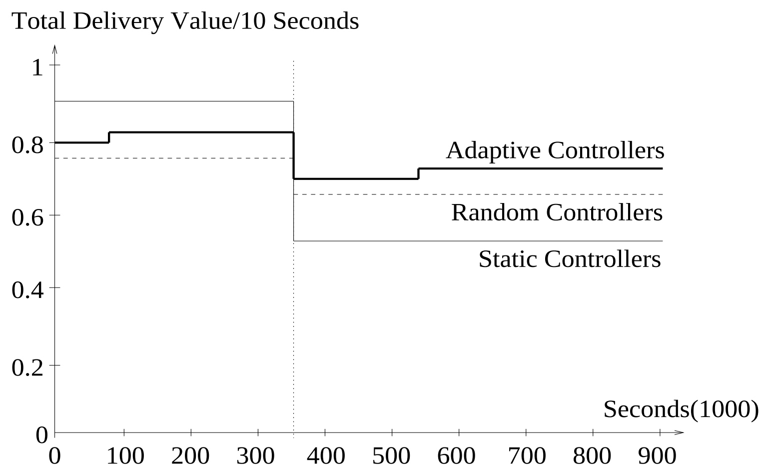

Figure 7: Group Performance on a Breakdown creating a Vacancy

9.3 Breakdown without Vacancy

To show that our algorithm is stable with respect to failures in robots assigned to low-value service-slots, we performed an experiment similar to the one presented in Section 9.2, but this time we removed a robot from the low-value circuit.

According to the vacancy chain formalism, this did not create a vacancy and hence, the system was expected to remain in the 3 : 2 state. We ran 20 individual 10 hour trials for this experiment, and the convergence time was 2.5 hours.

The time distribution during the stable period of this experiment, presented in Table 3, was not significantly different from the distribution produced during the experiment presented in Section 9.2.

As shown in Figure 8, performance fell significantly when the robot was removed, but remained significantly higher that the performance of five ran-

State	0:5	1:4	2:3	3:2	4:1	5:0
μ	0.3	6.7	34.6	47.1	10.5	0.7
σ	0.3	3.7	9.2	9.4	3.8	0.4

Table 3: State-Time Distribution after a Breakdown not creating a Vacancy

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

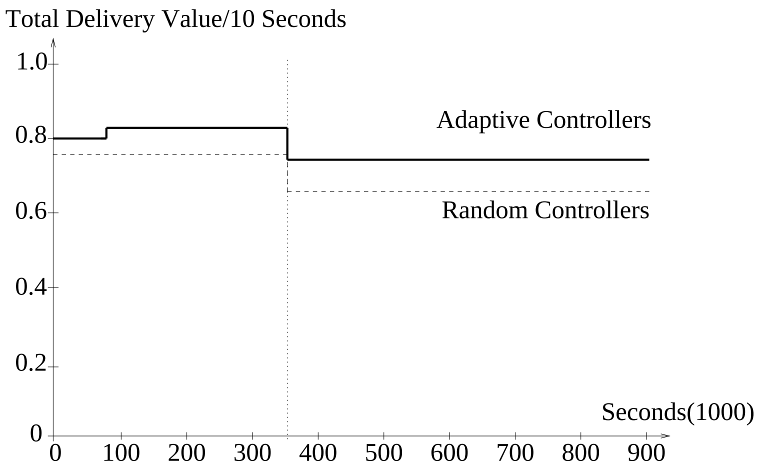

Figure 8: Group Performance on a Breakdown not creating a Vacancy

This result demonstrates that our algorithm produces the group structure required for vacancy chain distribution, independently of which particular robot may fail.

10 Discussion

Our experiments showed that the vacancy chain mechanism is efficient and robust as a task allocation algorithm. The fact that our 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 vacancy chain algorithm to improve on existing task allocation algorithms. In particular, we studied a prioritized transportation problem where the robots/machines were interaction dependent and where the tasks/jobs were spatially classifiable. Our experiments showed that using distributed reinforcement learning and rewards based on the vacancy chain framework, the allocation patterns that emerged were consistent with the definition of optimality defined by that framework.

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

10.1 Stability and Optimality

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. To converge to a stable solution, our algorithm depends on finding allocation patterns where no single robot can benefit from unilaterally changing tasks. In game theory, such allocation patterns are called *Nash equilibria* (Ritzberger, 2002). Anther important result from game theory is that some problems have sub-optimal Nash equilibria, and some problems do not have Nash equilibria. Problems with sub-optimal Nash equilibria, such as prisoner's dilemmas, would lead to sub-optimal allocation patterns. On problems without Nash equilibria, our algorithm would not converge.

Prisoner's Dilemmas Consider two robots m_1 and m_2 and two tasks classes k_1 and k_2 with identical value. Assuming identical traversal times, start and finishing times, and interaction functions, the interaction function defined in Equation 12 will lead to convergence on a sub-optimal allocation pattern.

$$g_i(\mathcal{A}) = \begin{cases} 2 & \text{if } \mathcal{A} = [k_1, k_1] \\ 4 & \text{if other robot changes class} \\ 1 & \text{if this robot changes class} \\ 3 & \text{if } \mathcal{A} = [k_2, k_2] \end{cases} \tag{12}$$

This interaction function assumes that some allocation patterns produce synergistic effects. One example of a synergistic effect is when the robots need to continuously repair a degrading path, e.g., by removing items 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 12 is presented in Table 4. This matrix shows the expected average processing time for m_1 and m_2 respectively based on the task the robots are allocated to.

		m_2	
		k_2	k_2
m_2	k_1	1/1	2/3
	k_2	3/2	4/4

Table 4: Processing Time Matrix Leading to Sub-Optimal Allocation

The optimal allocation pattern, with the minimal processing times, for the interaction functions given above, is $[k_1, k_1]$, with the minimal total average processing time, 4. The individual reward, due to the lower processing time, for a robot when unilaterally changing to task k_2 or *deflecting*, 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. The deflection of one robot will increase the remaining robot's processing time on k_1 task and eventually force it to deflect 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 rate will inhibit the robots from returning to the $[k_1, k_1]$ pattern as most exploratory changes are unilaterally and will yield a reward lower than the one for the $[k_2, k_2]$ allocation pattern.

Instability It is possible, with heterogeneous robots, to construct processing time matrices that imply ever changing allocation patterns. In game theory these are games without Nash equilibria. One such processing time matrix is given in Table 5.

With the job processing time matrix presented in Table 5, 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 allocation patterns.

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

Table 5: Unstable Processing Time Matrix

10.2 Satisfying the Vacancy Chain Formalism

The results presented above showed that our implementation of task allocation satisfied the restrictions on vacancy chain mechanisms reviewed in Section 5.

1. The reusability of a resource was demonstrated by the filling of a vacancy produced by a robot breakdown. The robot that filled the vacancy was using the same resource that was used by the robot that broke down.

2. The requirement of a vacancy before an individual took a new resource was demonstrated by the stability of the time distributions presented in Figures 4 and 6. These distributions did not change significantly until a vacancy was introduced. The ϵ-greedy action selection algorithm ensured that the robots constantly looked for vacancies and that the robots would choose to occupy new units when these had higher estimated utility.

3. In both time distributions presented in Figures 4 and 6, multiple robots consistently serviced the low-value circuit. The service-slots on the low-value circuit were suboptimal compared to the first three service-slots on the high-value circuit. This shows that multiple individuals occupied suboptimal resource units.

10.3 Allocation Strategies

The goal of task allocation is not always to maximize throughput. Keeping general geographical patterns can be another possible goal. Theoretically the vacancy chain algorithm can be manipulated through the reward function to produce specific allocation patterns. In dynamic environments and when robots are subject to individual failures, the adaptive properties can then be used to establish optimal patterns for the current number of active robots. Below we discuss briefly reward design for throughput- and pattern-based allocation.

Value-Based Allocation Patterns The reward functions we used in our experiments were based on task values and processing times. This resulted in task allocation patterns that optimized the throughput value of the system. Optimizing the throughput value implies finding a balance between task value and robot interference. Equation 6 formalizes this implication for applications that satisfy the restrictions in the vacancy chain framework. Equation 5 defines the individual slot values for a task i being serviced by j robots in terms of task value, v_i, and processing frequencies, $c_{i,j}$.

Slot-Based Allocation Patterns Reward functions can also, in theory, be used to produce pre-specified allocation patterns. This type of task allocation can be desirable in problem domains where there is no clear relationship between the values of the tasks. However, this reward function is based on an estimate of what service-slot the robot is currently filling. It can be difficult to estimate this accurately in complex or unfamiliar problem domains.

To specify the desired allocation pattern for any number of working robots we can use slot-values. By making the value of each service-slot explicit, we can implement any allocation pattern. The vacancy chain algorithm will allocate tasks to robots according to the specified patterns by ensuring that the high-value service-slots are always filled.

Imagine an application where it is desirable to have at least two robots dedicated to a task A. However, if there are also two robots dedicated to task B, then it is preferable to have the fifth robot helping on task A rather than B. One imaginable scenario demanding this kind of configuration is the exploration and analysis of unknown environments such as other planets. In this scenario it can be important to always have two robots dedicated to exploring the environment and localization of places of interest. If a place of interest is found and more robots are available, then two other robots should analyze the interesting site. If no further robots are necessary for the analysis, any other available robots should aid in the exploration.

A ranking, $\mathcal{R}$, of the service-slots in such a scenario is a vector of service-slots, as presented in Equation 13.

$$\mathcal{R} = [(1,1),(1,2),(2,1),(2,2),(1,3),(2,3),(2,4)] \tag{13}$$

We assume that unlisted service-slots all have the same ranking, one higher than the number of listed service-slots. This allocation pattern is presented graphically in Figure 9.

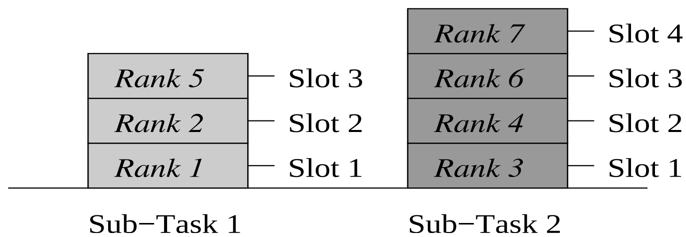

Figure 9: Service Slot Ranking for Slot-Based Allocation

A ranking defines a set of constraints on the average rewards, $r_{i,j}$, related to the individual service-slots.

The purpose of the reward function is to reflect the given allocation pattern and hence the values it produces must satisfy the constraints the pattern imposes. To do this, the reward function must differentiate between the different service slots on the same task. We have demonstrated how this can be done by using an estimate of the average processing time, $\overline{p}_{ij}$, produced by filling service-slot, j on task, i. For the experiments presented in this paper, this estimate showed itself to be a reliable indicator of the number of robots currently servicing the task.

In the prioritized transportation problem, for a given task, i, the processing time, p_{ij}, is a function of the number of robots, j, currently servicing that task.

If the interaction function, g_i, is *injective* or *one-to-one*, a reward function based on service-slots can trivially be made to mirror any desired allocation pattern. If the interaction function is not injective, but *one-to-many*, different or additional indicators must be used to allow the state estimator to differentiate between states with identical task processing times.

10.4 Commitment vs Opportunism

Østergaard, Matarić, and Sukhatme (Østergaard et al., 2001) 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.

10.5 Re-convergence

Mustapha and Lachiver (Mustapha & Lachiver, 2000) have previously presented work on re-convergence of trained RL systems. Their conclusion is that re-convergence can be either faster or slower than initial convergence depending on the difficulty of the learning task. Our experiments show that re-convergence can be slower of faster than initial convergence dependent on how closely related the new environment is to the old. In the case where one robot has to switch the task completely, effectively inverting the Q-table, the re-convergence takes roughly twice as long as the initial convergence, because the learned Q-values first have to be un-learned.

11 Future Work

Most other multi-robot task allocation algorithms, like Murdoch (Gerkey & Matarić, 2002), L-ALLIANCE (Parker, 1997), BLE (Werger & Matarić, 2000), and M+ (Botelho & Alami, 1999), can allocate tasks efficiently also for heterogeneous robots with different individual performance levels. We intend to conduct experiments with a modified vacancy chain task allocation algorithm using a static, non-Boltzmann softmax action selection function rather than the ϵ-greedy function we used in the experiments presented here. A softmax action selection function might allow the vacancy chain algorithm to solve the problem of allocating tasks among heterogeneous robots. A softmax function relates difference in utility to probability of selection. This indicates that high performing robots should have a higher probability of establishing themselves in the high-value service-slots. Since the vacancy chain algorithm is completely distributed, needs only a minimal amount of information about the problem at hand, and does not use any communication, it will be an attractive alternative to existing task allocation algorithms for domains with complex or unfamiliar dynamics.

The prioritized transportation problem has a very restricted interaction function which reduces the scheduling complexity. In the future we would also like to explore problems with more complex interaction functions in order to see how generally applicable the vacancy chain algorithm is.

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 (Chase et al., 1988). Inspired by such negotiated resource exchanges it might be possible to produce algorithms that go further in allocating high resources to high quality consumers. We intend to study local interactions in an attempt to generalize the vacancy chain task allocation algorithm to groups of heterogeneous robots.

Acknowledgements

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

References

Balch, T. R. (1999). The impact of diversity on performance in multi-robot foraging. In Etzioni, O., Müller, J. P., & Bradshaw, J. M. (Eds.), *The proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pp. 92–99, Seattle, WA. ACM Press.

Balch, T. R., Khan, Z., & Veloso, M. M. (2001). Automatically tracking and analyzing the behavior of live insect colonies. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01)*, pp. 521–528, Montreal, Canada.

Blum, C., & Sampels, M. (2002). 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)*, pp. 1558–1563, Honolulu, Hawaii. IEEE Press.

Botelho, S., & Alami, R. (1999). 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)*, pp. 1234–1239, Detroit, Michigan.

Brauer, W., & Weiß, G. (1998). Multi-machine scheduling - a multi-agent learning approach. In *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pp. 42–48, Paris, France.

Brucker, P. (1998). *Scheduling Algorithms* (Second edition). Springer.

Chase, I. D., Weissburg, M., & Dewitt, T. H. (1988). The vacancy chain process: a new mechanism of resource distribution in animals with application to hermit crabs. *Animal Behavior*, *36*, 1265–1274.

Corkill, D. D., & Lesser, V. R. (1983). The use of meta-level control for coordination in a distributed problem solving network. In *Proceedings* *of the Eight International Joint Conference on Artificial Intelligence (IJCAI'83)*, Vol. 2, pp. 748–756, Karlsruhe, West Germany.

Dahl, T. S., Matarić, M. J., & Sukhatme, G. S. (2002). Adaptive spatio-temporal organization in groups of robots. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pp. 1044–1049, Lausanne, Switzerland.

Dertouzos, M. L., & Mok, A. K.-L. (1989). Multiprocessor on-line scheduling of hard-real-time tasks. *IEEE Transactions on Software Engineering*, *15*(12), 1497–1506.

Gerkey, B. P., & Matarić, M. J. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, *18*(5), 758–768.

Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. S., & Matarić, M. J. (2001). 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)*, pp. 1226–1231, Wailea, Hawaii.

Goldberg, D., & Matarić, M. J. (2000). Learning multiple models for reward maximization. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, pp. 319–326, Stanford, California.

Goldberg, D., & Matarić, M. J. (2001). Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In Balch, T., & Parker, L. E. (Eds.), *Robot Teams: From Diversity to Polymorphism*, pp. 315–244. A K Peters Ltd.

Han, K., & Veloso, M. (1999). Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the Ninth International Symposium on Robotics Research (ISRR'99)*, pp. 199–204, Snowbird, Utah.

Holland, O., & Melhuish, C. (1999). Stigmergy, self-organization and sorting in collective robotics. *Artificial Life*, *5*(2), 173–202.

Lerman, K., Galstyan, A., Martinoli, A., & Ijspeert, A. J. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, *7*(4), 375–393.

Matarić, M. J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents*, *9*(2–3), 323–336.

Matarić, M. J. (1994). *Interaction and Intelligent Behavior*. Ph.D. thesis, Massachusets Institute of Technology.

Mustapha, S. M., & Lachiver, G. (2000). RL-Cyclist: A Self-Teaching Agent Driving a Bicycle. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, pp. 222–226. Banff, Canada.

Østergaard, E., Sukhatme, G. S., & Matarić, M. J. (2001). 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)*, pp. 29–30, Montreal, Canada. ACM Press.

Parker, L. E. (1997). 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.

Ritzberger, K. (2002). *Foundations of Non-Cooperative Game Theory*. Oxford University Press.

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

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

Tangamchit, P., Dolan, J. M., & Khosla, P. K. (2000). Learning-based task allocation in decentralized multirobot systems. In Parker, L. E., Bekey, G., & Barhen, J. (Eds.), *Distributed Autonomous Robotic Systems 4, Proceedings of the 5th International Symposium in Distributed Autonomous Robotic Systems (DARS'2000)*, pp. 381–390, Knoxville, Tennessee.

Werger, B. B., & Matarić, M. J. (2000). Broadcast of local eligibility for multi-target observation. In *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS'00)*, pp. 347–356, Knoxville, TN. Springer.

Yan, H., & Matarić, M. J. (2002). 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)*, pp. 2770–2775, Lausanne, Switzerland.

Zhang, W., & Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 14th International*

Joint Conference on Artificial Intelligence (IJCAI'95), pp. 1114–1120, Montréal, Canada. Morgan Kaufmann.

Zomaya, A. Y., Clements, M., & Olariu, S. (1998). A framework for reinforcement-based scheduling in parallel processor systems. *IEEE Transactions On Parallel and Distributed Systems*, *9*(3), 249–259.