On Benefits of Adaptive Bidding with the Contract Net Protocol in Multi-Robot Systems

By

Jonathan A. Kensler

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Dr. Arvin Agah
(Committee Chair)

Dr. Nancy Kinnersley
(Committee Member)

Dr. Jerzy Grzymala-Busse
(Committee Member)

Date of Acceptance

The Thesis Committee for Jonathan Kensler certifies
That this is the approved Version of the following thesis:

ON BENEFITS OF ADAPTIVE BIDDING WITH THE CONTRACT NET PROTOCOL IN MULTI-ROBOT SYSTEMS

____
Chairperson*

____

____

Date Approved: ______________________________

Abstract

This thesis investigates the effectiveness of using the Contract Net Protocol, an auction type system, for controlling task allocation among a group of robots, and presents and evaluates a strategy of using Artificial Neural Networks to formulate adaptive bids within the framework of the Contract Net Protocol. The robots were used in a foraging environment and showed that excellent communication among robots leads to a need for a social control mechanism for managing the robots, such as the Contract Net Protocol. The experiments also confirmed that a moderate benefit can be gained by using adaptive bidding within the framework of the Contract Net Protocol.

Acknowledgements

I would like to thank my parents, Robert and Terry Kensler for having the wisdom and foresight to provide me with my own computer far before it became so common for individuals, much less small children, to have their own computer.

I must also give thanks to my elder brother, Andrew Kensler, who has been my lifelong programming mentor.

My wife Jennifer is to be thanked for putting up with me and for helping with the statistics in this project.

Dr. Nancy Kinnersley also has my gratitude for providing me with the opportunity to be a Graduate Teaching Assistant, which has been a learning experience in and of itself, and has also provided me with this opportunity to pursue my studies.

Finally, I must give much thanks to Dr. Arvin Agah for putting up with one fiercely and perhaps foolishly independent graduate student.

Table of Contents

List of Figures

1. Introduction

A. Motivation

Imagine she is new to the US and is going to do some traveling by car. She is planning on traveling first from Kansas City, Kansas to Chicago, Illinois. Later, she is to travel from Omaha, Nebraska to Cleveland, Ohio. Then from Minneapolis, Minnesota to Kansas City, Kansas. Finally, she is to travel from Kansas City to Chicago again.

At first, one does not know anything about US roads, so she finds a map and determines the most direct roads to get from Kansas City to Chicago and travel this path. She finds that the roads involved are small roads with low speed limits.

Next, she travels from Omaha, Nebraska to Cleveland, Ohio via a major highway, and finds that the roads have a number of lanes and also have a higher speed limit than the roads from Kansas City to Chicago. This includes a section of highway between Chicago and Des Moines, Iowa. On the next trip, she again takes a major highway between Minneapolis and Kansas City, including a section between Des Moines, Iowa and Kansas City.

The next trip involves once again going from Kansas City to Chicago. The question now arises as to what path she is going to take. Is one going to take the most direct path, which because of the nature of the roads will take 20 hours to get from Kansas City to Chicago, or take a less direct path, via major highways and get to Chicago in 12 hours?

For truck drivers who base their livelihood on driving, such decisions can be extremely important. While getting to Chicago quicker would be nice, it requires driving a longer distance and at much higher speeds, resulting in increased gas consumption. Thus a truck driver must make a decision based upon her own learned experience as to whether increased speed at the cost of increased gas consumption, or lower gas consumption at the cost of speed, is ultimately going to be more profitable.

This evaluation based upon learning is then going to form the basis of how much the truck driver is going to charge for her work. Additionally, even if two trucks take the same path, there may be other considerations, such as how much can each truck carry at a time? One truck may be able to only carry two thirds of what needs to be moved, thus requiring two trips, while the other truck may be able to carry it all in one trip.

Given this, it is not expected that every truck should charge the same amount for accomplishing a task (moving something), because the trucks have different capabilities. The variation on pricing then essentially acts as a function of the efficiency of using a particular truck. The system as a whole then, benefits; and an individual with a job for a trucker chooses the trucker who offers to do it for the least amount of money, as that trucker is likely going to be the most efficient.

Getting quotes from multiple truckers for the cost of a job then becomes an important part of keeping the system working efficiently. This also results in a de facto auction.

In this thesis we examine whether the concept of informed bidding improving overall system efficiency can be carried over to a multi-agent robotic environment.

Although the cost of robotics is currently quite high, it is likely that the cost will decline and give rise to more widespread use of multi-robot systems. These systems will need to be coordinated. Task distribution through an auction type system with adaptive bidding is a promising form of coordination as it mirrors human situations such as truck drivers moving items across the country.

B. Approach

In this research, artificial neural networks are used for path planning to create informed bids for use in a contract net (auction) system for the foraging problem.

Path planning in this case involves deciding on a path to and from foraging locations and a location at which foraged items may be gathered. This involves deciding whether or not to go through areas of harsh terrain that can slow down a robot and possibly damage it.

A foraging problem was used for the experiments because it is a problem concerned with how individual behaviors affect overall system efficiency with sources of inefficiencies being the result of environmental factors, and not hostile agents actively trying to cause inefficiencies.

Within the foraging problem, there are multiple tasks to be accomplished, in the form of multiple food sources that may be gathered from. In this experiment, we examine the use of the Contract Net Protocol [1], which is a task distribution system based on auction, with adaptive bidding.

The Contract Net Protocol has previously been successfully used in a robotic multi-agent system [11] and for path planning [12], but the importance of the formation and adaptation of the bids to the overall system efficiency has not been examined, which is the focus of this thesis. For this reason, an environment devoid of hostile agents was desired, so as to simply show the importance of adaptive bidding, although examination of an adaptive bidding system in an environment with hostile agents would likely prove interesting future work.

An artificial neural network was used to adaptively create bids. Artificial neural networks have previously been used for path finding purposes [13], but in this thesis, numerous such networks are used not only for path finding, but are also used as the basis of a bidding process.

C. Thesis structure

This thesis is organized into five sections. The first section details the motivation, with a real life analogy and introduces the problem, followed by a brief description of the approach taken for the experiment in this thesis. The second section describes background and related work in depth that forms the foundation of the experiment conducted in this thesis. The third section describes the methodology of the experiment. The results of the experiments are presented and discussed in the fourth section; the conclusion and final remarks are in the fifth section.

2. Background and Related Work

A. Artificial Intelligence

Artificial Intelligence is a well known name for a rather nebulous field of study, that is in fact a collection of fields. There are four different main views of what Artificial Intelligence is (as defined by Russell and Norvig [14]): "systems that think like humans", "systems that think rationally", "systems that act like humans", and "systems that act rationally". Part of the nebulosity of the definition of the field arises from it having borrowed elements from a number of fields such as philosophy, mathematics, economics, neuroscience, psychology, linguistics, and anthropology.

There are numerous subfields in artificial intelligence such as knowledge-based systems, neural networks, multi-agent systems, natural language processing, automated reasoning, machine learning, robotics, planning, game playing, etc.

At its most general definition artificial intelligence is about creating "intelligent entities" [14] or taking entities and making them more "intelligent".

B. Distributed Artificial Intelligence and Multi-Agent Systems

Within the field of artificial intelligence, one may be concerned with making a single entity intelligent, or may be concerned with making multiple entities intelligent. The study of how to make multiple entities intelligent comprises a major sub-field in artificial intelligence: distributed artificial intelligence. Distributed artificial intelligence can be defined rather generally as "the subfield of artificial intelligence (AI) concerned with concurrency in AI computations, at many levels"[16] or more specifically in relation to multi-agent systems as "the study, construction, and application of multi-agent systems, that is systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks" [15]. These agents may be heterogeneous or homogenous, they may also be cooperative or competitive [17].

In the experiment investigated in this thesis, multiple types of robots are used, but used separately in a homogenous fashion. The robots were both cooperative and competitive, in that they were individually competitive, but collectively this competitive nature resulted in cooperation to increase the efficiency of the system as a whole.

C. Contract Net Protocol

The idea of using an auction system for task distribution in multi-agent systems (MAS) has been around for a number of years. One prominent auction system proposed for task distribution in MAS is the Contract Net Protocol proposed by RG Smith in 1980 [1].

In this protocol, an agent finds a task to be completed (how this is done is not described). This agent then acts as a manager for the task if it cannot or will not complete the task by itself. The manager may also decompose the task if possible, feasible, and required. This manager sends out a call for bids to other agents, and these agents bid on the task, or the decomposed sub-tasks. The manager selects the best agent for the job based on some criteria. This basic concept has been used in a variety of approaches, including in systems with multiple unmanned aerial vehicles (UAVs) [2].

D. Selection of Auction House Type

There are a number of different auction systems to consider. Four main types [3] are:

1.The English auction system, which is a first-price, open-cry system where each bidder may raise their bid at any point during the auction so long as the new bid is higher than all other bids already placed by competing bidders. All bids are public.

2. The First-Price, Sealed Bid auction system, in which a bidder submits a sealed bid to an auctioneer without knowing how other bidders have bid.

3. The Dutch auction, in which an auctioneer starts a bid at a high price and gradually lowers the price until someone is willing to pay for it.

4. The Vickrey auction, in which each bidder submits a sealed bid to the auctioneer. The highest bidder wins, but pays the amount bid by the second highest bid (although a variation of this method as used for selling multiple items is also sometimes called a Dutch auction [4]).

The traditional Contract Net Protocol uses a version of First-Price, Sealed Bid auction, in which only the winning bidders are notified of the auction's termination. The reason cited for this is that informing losing bidders that they have lost the auction will drastically increase the amount of communications [1], which will have a detrimental effect on the system as a whole. If another system, such as the English auction system was instead used, the amount of message traffic would significantly increase.

In the area of robotics, the volume of inter-agent communications has generally been a limiting factor, especially in ad-hoc systems that 'broadcast' messages because broadcasts are actually emulated, not built directly in [5]. In this thesis, we are examining a system that may not be fully feasible at the present time. One such potential assumption made is that the volume of communications that are permissible will increase in the future to such a point that using any auction system, regardless of the volume of communication required, will be feasible. Given this assumption, we must examine different types of auctions and see whether one or more types of auctions would work better in a system of competitive contractors.

In the English auction system, bids are made public, which reveals the bidding strategies of competitors. In the case of contractors, using an English auction system would allow each bidding contractor to know for how much competing contractors believe they can do a job.

This can potentially be useful for giving a contractor who does not have a good idea of how much a job will cost some initial idea of the cost of the job (assuming other contractors have a better idea of how much the job will likely cost). While there is certainly some potential for increasing the efficiency of the system as a whole by allowing this kind of inferred knowledge transfer through the bidding process, this is not within the focus of this thesis. In this work we want to use an agent's learning from *its own* experiences as the basis of the formulation of its bid.

The only remaining potential benefit of the English auction system is that a bidder may win an auction for less than what they were willing to spend. In the case of this thesis however, the bids are only important for establishing which contractor believes it can do the specified job with the least energy, the exact quantification of the bid is irrelevant.

Thus using the English auction system provides no special benefit for this work. However, a possible extension of this project would be to use the English auction system. and have a contractor take into consideration how much other contractors have bid. Although there is an assumption that the robots in our system will be able to handle the volume of communications needed for any type of auction, we are still left with the English auction system providing no benefit, but requiring additional amounts of communication compared to other types of auctions.

The next type of auction to consider is the First Price, Sealed Bid auction. In this type of auction, the bids are not made public. As it has already been described, in the current context of this project, bids do not need to be made public as bidders are learning from their own experiences in completing work, rather than learning from the bidding process. In this system, each bidder makes an individual estimate of how much energy will be required to complete a task. Each bidder makes only one bid, thus the volume of communications required for this type of auction is held to a minimum. The robot that believes that it can do the task using the least amount of energy will be given the task. As with the English system, this type of auction provides no special benefit in the context of this project, but it requires less communication than the English auction.

The Dutch auction involves an auctioneer gradually lowering a price (or in the case of contractors, increasing the energy believed to be needed) until the price is low enough that a bidder bids and wins the auction. This type of auction is semi-public in that the winning bid is known, but other non-winning bidders' estimation of the cost of the item is not known. This will not provide as much inferred knowledge as the English auction; although, it will likely need less communication than the English system. Therefore there are no special benefits from using this type of auction in the context of this project. It requires less communication than the English system, but will require more communication than the First Price, Sealed Bid Auction.

The last type of auction is the Vickrey auction, which is a Dutch auction where the winning bidder wins the auction, but for the amount that the second highest bidder bid. As was mentioned earlier, we are not concerned with the exact bidding amount, but are simply concerned with establishing which bidder believes that they can do a task using the least amount of energy. Thus the Vickrey auction is not different in the context of this thesis from the Dutch auction. As it performs nearly identically to the Dutch auction, it will provide no special benefit, and will require less communication that the English auction, but more than the First-Price, Sealed Bid auction.

Thus in the context of this project, no auction type provides any significant benefits over any other type of auction, but using the First-Price Sealed Auction will require less communication and also requires less effort to implement.

E. Use of Contract Net Protocol in Robotics

The use of Contract Nets for controlling multi-robot systems was really established in [19] in response to a criticism that "[Negotiation] solutions have not been adequately demonstrated in situated agent (i.e., robotic) teams, which have to live in and react to dynamic and uncertain environments amidst noisy sensors and effectors, frequent agents failures and a limited bandwidth, noisy communication mechanism" [20]. The problems described in this criticism are indicative of an apprehensiveness to use negotiation solutions, such as the Contract Net Protocol. This has limited the use of the Contract Net Protocol in multi-robot systems.

More generally, the benefit of using a market economy for controlling the exploration of a robot system was studied in [18], in which is was found that robot systems that did not communicate had an exploration coverage efficiency that was only 29% of one which used a market economy for controlling exploration. However it was not specifically using a Contract Net System, and did not investigate using an adaptive system for negotiations or bidding.

F. Neural Networks

The general problem of task distribution is exponential in complexity (NP-Complete) [21]. Artificial Neural Networks have been shown to provide feasible solutions to some NP-Complete problems [22].

Neural Networks originated in the branch of artificial intelligence that tries to make computers think like humans. The basic unit of a neural network is the neuron. A neuron has several inputs, which are individually weighted. An input function sums the weights into a single value, and this value acts as the input to an activation function (typically non-linear), which determines what values should be provided as output values. Each of these output values in turn will typically go to other separate neurons and act as inputs for other neurons.

There are two main types of neural networks, the feed-forward (acyclic) neural network and the recurrent (cyclic) neural network [23]. In recurrent neural networks, there is no end point to the network, but after data goes through the "last" neurons, the outputs for these neurons become the input values for what had been the first neurons, thus producing a cyclic pattern. This type of network is extremely complex, so feed-forward neural networks are more commonly utilized, which is the case for this thesis.

A feed-forward neural network when given a set of input values, will calculate a set of output values. As the data passes through each neuron, a weight is applied to it, and it is combined with other weighted pieces of data, and a function is applied to this combined value. Eventually some neurons will be reached whose outputs do not serve as inputs to other neurons. The outputs from these neurons are the outputs of the neural network. These values may be incorrect though. One main reason for this is because the weights applied to each input may be completely wrong, and almost always are initially as they are customarily random values to start with. They are random values to start with because initially it is impossible to know what the correct weights should be set to and yet at the same time is has been found that the initial weights must not be equal [22]. This then implies the need for a way to adjust the weights, and such a method for adjusting weights exists and is called back-propagation.

There are two main stages of establishing a feed-forward neural network: training and testing. In the training stage, inputs are given to the neural network, and the neural network produces output values. The correct output values for each piece of training data are known. The network is given the correct output and it is able to calculate the error between the values that it produced as output versus the actual known output values. Back propagation involves using this error in conjunction with formulas derived from energy and momentum in physics to adjust the weights so as to try to produce more accurate outputs.

The network goes through the cycle multiple types of using the training data as inputs, finding an error value when compared to the actual known outputs, and adjusting the weights. This cycle is usually done several hundred times in order to get the weights to stabilize, so that additional cycles will produce very little change in the values of the weights. In some cases, the weights never stabilize sufficiently.

After this, the neural network is given inputs from the test data, and it produces output values. The actual known outputs for the test data are not given to the neural network, but are instead compared with the values that are outputted, so as to assess the efficacy of the network in determining the output values.

Even after all of the adjustments to the weights are done during the training process, there is no guarantee that the neural network will produce the correct outputs during testing. If the weights have stabilized, there is the possibility of the network having memorized the training data instead of having generalized it (finding the descriptive patterns in the data).

If the weights have not stabilized, the network may in fact work well with the test data, but it may not. If not, then the architecture of the neural network may be at fault. The ability of a feed-forward neural network to learn patterns is held in the weights involved in the neural network. These weights are on each connection between each neuron in the neural network. If there are too few neurons, there are too few weights with little capacity for learning a pattern [27]. Too many neurons may be prone to memorization.

As the feed-forward neural network is not cyclical, there are layers of neurons, with one layer's outputs being the inputs of another layer. There are at least two layers, an input layer, that accepts the input values to the neural network and an output layer that outputs values from the neural network. Using only two layers limits the neural network to only linear models [25]. More often, one or more hidden layers are used which allows for non-linear models [26] by providing greater computational ability, as well as providing more weights in which patterns may be learned. There may be multiple hidden layers, with each having different number of neurons. The number of hidden layers combined with the number of neurons in each hidden layer affects how complex a pattern the neural network can learn.

The extremely complex and random nature of neural nets means that there is no way to know what the optimal network architecture (number of hidden layers and number of neurons in each) should be [24]. The determination of what network architecture to use involves a great deal of trial and error. While the unpredictability involved in determining the architecture of a Neural Network may seem like it would make Neural Networks useless, it is at the same time this very nature that makes them very useful, in that they can sometimes provide excellent quick approximations of non-linear systems that would otherwise take exponential time to compute. It is for this reason that neural networks were used in this thesis, with inputs being a path and outputs being expected energy and damage.

3. Methodology

A. Tools

There were two main steps for evaluating the tools to be used in this project. The first step was the selection of a robot simulation package, followed by the selection of a neural network software package.

I. Robot Simulation Platforms

A premise of the experiments was that the robots share a level of communication between robots that is not yet readily available. Given this, it was decided to implement the experiment on a robot simulation software platform. Several different platforms were considered in depth for use.

The criteria used for considering a platform were (1) the difficulty involved in implementing a foraging scenario (2) the availability of quality documentation and (3) the availability and ability to use existing neural network software in conjunction with the robot simulation platform.

A number of platforms were considered, but ultimately three platforms emerged as the main candidates for use: TeamBots [7], Gun-Tactyx [8], and Robocode [9].

a. TeamBots

TeamBots is a well known robot simulation platform, allowing simulated robots to be written in Java (with some C as well). TeamBots offers significant flexibility in simulating a large number of environments. Significant amount of work is required to set up an environment in TeamBots. TeamBots however has not been updated since 2000 and over time documentation for it is has disappeared and fallen apart, rendering it extremely hard to learn.

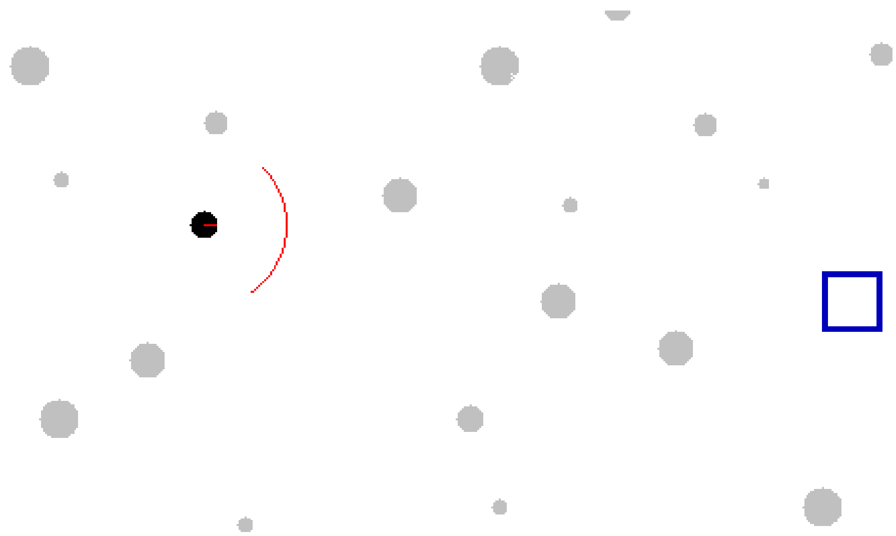

Figure 1: A simple robot simulation in TeamBots.

b. Gun-Tactyx

Gun-Tactyx is the newest of the three main robot simulation platforms that were considered and unlike TeamBots and Robocode which are two dimensional (2D) simulations, Gun-Tactyx is a three dimensional (3D) simulation. Although it was originally conceived as a simulation of robots battling each other, it offers significant flexibility and has been extended for uses such as robot soccer. However, it is not well documented; compounding this was the problem that it required that robots be written for it in a non-standard language. This would have precluded the use of existing neural network software.

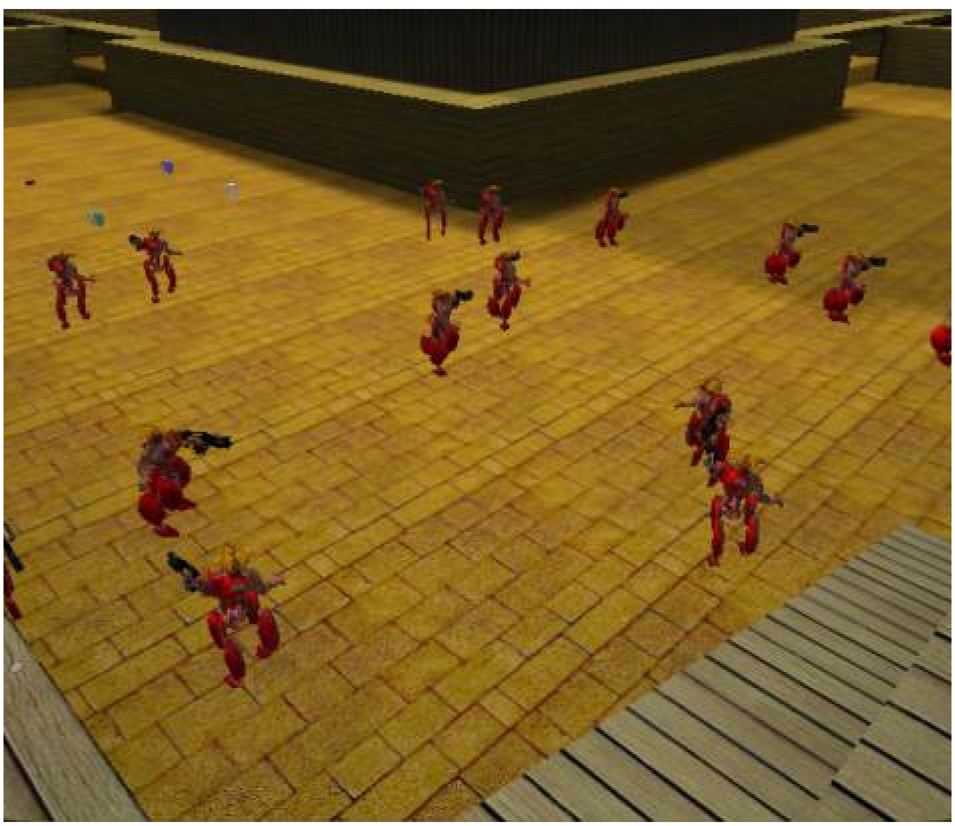

Figure 2: A robot simulation in Gun-Tactyx.

c. Robocode

Robocode is the robot simulation platform that was ultimately chosen for use. It is a robot simulation package intended for simulating robots battling each other. It was started by Mathew Nelson at IBM [6]. After being maintained by IBM for several years, it was made into an open source project [28], and continues to be regularly updated.

One major drawback to using Robocode was that it was less flexible than either TeamBots or Robocode in allowing a foraging scenario to be implemented, instead of a combat scenario. Robocode is implemented in Java, and robots are written using a well documented Java API [29]. The use of a major language and level of documentation allowed for a foraging scenario to be implemented, but not to be displayed graphically.

Robocode is implemented purely in Java, and the robots are also written in Java. As Robocode was the only platform of the three that was both fully implemented in a single, major language and also had robots which were implemented in a single, major language, Robocode seemed most likely to work well with already existing neural network software. Unfortunately, it was later found that because of security features built into Robocode, it was harder to use existing neural network software than was originally thought, although it was accomplished.

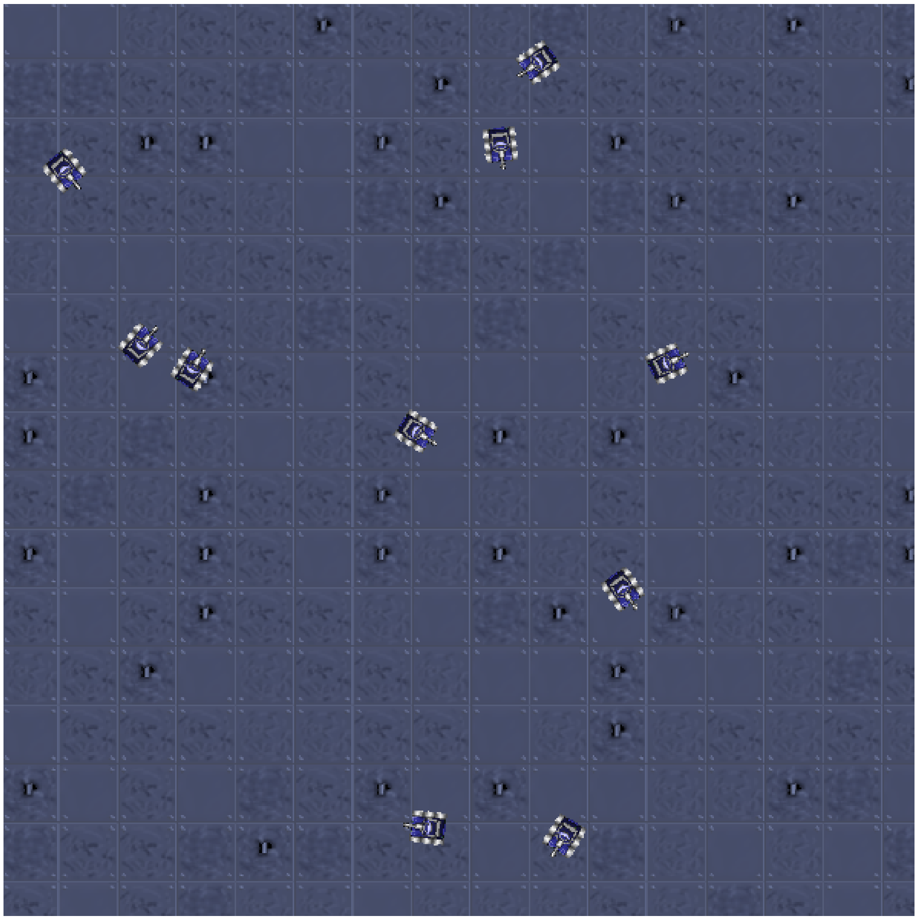

Figure 3: A simple view of robots in Robocode.

II. JOONE

After the Robocode was selected as the robot simulation platform, a search of available neural network packages implemented in Java was conducted. The search revealed that there was one popular, maintained, well documented, powerful and flexible neural network software package available. This package is called the Java Object Oriented Neural Engine (JOONE) [30].

A number of available Java neural network packages are GUI based only. While JOONE does allow for setting up a neural network via a GUI, it also allows for a neural network to be set up via a Java API. Additionally, not all Java based neural network packages are free and allow for alteration of the code that runs the neural network. JOONE is Open Source Software, which became an important factor as some adjustments to the code were necessary to permit the code to run in conjunction with Robocode.

B. Experiments

There were two main phases to conducting the experiments. The first was setting up the foraging environment and the second was developing the programs that guided the robots behavior.

I. The Map

The map used was of size 1000 pixels by 1000 pixels, with robots being 18 by 18 pixels in size.

The map consisted of three parts: (1) a home base at which robots can return to drop off food and restore their energy reserves, (2) food sources with limited amounts of food that can be collected from and eventually become depleted, and (3) obstacles that slow down the robot, or damage parts of the robot. In this thesis we consider obstacles to be rough terrain that may slow down and damage a robot, but does not completely stop a robot.

A small program was written that randomly placed specified numbers of food sources and specified numbers of obstacles on the map. Each map was written to a file, and loaded for experimentation with different kinds of robots and varying numbers of robots.

a. Food Sources

In the foraging problem, robots must collect food. Food sources were implemented as randomly placed points throughout the map. Each food source starts with a randomly selected amount of food between 0 and 5 (continuous) units. Robots can collect up to 1 unit of food, but the amount of food collected is often less because a robot's ability to collect and carry food can be damaged, resulting in the ability to collect less food than this. Additionally, robots may be already carrying a partial load of food (such as if the robot went to another food source and the food there ran out), which would reduce the amount of food that the robot can collect from another food source. Once a food source has been depleted by having the amount of available food reduced to 0, that food source is ignored by the robots.

b. Obstacles

The experiments involve a team of robots' ability to not only decide what food sources to select from, but also to get to those sources of food, and preferably via a path that uses less energy and causes less damage.

When a robot attempts to go through an obstacle, the robot is temporarily slowed down by an amount depending on what type of obstacle is involved. In addition there is a chance that the robot may suffer permanent damage. There are three types of damage that a robot may suffer: (1) damage to the engine, (2) damage to the scanner, and(3) damage to the food collector. When a robot suffers damage to the engine, the robot will take longer to cover a specified distance than when not damaged. In addition, more energy will be used to travel this distance. Eventually a robot's engine may become so damaged that the robot becomes disabled and unable to move. When a robot's scanner becomes damaged, the distance at which it can spot a new food source is reduced. When a robot's collector becomes damaged, the amount of food that a robot can collect and carry is reduced.

There are four types of obstacles: mud, sand, thick brush, and snow. Mud results in high chance of engine damage, medium chance of scanner damage, and low chance of collector damage. Sand results in medium chance of engine damage, medium chance of scanner damage, and medium chance of collector damage. Thick brush results in medium chance of engine damage, medium chance of scanner damage, and high chance of collector damage. Snow results in low chance of engine damage, medium chance of scanner damage and medium chance of collector damage.

II. The Robots

Three different robots were simulated to work in a foraging environment. These were:

- A basic robot
- A robot that uses a contract net without adaptive bidding (using a first come-first serve strategy)
- A robot that uses a contract net with adaptive bidding (using a neural network to estimate the amount of energy required and probable damage)

a. Teams

In order to compare the performances of each type of robot, homogeneous teams of robots were simulated operating in identical environments, where only the starting positions of the robots were changed. The reason for the change in starting positions is that robots in Robocode are placed in random locations at the beginning of each simulation. To account for this, averages were found over 10-20 simulations for each type of robot, for each team size. At no point were robots of different types placed on the same team and at no time were multiple teams simulated at the same time.

The size of the robot teams was varied from three to fifteen robots. Each robot in Robocode is run as its own thread. Each robot that used adaptive bidding used the JOONE neural network package, in which each node is run as its own thread. For these reasons, when using adaptive bidding, there was an explosion of threads trying to run at the same time. Thus memory problems arose when using more than 15 robots.

b. Communication

A current problem in mobile robotics that particularly affects the Contract Net Protocol is limitations in the communication capabilities between robots [1][8][10]. In this thesis, we relaxed the current practical communication problems in the view of examining the usefulness of using the Contract Net Protocol with adaptive bidding at some point in the future when communication capabilities between robots have been improved. In the experiments, Robocode's built in communication tools were used that allow a robot to communicate with any other robot that is on the same team, or allow a robot to communicate with every other robot that is on the same team (broadcast).

c. Criteria for Success

There are several different criteria used for judging the success of task allocation problems. Gerkey and Mataric [11] considered there to be two different criteria, quality and cost. One measure of cost is often the amount of energy expended in completing a task.

Quality can be less defined and may have task specific definitions, such as in the case of the foraging problem, the amount of food gathered can be considered part of the quality. Time required for completion of a task is also sometimes considered part of the quality of a completed task [11], but can also be considered part of the cost of task completion. In the case of this thesis, we consider it as a separate criterion from the cost (energy expended), or quality (which is task dependent). In the experiments we are imposing a time limit on each simulation so as to be able to perform numerous simulations. This means that time could not be used as a criterion for success.

Therefore, energy consumption and the amount of food gathered are used as criteria for success. There are a number of cases in which success may be considered. There is individual robot consideration, in which a robot evaluates its own performance after a task has been completed, so as to be able to then determine how it should bid in the future. There are also end-of simulation considerations. In end-of simulation considerations, all experimental data is known, and conclusions can be drawn for a specific task as to whether it is better to maximize time or energy, or find a threshold for tradeoff. However there is an initial paradox for individual robot's considerations. Individual robots during simulation will not know whether energy or food is more important. For this reason, it is best to have robots heuristically try to maximize only energy or food. For this project, robots will be attempting to maximize food gathered.

d. Basic Robot

A basic robot was created as a control group. This robot followed a greedy strategy with no task allocation strategy, but with information sharing.

For a fair comparison of robots, all robots should have the same capabilities, including communication. This means that reasonable communication can be established between the basic robots. Sharing of information about food source locations was included as it was thought likely to improve team performance without having a task allocation strategy.

Initially, the basic robots do not know where food sources are. The first task of the basic robot is to explore the map searching for food, until it finds a food source to collect from.

Once a food source is found, the location and amount of food at the food source is communicated to other robots, and food is collected. If the food source has more food than the robot can carry, then the robot will take as much food as it can carry and proceed back to the home base. If the robot is able to carry more food than can be collected from the food source, the robot takes the available food and communicates to other robots that the food source has been depleted. In this case, if the robot knows of other food sources, it will proceed to another food source and will collect food from there. If the robot does not know of other food sources, then the robot will again search for food sources.

At every point in which the robot travels, an estimation is made of how much energy the robot requires in order to return to the home base. This estimation is based upon the Euclidean distance between the robot and the home base, with a multiplier included to help account for obstacles. Certain energy is needed to account for the time needed for making the calculation and responding to the calculation. If at any point the robot's energy level drops below this estimated energy level, all other actions of the robot are suspended and the robot immediately heads to the home base, regardless of whether the robot is searching or going to a known food source.

Every time a robot returns to the home base, it determines what its next task should be, whether searching for new food sources, or collecting food from a known food source. The basic robot follows a greedy strategy and will collect food from the closest food source.

While collecting food or returning home, a robot may encounter previously unknown food sources. In this case, the robot will record the location of the food source; and it will alert other robots as to the location of the food source and amount of food it contains. If the robot is returning home, then it will continue returning home, and after that if the discovered food source is the nearest food source, then the robot will collect from there. If the robot is on its way to collect food from another food source, the robot will continue to that other food source. Upon the robot returning to the home base, or the robot depleting the food source and being able to carry more food, the robot will determine the nearest food source, which may be the newly discovered food source, and will collect food from that food source.

While exploring, collecting food, or returning home, a robot is continually scanning for obstacles in its path. For the basic robot, when an obstacle is found ahead, the robot will look towards the left, and look towards the right. If in either direction, there is no obstacle, then the robot will proceed in that direction for a distance and then the direction needed to get to its destination, if known. If an obstacle is observed in both directions, then the robot assumes that there is no better path to its destination, so proceeds forward in the original direction.

Additionally, robots can take damage from running into the walls surrounding the environment, and from running into other robots. When the robot detects a wall in its path, it slows down or stops and readjusts its direction. Similarly, when a robot detects another robot in its path, it will slow down or stop, then changes direction to the left to try to go around the robot. This is a form of ‘social law’ that allows robots to generally avoid each other [31].

e. Contract Net Without Adaptive Bidding Robot

Another type of robot that was developed was a robot that uses the Contract Net Protocol to coordinate tasks among robots, but without adaptive bidding.

Initially, robots of this type have no knowledge of food sources, so similar to the basic robot, they begin by exploring the environment until a food source is found.

Upon finding an unknown food source while exploring for food, a robot communicates to other robots the location of the food source. Other robots consider that food source as belonging to the robot that discovered and communicated the location of the food source. These other robots will not forage food from this food source. The robot that discovered the food source will then forage food from this food source.

When while collecting food or returning home a robot discovers an unknown food source, that robot will communicate to other robots where the food source is. Other robots then will not forage food from this food source. As the robot is in this case working on foraging from another food source, it will continue to collect food from the food source that it had been collecting from, not from the newly discovered food source. The robot will however let other robots know that it has a task to be performed by other robots and invites other robots to bid on the task. If another robot is in the act of searching for a food source, it will bid on the task. For robots that used the contract net without adaptive bidding, the robot that discovered the food will accept the first bid that it receives.

Just as robots that find new food sources while harvesting from other food sources do not abandon their original tasks, robots do not give up on a task that they have been given by another robot. That is to say that after a robot has accepted a contract, it may not decommit [35] [36] from that contract.

If a robot is foraging from a food source, and discovers a new food source, it will advertise for bids for the task of foraging from the newly discovered food source. This does not mean that the robot will necessarily receive any bids for the task. If the robot depletes the food from the food source it is currently working on, i.e., it finishes its task, if it has discovered a food source and advertised for bids, but not received any and given the task to another robot, then the robot that discovered the food source will start collecting from that newly discovered food source and will no longer advertise for bidders for the task.

As with the basic robot, the robot that uses the Contract Net Protocol without adaptive bidding continually keeps track of an estimate of the energy level at which if the robot drops significantly below this level, it may not have enough energy to return home and replenish its supply of energy. Just as was the case with the basic robot, if at any time the robot's energy level drops below this estimate, the robot will immediately suspend its current activity and proceed home.

The robot that uses the Contract Net Protocol without adaptive bidding behaves in the same way as the basic robot when the robot detects a wall or robot ahead.

f. Contract Net With Adaptive Bidding Robot

The final type of robot developed was the robot that uses the Contract Net with adaptive bidding. Similar to the basic robot and the robot that uses the Contract Net Protocol without adaptive bidding, the robot that uses the Contract Net Protocol with adaptive bidding starts without knowledge of where food sources are located, and has an initial task of exploring the environment to try to find new food sources.

Once a new food source has been found, the robot that discovered the food source sends out a message to other robots alerting them to the location of the new food source. The other robots then will not take food from that food source, just as with the robots that did not use adaptive bidding. If the robot that discovered the food source was searching for food, then that robot will take the task of collecting food from that food source. If the robot that discovered the food was collecting food from another food source at the time, then that robot will continue collecting food from its original food source and will advertise for bids from other robots on the task.

III. The Bidding Process

The robot that uses the Contract Net Protocol without adaptive bidding accepts the first bid that it receives. With adaptive bidding, bids have values that reflect how efficiently the robots that make the bids believe that they can carry out the task. Since there can be differences in the bid, the robot that advertises for bids will have to consider multiple bids. However, since the robots are run in separate threads, there can be some variance in the amount of time it takes for a robot to process a bid request and send a bid. For this reason, when using the Contract Net Protocol and not giving the task to the robot that sent the first bid, one of two strategies must be employed for determining when to stop accepting bids: (1) placing a time limit after which the robot with the best bid is awarded the contract to do the task, or (2) establishing a threshold (a reserve price in the auction terms) for which the first bid to meet the threshold is awarded the contract. Often there is a time limit built in as well, in case no bid meets the threshold. In order to establish a threshold however, a robot needs to be able to estimate a reasonable cost for accomplishing a task, which requires knowledge about other robots. Thus, in order for a threshold to be used, the robot requesting bids must have significant knowledge of the likely cost of accomplishing a task. Yet it can not be assumed that any robot has such significant knowledge. In fact, the collective knowledge of the robots offering bids is likely to be greater than the robot requesting a bid, and so it will be better to only use bids with a time limit without using a threshold. In this thesis a robot that uses the Contract Net Protocol with adaptive bidding that request bids waits 300 milliseconds for bids to be submitted. At the end of this time, the robot that requested the bids sends a message to the robot with the best bid, informing it that it has been given the contract to carry out the task. When a robot submits a bid, it waits for 300 milliseconds. If it has not received a message indicating that it has been awarded the contract, then it continues searching for new food sources or submits a bid for a new contract. A robot may not have two bids submitted at the same time.

As the robots are assumed to be honest robots, and they are ultimately homogenous, each robot that contracts out a job has no need of knowing the exact capabilities of robots that submit bids, but only needs to concern itself with finding the best bid. This allows for robots that only have knowledge of their own individual problems, without need for greater knowledge of the social system as a whole [32][33][34].

IV. Formation of Bids

Obstacles affect how efficiently a robot can carry out a task, whether from requiring more energy to go around an obstacle, or requiring more energy to go through an obstacle, as well as sometimes damaging robots that go through obstacles. This damage also affects how well a robot can carry out its tasks.

Initially however, robots using the Contract Net Protocol with adaptive bidding have no knowledge of where the obstacles are or what types of obstacles they are. This means that initially these robots cannot make reasonable bids. For this reason, robots initially revert back to the behavior of the robots that use the Contract Net Protocol without adaptive bidding, with robots that have requested bids awarding contracts to the first robot to make a bid. After a robot has gathered knowledge on 10 ‘paths’, it assumes that other robots have likely gathered some initial knowledge of the environment as well, and thereafter the robot switches to waiting after sending out bid requests and accepting the best bid as opposed to the first. Furthermore, the bid request is altered to indicate to other robots that only robots with this basic knowledge of the environment should submit bids.

For determining paths, a robot starts by making a record of its state, including its position, energy, level of damage, and current speed. After the robot has traveled a short distance, its new position, energy, level of damage, and current speed are recorded and compared with the earlier records. After 10 such paths were recorded, a neural net was trained with these 10 paths as training data. The coordinates acted as input data, while the rest of the data acted as expected outputs. In this way, the neural net was trained to try to learn about the environment, i.e., learn the map.

The initial results did not show a clear improvement with robots set up using this approach. It was surmised that the cause of this was the neural network was trying to learn too many pieces of information. The robots were then altered to not use the levels of damage and current speeds as part of the expected output data, but instead to only use the difference in energy as the only piece of expected output data.

V. Network architectures

In order for a robot that uses the Contract Net Protocol with adaptive bidding to have any potential to outperform a robot that uses the Contract Net Protocol, it must have the ability to adapt quickly to make reasonable bids. For this reason, numerous neural network architectures were examined for use, starting with the simplest architectures, and progressing to more complex ones.

The initial type of network architecture had one hidden layer. The number of nodes in this hidden layer was varied between two and sixteen nodes.

Neural networks have a learning rate, which varies the rate at which the network tries to learn a pattern. The reason for this is to avoid cases where the network would try to learn too quickly and would be less likely to stabilize. “The convergence speed of backprop is directly related to the learning rate” [37]). The learning rates tested were 0.4, 0.5, 0.6, 0.7, and 0.8. The best learning rate was found to be 0.7.

Initial results were not especially promising, so a second layer was also used. When using two hidden layers, the number of nodes used for each hidden layer was varied between two and 10 nodes. The reason for the difference in the highest number of hidden nodes tested was because of the significantly higher number of weights involved when two hidden layers. For example, with four inputs, when using one hidden layer with 16 nodes and 5 output nodes, there are 4 x 16 + 16 * 5 = 144 weights. With four inputs, when using two hidden layers with 10 nodes each and 5 output nodes, there are 4 x 10 + 10 x 10 + 10 x 5 = 190 weights. Thus even though the maximum number of nodes per hidden layer when using two hidden layers is lower, the number of weights is larger than when using only one hidden layer. Additionally, each robot has its own neural network, further expanding the number of weights being kept track of, and each of these weights is updated multiple times. For these reasons, the number of nodes has to be kept relatively low.

Additionally, the number of epochs (the number of times that errors were calculated and weights were adjusted), had to be kept moderately low in order for the network to be quickly trained. The number of epochs chosen was 200, as this resulted in a quick training time.

Early results did not show significant improvements with a robot using the Contract Net Protocol with adaptive bidding through a neural network with 5 outputs over a robot that used the Contract Net Protocol without adaptive bidding. Initial analysis suggested that the network was being trained to try to learn too many outputs. The networks were altered to only have one output: the energy expended on a path. Networks with both one and two hidden layers were then reexamined.

VI. Team Size

This thesis is ultimately an examination of team performance. One important variable then was the team size. The size of the teams were limited to a maximum of 15 robots because of memory limitations. The size was reduced incrementally by 3 robots.

Each network architecture was tested with each team size in order to examine how the architecture in conjunction with the team size affected the results.

VII. Computing

This experiment was run on 2.8 GHz Pentium 4 with 1024 MB RAM on Windows XP, with Java 6(u1), Robocode 1.2, and JOONE 2.0.0RC1. Robocode has fairly strong security features which make using external packages somewhat difficult. Therefore JOONE had to be repackaged and some of the source files had to be adjusted to accommodate repackaging.

4. Experimental Results

A. Basic Robot

Five team sizes comprising of robots of the basic robot type were used. The teams consisted of 3, 6, 9, 12, and 15 robots. Each team was simulated twenty times and the results were averaged together.

I. Energy Usage

The energy was initially examined for the Basic Robot. One unit of energy is defined as the amount of energy required for a robot to move one pixel, when the engine is working at full capacity (no damage), and not in any obstacle (slowing down the robot).

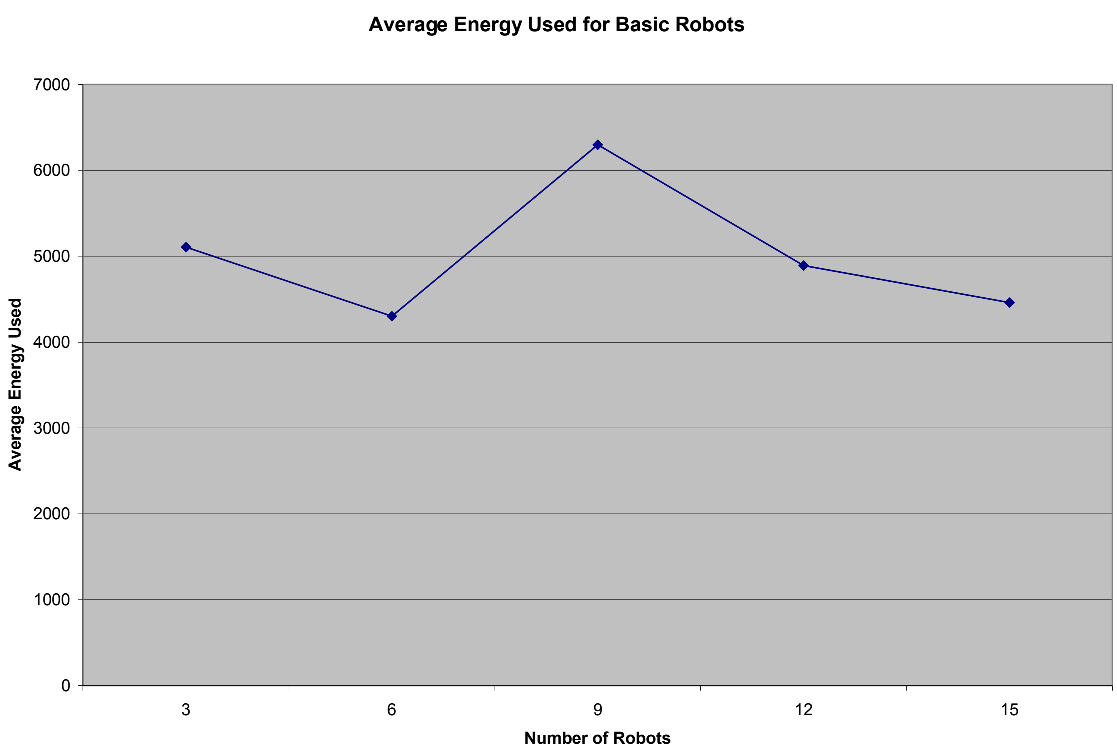

Figure 4: Per Capita Average Energy Used for Basic Robots

The average amount of energy used by the teams with 3, 6, 12, and 15 robots was very similar, with the amount of energy used by the team with 9 robots was slightly higher, but this can likely be attributed to simple variation and not necessarily indicative of a trend.

II. Food Gathered

Food sources had between 0 and 5 units of food. A robot can collect up to one unit of food before having to return back to the home base. However this is a maximum amount, and the robot's collector may become damaged, which will reduce the amount of food the robot can collect from this maximum.

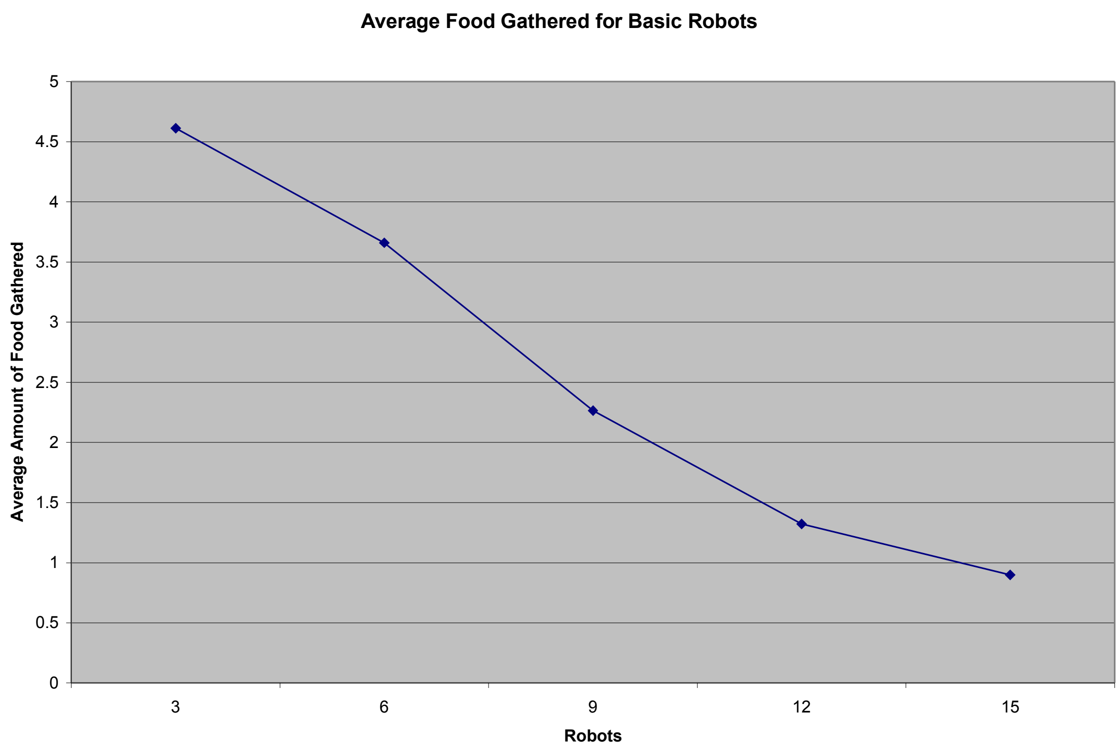

Figure 5: Per Capita Average Food Gathered for Basic Robots

There is a clear trend involved in the average amount of food collected by teams consisting of the basic robot, with the average amount of food collected being greatest when using the fewest amount of robots, and the average amount of food collected continually decreasing, with the team with 15 robots having the lowest average collected food. Observations showed that the greedy strategy of the robots, in conjunction with information sharing, led to robots concentrating in small areas and having difficulty moving around each other.

III. Damage

The damage is a measure of the performance of the three main components, the engine, the scanner, and the collector. In Figures 6, 9, 14 and 17, damage is represented by the average performance of these three components at the end of the simulations as compared to the performance at the beginning of the simulation. For example, at the beginning of a simulation, it takes one unit of energy for a robot to move one pixel in distance. If the robot has an average end efficiency of 20% for its engine, then it will take 5 units of energy to move one pixel in distance at the end of a simulation. A 20% end efficiency for the collector means that a robot can only carry 20% as much food per load as when the simulation began. Similarly, a 20% scanner end efficiency means that a robot can only scan 20% of the distance ahead as when the simulation began.

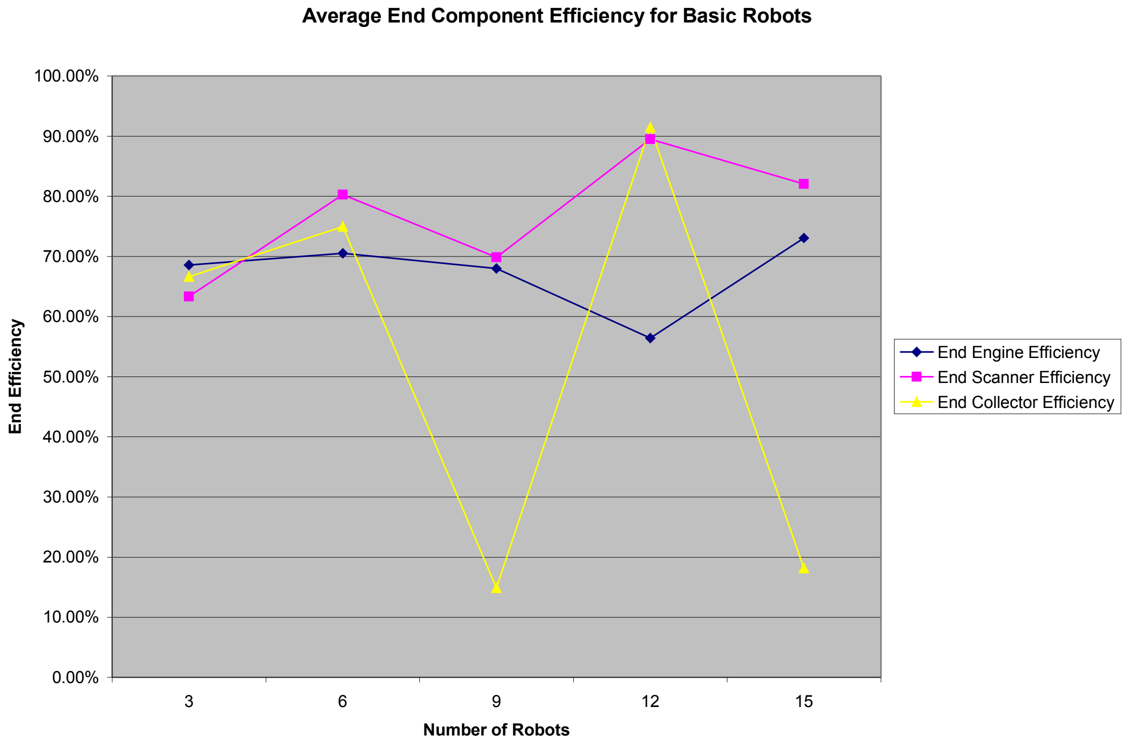

Figure 6: Per Capita Average End Component Efficiency for Basic Robots

Figure 6 shows a high amount of variability, with few to no significant trends. This information is presented as it was originally thought that damage might be an important indicator of the success of a robot, although this was not found to be the case.

B. Contract Net Without Adaptive Bidding Robot

I. Energy Usage

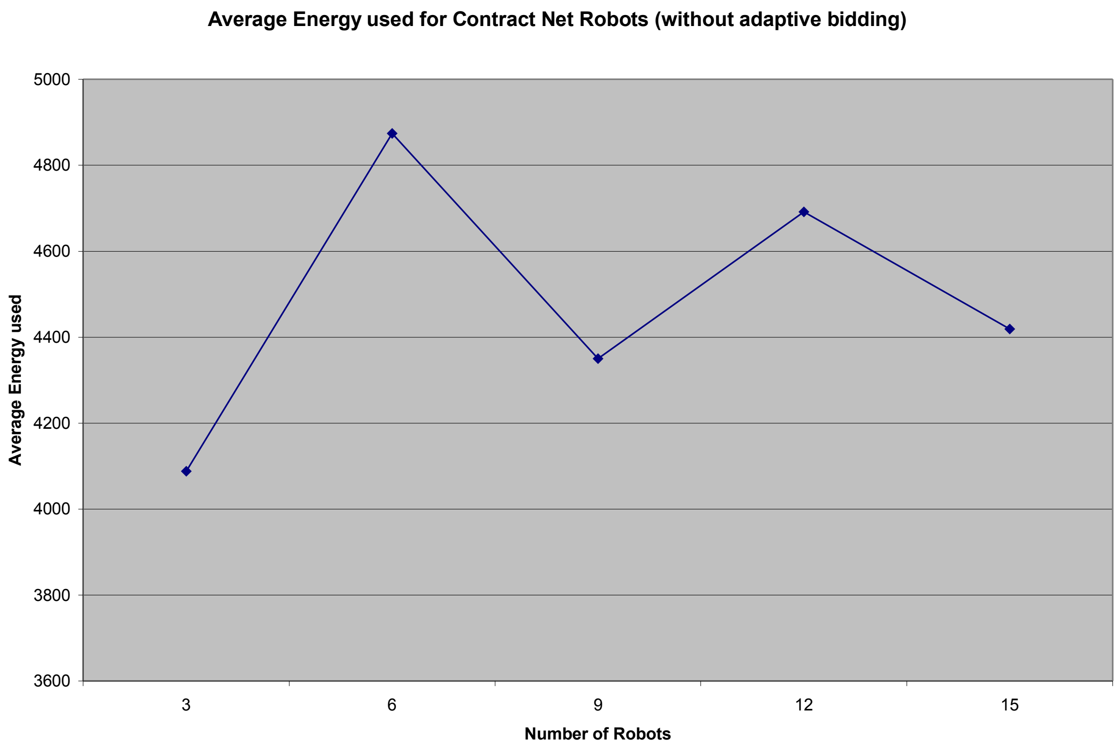

Figure 7: Per Capita Average Energy used for Contract Net Robots (without adaptive bidding)

The energy usage again did not show significant trends for teams consisting of robots that used the Contract Net Protocol without adaptive bidding.

II. Food Gathered

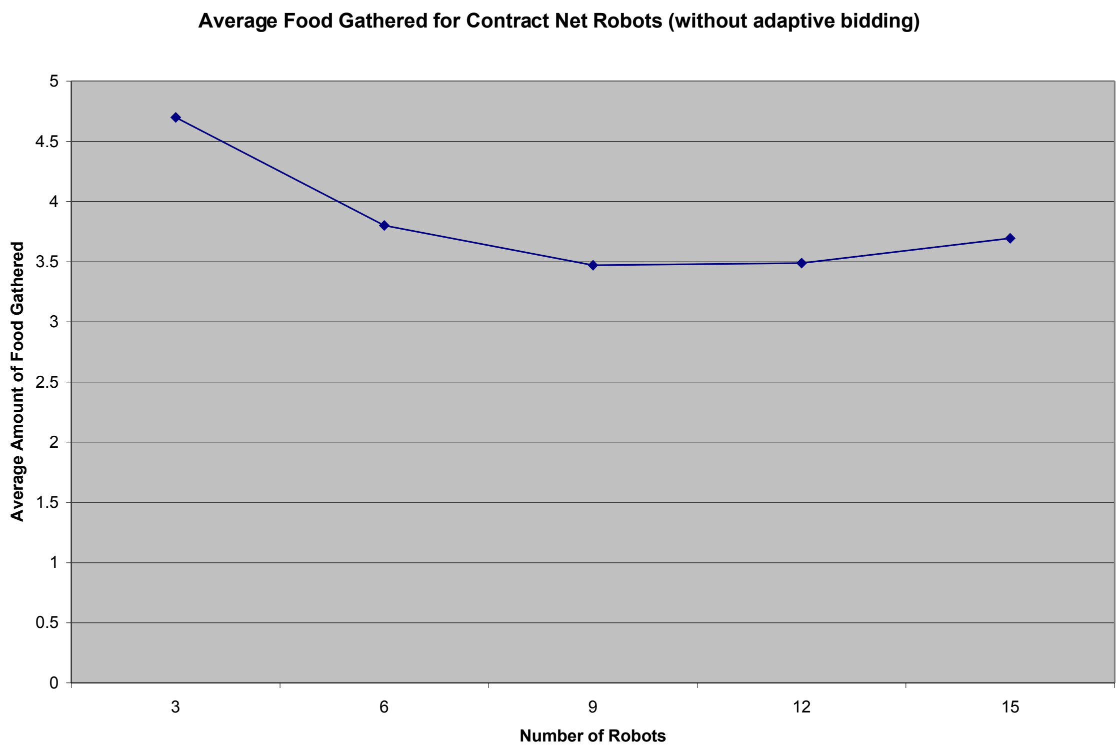

Figure 8: Per Capita Average Food Gathered for Contract Net Robots (without adaptive bidding)

In Figure 8, the average food gathered for teams consisting of robots that did use the Contract Net Protocol without adaptive bidding did once again show a significant trend. The average amount of food gathered decreased with greater numbers of robots, but plateaued once 9 or more robots were used on a team. There was a slight rise at 15 robots, but this may still be a part of the plateau, but just be a small random variation.

III. Damage

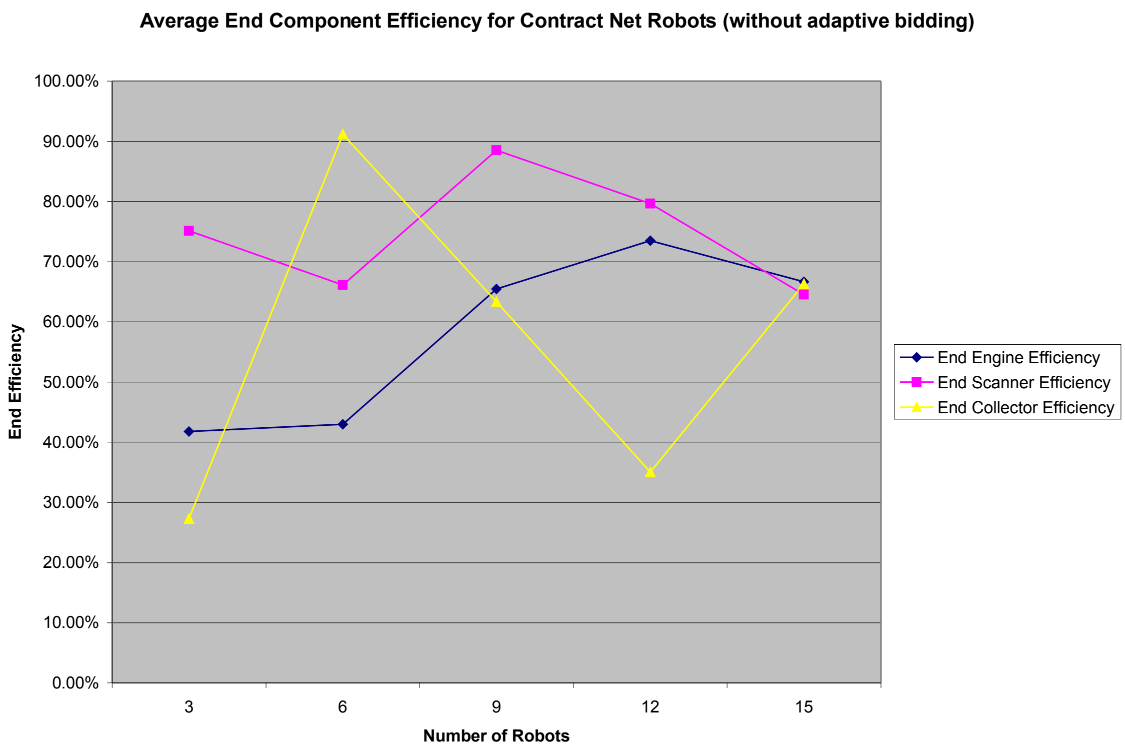

Figure 9: Average End Component Efficiency for Contract net Robots (without adaptive bidding)

There seems to be a fairly high level of variation once again, with the level of damage to the Engine and Scanner being more stable than the level of damage to the Collector, with the damage to the Engine again being more severe than the damage to the Scanner. The amount of damage to the Collector showed was once again highly varied.

C. Contract Net With Adaptive Bidding Robot

After the Basic Robot and the Robot using the Contract Net Protocol without adaptive bidding were tested, the robot using the Contract Net Protocol with adaptive bidding was tested. It used a neural network in order to adaptively formulate bids. Initially the network had 5 outputs: the energy consumed, the current speed, and the damage to the three components of the robot. Initial experiments showed that the number of nodes used in the hidden layers did have an impact on the amount of food gathered (as shown in Figure 10), although the addition of extra nodes increasingly varied the results.

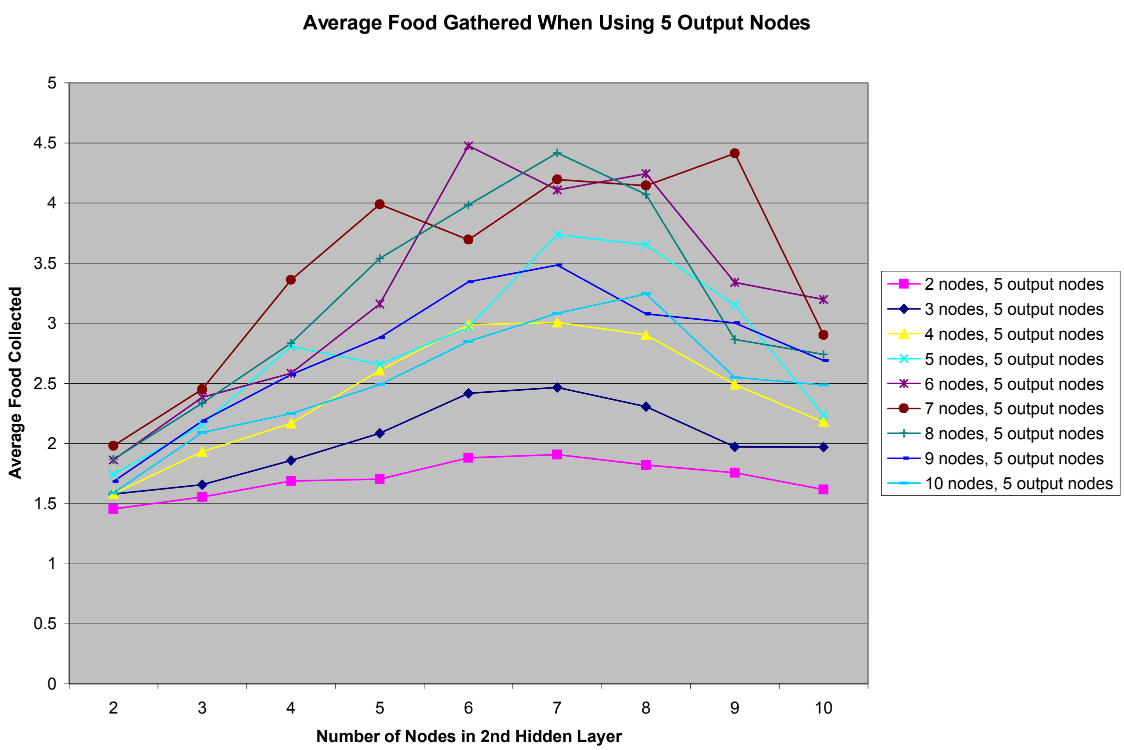

Figure 10: Per Capita Average Food Gathered When Using 5 Output Nodes

In Figure 10 each line represents a certain number of nodes being used in the first hidden layer, while the x axis represents the number of nodes in the second layer, and the y axis represents the amount of food gathered by robots using this network architecture. The amount of food is averaged for teams of each of the five team sizes.

Ultimately however, it could not be established with confidence that the robots using the best network architecture (6 nodes in each of the two hidden layers) was significantly different than when not using adaptive bidding. An initial suggestion was that this might be due to the network trying to learn too many different types of information, so the expected outputs were reduced to one piece of information: the amount of energy used.

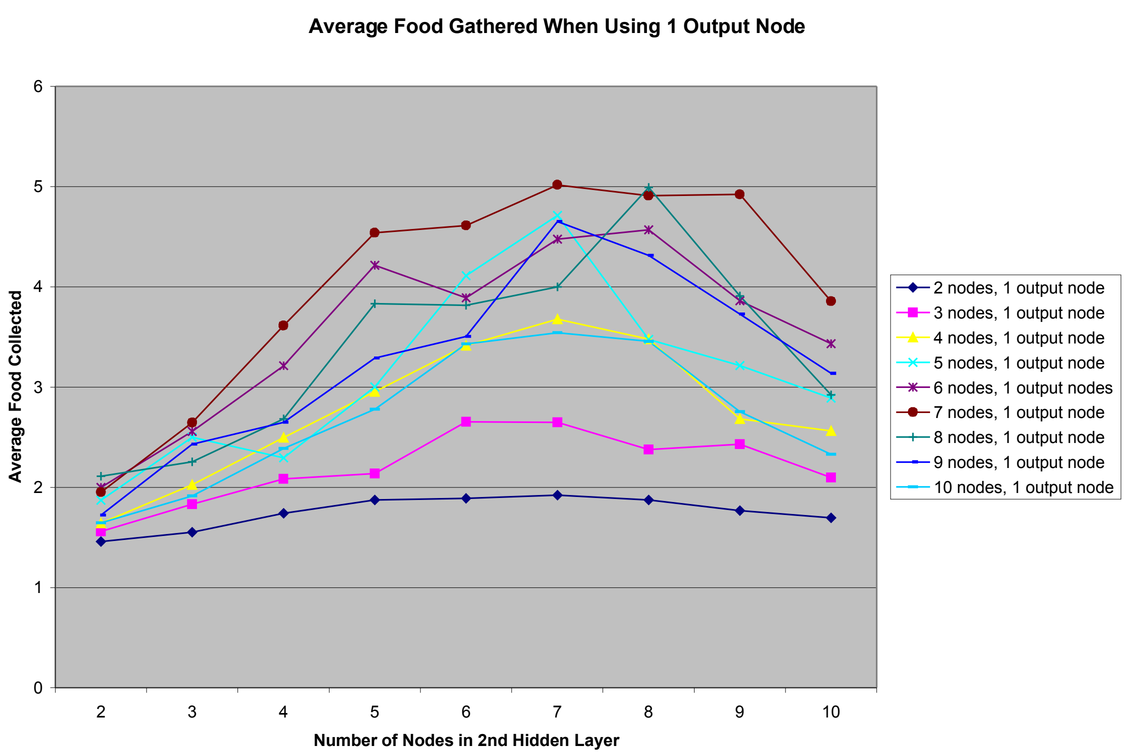

Figure 11: Per Capita Average Food Gathered When Using 1 Output Node

The benefit of only using energy as the output was not a huge difference, but it was enough so that there was reasonable confidence in the performance of the best architecture (8 nodes in each hidden layer) over the performance of robots using the Contract Net Protocol without adaptive bidding.

I. Using Five Output Nodes

First, the performance of robots using the best architecture (when using five outputs) is examined.

a. Energy Usage

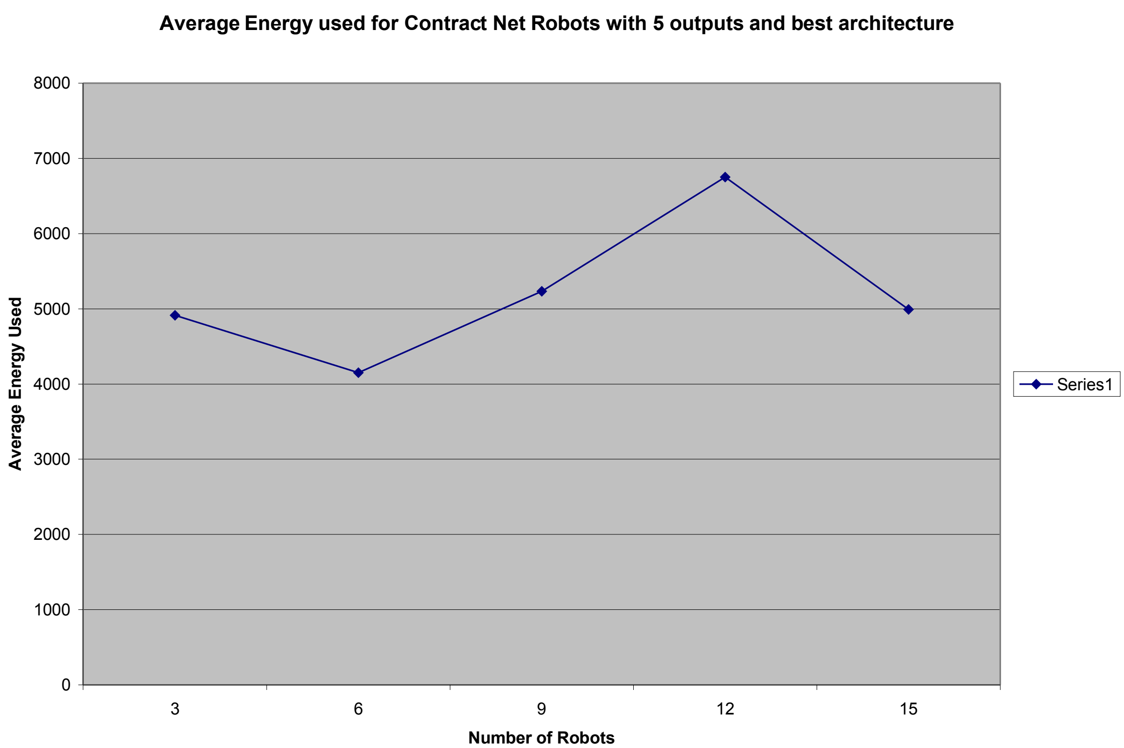

Figure 12: Per Capita Average Energy used for Contract Net Robots with 5 outputs and best architecture

As has been the case, the amount of energy used has not shown much of a trend, but at the same time it has kept generally kept within the 4,000-7,000 range. This is likely due to the time limit imposed on simulations. When a robot moves through rugged terrain, the robot is slowed down and expends more energy than would be the case if moving at that speed in clear terrain. If the robot does not go through harsh terrain, then it will likely get to its destination quicker and start moving toward a new destination, continuing to spend energy. Thus because the experiment is time limited and does not run until every bit of food is collected, the robots are constantly expending energy and the amount of food gathered is more indicative of success.

b. Food Gathered

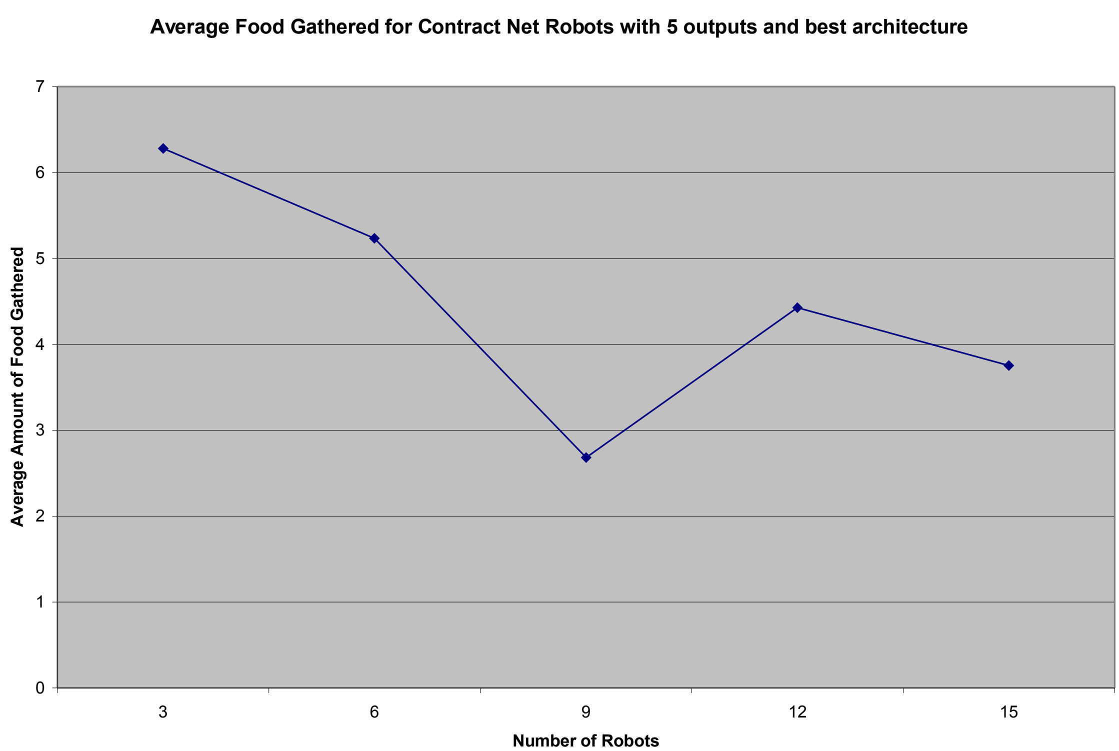

Figure 13: Per Capita Average Food Gathered for Contract Net Robots with 5 outputs and best architecture

One especially ominous sign for robots using the Contract Net Protocol with 5 output nodes was that although the amount of food gathered seemed to be higher than for when using robots without adaptive bidding, there was nonetheless a trend of each robot gathering less and less food when more robots were introduced.

c. Damage

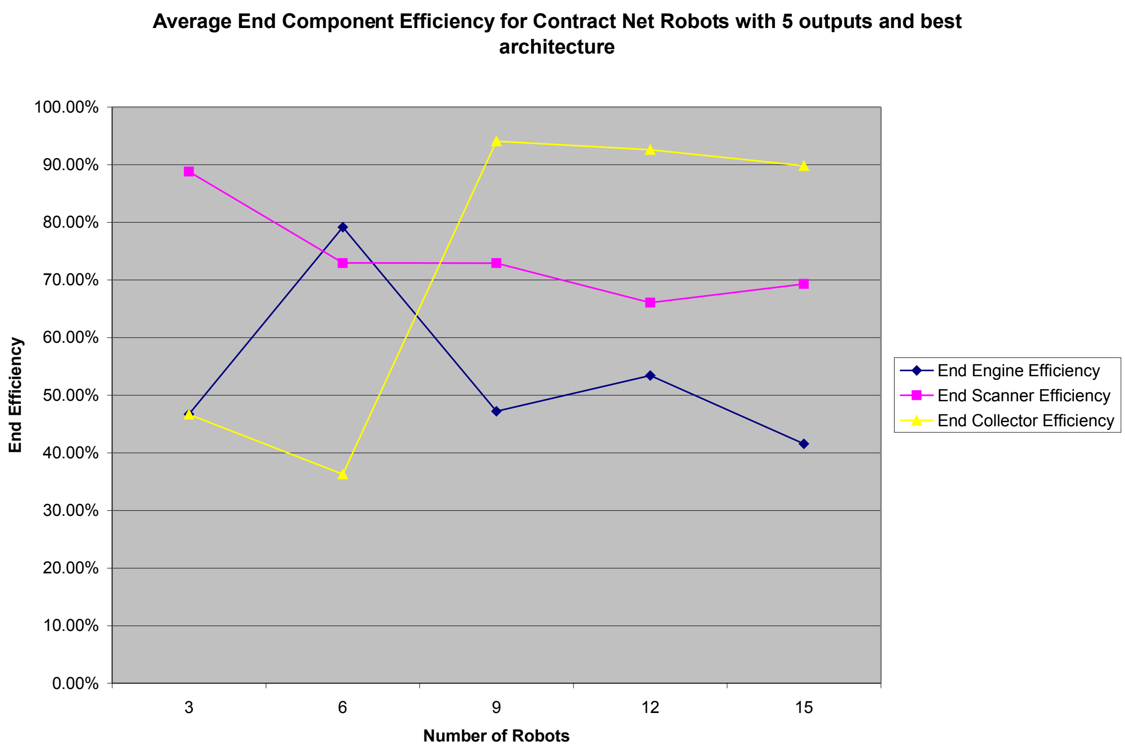

Figure 14: Per Capita Average End Component Efficiency for Contract Net Robots with 5 outputs and best architecture

In Figure 14, the amount of damage to the components was yet again wildly variable.

II. Using One Output Node

a. Energy Usage

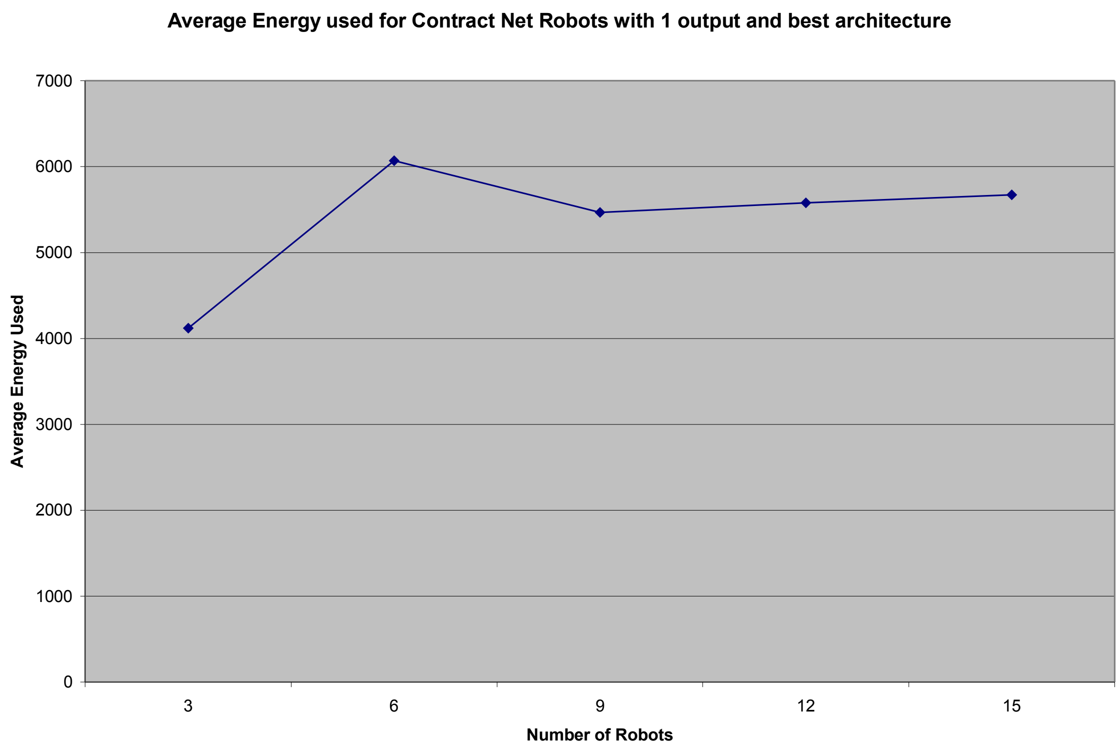

Figure 15: Per Capita Average Energy used for contract Net Robots with 1 output and best architecture

The amount of energy used was consistent when using only one output.

b. Food Gathered

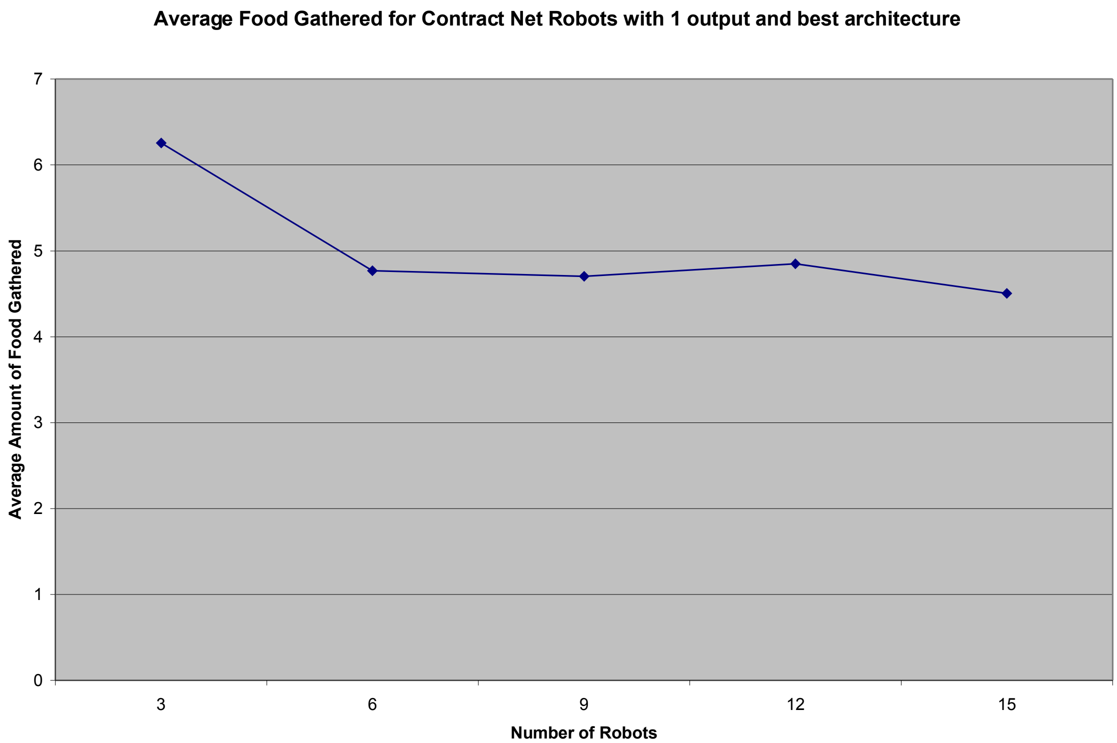

Figure 16: Per Capita Average Food Gathered for Contract Net Robots with 1 output and best architecture

Although the amount of energy seemed to remain high, there was the benefit of the amount of food gathered being fairly high, and remaining fairly stable, with a plateau after a short drop between when 3 and 6 robots were used.

c. Damage

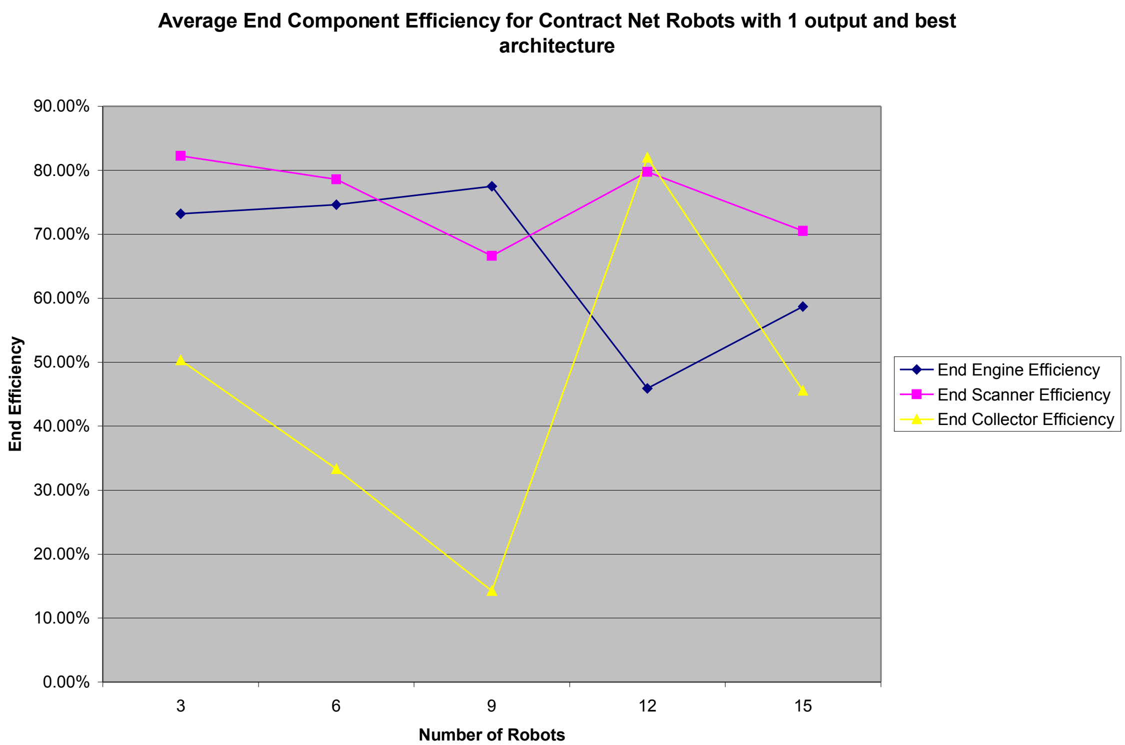

Figure 17: Per Capita Average End Component Efficiency for Contract Net Robots with 1 output and best architecture

Although the amount of energy and food gathered seemed fairly stable when using one output, the end damage to the robots showed no such stability.

III. Five or One Outputs

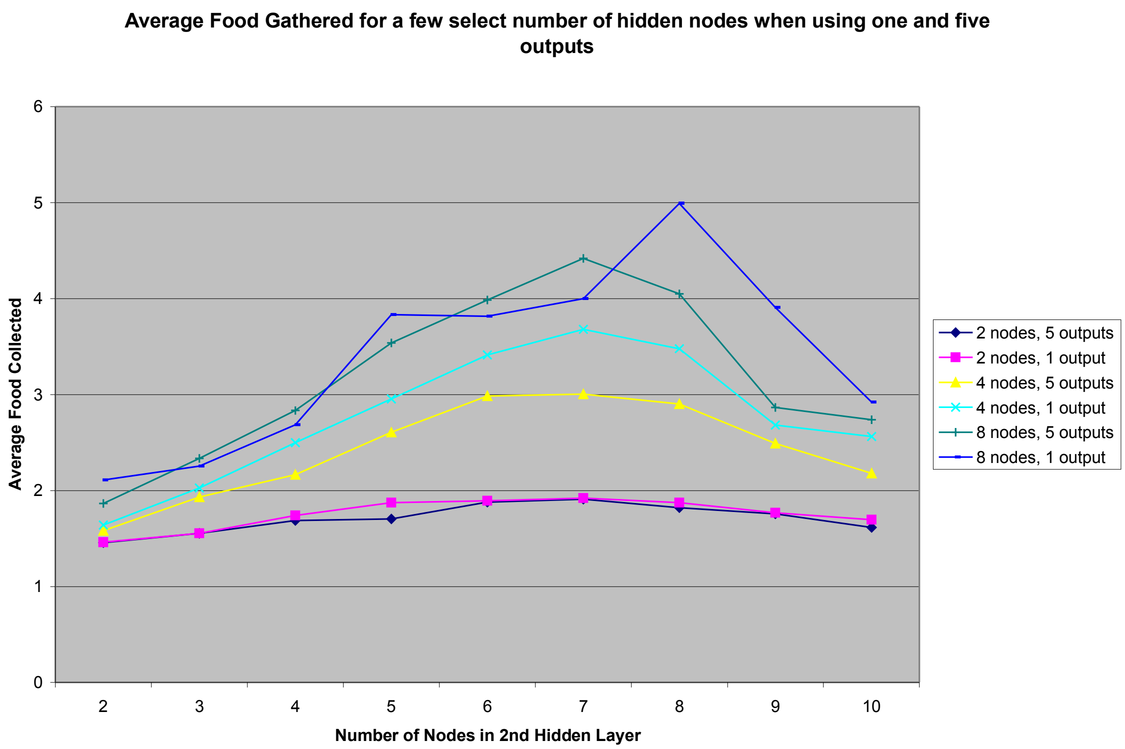

Figure 18: Per Capita Average Food Gathered for a few select number of hidden nodes when using one and five outputs

Figure 18 shows a few select points that show how initially, when using few nodes in the hidden layers, the performance difference between when using one and five outputs is quite small. As the number of nodes increases, the benefit of using only one output becomes more pronounced, however when the number of nodes rises further, although using one output ultimately produces the best results, there are more variability in the results so that in some network architectures it is better to use five outputs instead of one.

D. System Comparison

I. General Analysis

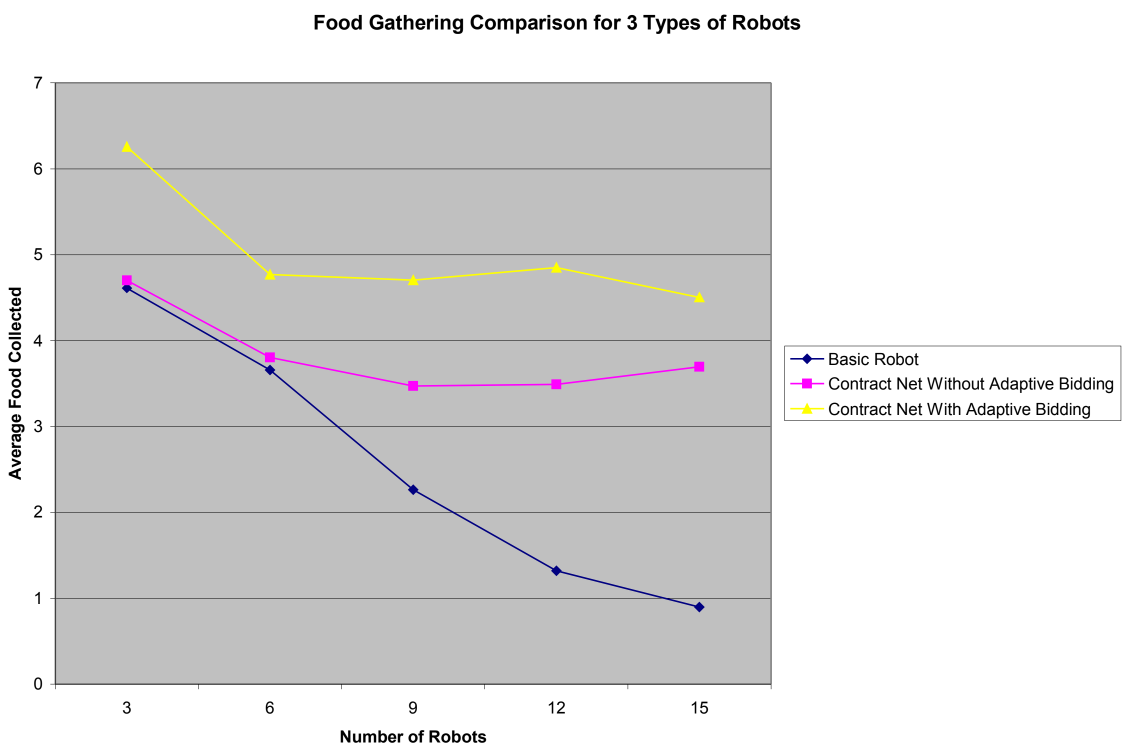

Figure 19: Per Capita Food Gathering Comparison for 3 Types of Robots

The performance of the basic robot suffered heavily as more robots were used. Observations indicated that this was a result of the greedy nature of the basic robot in conjunction with information sharing. This resulted in the basic robots tending to try to gather food from the same food source, which led to heavy congestion with robots spending a significant amount of their time just trying to navigate around each other.

The robot that used the Contract Net Protocol without adaptive bidding started off very similar to the basic robot, but had far less of a performance drop as additional robots were used on the team. Observations suggest that this is due to robots not gathering food from each other's food sources, significantly reducing the amount of time robots spent navigating around each other.

The robot that used the Contract Net Protocol with adaptive bidding (7 nodes in each hidden layer with one output) performed the best. This included the beginning which is likely to just be a random variation, given that the results for when using more team members is fairly stable (albeit lower) and seems to correspond to the starting level.

II. Statistical Analysis

An ANOVA test was used (Figure 20) and showed with 95% confidence that the results from the three robot types were not the same. It also showed with 95% confidence that the results from teams of robots of different sizes were not the same. It lastly also showed with 95% confidence that the combination of robot type and team size made a difference in the results.

5. Conclusion

A. Contributions

This thesis first confirmed the findings of [19] in showing that the use of the Contract Net Protocol is both practical and beneficial in multi-robot systems. The major contribution of this thesis however was demonstrating the importance of bid formulation when using the Contract Net Protocol in multi-robot systems and showing that system performance can be enhanced by using adaptive bidding through use of neural networks. Additionally, this thesis showed the need for a social control mechanism when using improved communication and demonstrated the benefit of using on such mechanism, the Contract Net Protocol. Finally, this thesis showed that when using a neural network for adaptive bidding with the Contract Net Protocol in a time limited setting for foraging, a neural network provides the most benefit when concentrating on energy.

B. Benefits of Contract Net

There are several benefits of using a Contract Net for controlling a group of robots. The first is that some sort of social order must be established for who has permission to do what. Although when there are only a few robots, such order makes little difference, as the number of robots increases it becomes critical to have some such semblance of order, otherwise a series of robots is prone to heavy congestion, drastically reducing the efficacy of the individual robots, and the system as a whole. The order imposed by the Contract Net Protocol may seem like a limitation at first from the perspective of the individual robot. From an individual robot's perspective, it will often limit the robot from gathering from the closest food source, which from the robot's perspective is usually going to be the best choice. The robot will then wind up gathering from a less desirable food source. However because of the structure involved, that limits not only that one robot, but all the other robots, the individual robot nonetheless ultimately benefits, as does the system as a whole.

There are more benefits to using the Contract Net Protocol other than simply the social organization. When using adaptive bidding, the Contract Net Protocol is a full auction system. Although the performance benefit between when using the Contract Net Protocol with adaptive bidding and using the Contract Net Protocol without adaptive bidding was not as large as the performance benefit between using the Contract Net Protocol without adaptive bidding and using the basic robot, there was nonetheless a definite benefit as robots were given tasks for which they were slightly better suited than their fellow robots.

As it stands today, the benefit of using the Contract Net Protocol in robotic teams may be rather small because of the large communications requirements of the protocol. In the future however, when greater bandwidth and more reliable communications between robots is commonplace, the Contract Net Protocol can be a suitable task allocation strategy for autonomous mobile robots.

C. Neural Network Architectures

Although it has been shown here that adaptive bidding does increase the performance of a team of mobile robots that uses the Contract Net Protocol, setting up a neural network that can adaptively form good bids is difficult even in a simulated environment.

There are issues of uncertainty that mean that the network is being trained with inconsistent data. For example, in these experiments a robot suffered damage only some of the time it went through an obstacle. This means that the results indicating which network architecture was best may not be entirely accurate, despite being run multiple times. The best architecture may have to some degree simply lucked into good results, while an architecture that seemed to perform more poorly may have simply had some back luck.

This is the nature of neural networks. While they can be designed so that they generally perform well for a particular problem, the performance can almost never be guaranteed. In particular, the more complex a neural network is, the more potential problems can be encountered by the neural network. With a more complex simulation or an actual robot, the neural network is likely to have to be more complex and finding a good network architecture will take significant time. In addition, the usage of a neural network will preclude being able to guarantee the performance of a team. While a team using the Contract Net Protocol with a neural network used for adaptive bidding may generally perform quite well, there may be cases in which it can perform poorly.

Furthermore, neural networks are currently relatively computationally expensive. At present it would be extremely expensive to create such teams of robots with sufficient computational power to run a fairly complex neural network. This however is a problem that will likely change in the relatively near future, almost certainly before the communication needs for such a team of robots can be satisfied. Current trends are to improve computational power by using multiple processors. This could be ideal for neural networks because the computations for the nodes in a layer of a neural network can be done in parallel, meaning that cheaper multiple processor systems will greatly increase the usability of neural networks in robot systems.

D. Effects of Adaptive Bidding

When these experiments were being planned, it was expected that the inclusion of adaptive bidding would make a significant difference. However, a moderate performance benefit was achieved and it was not nearly as large as initially suspected. The larger issue turned out to be a need for some sort of social control mechanism for the robots. Such a social control mechanism will certainly be needed for teams of robots, but usage of a social control mechanism, namely the Contract Net Protocol, in conjunction with adaptive bidding is certainly going to provide better results.

Computational limitations limited the number of architectures that were tested, and it is quite possible that with further resources, adaptive bidding could provide even better results. The results of using adaptive bidding were sufficient to show that a definite benefit can be obtained by using adaptive bidding.

E. Issues with Better Communication

Chief among the unanticipated results of this experiment were the implications of better communication among robots. The assumption of fairly reliable communication that supported moderately high bandwidth communications entailed information sharing among robots. This information sharing really required a social control mechanism among robots in order for robots to not mob a select few food sources.

A number of real world analogous situations can be applicable here, for example in cases where on the news it is announced that a certain gas station is selling gas for much cheaper than competitors (it has on rare occasion been known to happen). This announcement on the news alerts a large number of people, who examine the perspective from their own perspective and see a personal benefit to going to this gas station in expectation of getting cheaper gas. Yet this situation will likely cause a flood of people to go to this gas station, causing numerous problems including significant traffic congestion.

Ironically, it may be the case that systems that do not use a social control mechanism such as the Contract Net Protocol may benefit from not having better communication. If fewer robots know about a good food source, then there will be less congestion resulting from such numerous robots trying to access this food source, while other robots will congregate around food sources that are in closer proximity, even though these food sources may not be quite as appealing.

Better communication among robots is almost certainly going to become a reality at some point. This in turn will necessitate social control mechanisms, and the Contract Net Protocol has shown itself to be a promising such mechanism, which can be improved by using adaptive bidding. Thus the use and further research of adaptive bidding is a necessity that cannot be ignored.

F. Future Work

There are a number of areas in which this work can be expanded. Unfortunately one such prime area is to test this approach on actual robots, but because of the assumptions made, this is impractical at the present. Nonetheless, a time will come in which these experiments can be run on actual situated agents. Until that time however, there are some adaptations that can be made to the simulation.

One such adaptation that can bring the simulation closer to real robots would be to use multiple computers for the simulation, having each simulated robot be controlled by a separate computer. This would allow for more complex network architectures to be experimented with.

Furthermore it would be extremely interesting to see the effects of using heterogeneous network architectures. In this experiment, teams using the Contract Net Protocol with adaptive bidding consisted only of robots with the same number of nodes. There is no guarantee that this is the best way of having a team use adaptive bidding, and it is possible that having multiple network architectures would alter the bidding process and that alteration may be an improvement.

[1] Smith, RG. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, Volume C-29, No. 12, December 1980

[2] A distributed tasks allocation scheme in multi-UAV context. Lemaire, T.; Alami, R.; Lacroix, S.; Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on Distributed Robotics, (4), 2004. Page 3622

[3] Weiss, Gerhard, Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence. The MIT Press, 1999. Page 212

[4] http://pages.ebay.com/help/buy/buyer-multiple.html, accessed May 1, 2007.

[5] Scerri, P. Coordination of Large-Scale Multiagent Systems. Springer, 2006. Page 78.

[6] http://www-128.ibm.com/developerworks/java/library/j-robocode/, May 1, 2007.

[7] http://www.teambots.org/, accessed May 1, 2007.

[8] http://gameprog.it/hosted/guntactyx/, accessed May 1, 2007

[9] http://robocode.sourceforge.net/, accessed May 1, 2007

[10] Negotiation as a Metaphor for Distributed Problem Solving. Cambridge, Mass. : Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1981.

[11] Gerkey, B. Mataric, Maja. A market-based formulation of sensor-acuator network coordination. Proceedings of the AAAI Spring Symposium on Intelligent Embedded and Distributed Systems, Palo Alto, California, March 25-27, 2002. Page 21.

[12] Multi-Agent Path Planning Based on Task-Swap Negotiation. M Golfarelli, D Maio, S Rizzi - Proceedings 16th UK Planning and Scheduling SIG Workshop, 1997. Page 69.

[13] Neural network dynamics for path planning and obstacle avoidance. R Glasius, A Komoda, S Gielen - Neural Networks, (1), 1995. Page 125

[14] Artificial Intelligence: A Modern Approach. Russell, S., and Norvig, P., 2nd Ed. (2003). Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, page 1.

[15] Weiss, Gerhard, Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence. The MIT Press, 1999. Page 1.

[16] Readings in Distributed Artificial Intelligence. Bond, A.H., and Gasser, L., Morgan Kaufmann Publishers, 1988.

[17] Cooperative mobile robotics: Antecendents and directions. Cao, Fukunaga and Kahng. Autonomous Robots, (4), 1997. Page 7

[18] Multi-robot exploration controlled by a market economy. Zlot, Stentz, Dias, Thayer, Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Page 3016

[19] Gerkey, Brian. Mataric, Maja. Sold!: Auction Methods for Multirobot Coordination. IEEE Transactions on robotics and Automation, Vol. 18, No. 5, October 2002. Page 758

[20] L.E. Parker, "ALLIANCE, An architecture for fault-tolerant multirobot cooperation," IEEE Transactions on Robotics and Automation, vol. 14, p220-240, Apr. 1998

[21] Methods for task allocation via agent coalition formation. Shehory, O. and Kraus, S., (1998). *Artificial Intelligence*, Vol. 101, No. 1-2, 165-200.

[22] Neural Networks: an introduction. Muller, Berndt, Reinhardt Joachim. Springer-Verlag New York, Inc. New York, NY. 1990

[22] Learning internal representations by error propagation. DE Rumelhart, GE Hinton, RJ Williams - 1986 - MIT Press Cambridge, MA, USA

[23] Artificial Intelligence: A Modern Approach. Russell, S., and Norvig, P., 2nd Ed. (2003). Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, pg 738.

[24]"How many hidden units should I use?", http://www.faqs.org/faqs/ai-faq/neural-nets/part3/index.html

[25] McCullagh, P. and Nelder, J.A. (1989) Generalized Linear Models, 2nd ed., London: Chapman & Hall.

[26] Neural Networks: A Review from a Statistical Perspective. Bing Cheng, D. M. Titterington. Statistical Science, Vol. 9, No. 1 (Feb., 1994), pp. 2-30.

[27] Geman, S., Bienenstock, E. and Doursat, R. (1992), "Neural Networks and the Bias/Variance Dilemma", Neural Computation, 4, 1-58.

[28] http://sourceforge.net/projects/robocode, accessed May 1, 2007

[29] http://robocode.sourceforge.net/docs/robocode/, accessed May 1, 2007

[30] http://www.joone.org/, accessed May 1, 2007

[31] Co-learning and the evolution of social activity. Shoham and Tennenholtz. Technical Report STAN-CS-TR-94-1511, Stanford University, 1994.

[32] On the Synthesis of Useful Social Laws for Artificial Agent Societies. Shohamand Tennenholtz. *AAAI-92*, 704-709, 1992.

[33] Artificial Social Systems Part I: Basic Principles. Moses and Tennenholtz. Technical Report CS90-12, Weizmann Institute, 1990.

[34] On Computational Aspects of Artificial Social Systems. Moses and Tennenholtz. Proceedings of Distributed Artificial Intelligence, 1992.

[35] Extending the contract net framework. Sandholm and Lesser. Proceedings of the 1st International Conference on Multi-Agent Systems. 1995. Pages 328.

[36] Advantages of a Leveled Commitment Contracting Protocol. Sandholm and Lesser. Thirteenth National Conference on Artificial Intelligence. 1996. Page 126

[37]Fundamentals of Artificial Neural Networks. Assoun, Mohamad. MIT Press, Cambridge, MA. 1995. Page 211.

7. Appendix

A. ANOVA

ANOVA: Two-Factor With Replication

SUMMARY	3	6	9	12	15	Total
Basic Gatherer						
Count	20	20	20	20	20	100
Sum	92.26	73.19	45.28	26.43	17.96	255.12
Average	4.613	3.6595	2.264	1.3215	0.898	2.5512
Variance	0.631338	0.439542	0.928741	0.392066	0.203427	2.479277
Contract Net						
Count	20	20	20	20	20	100
Sum	94	76.04	69.44	69.77	73.91	383.16
Average	4.7	3.802	3.472	3.4885	3.6955	3.8316
Variance	0.522063	0.328291	0.300406	0.374708	0.3115	0.558721
7/7/1						
Count	20	20	20	20	20	100
Sum	125.12	95.38	69.44	96.99	90.07	477
Average	6.256	4.769	3.472	4.8495	4.5035	4.77
Variance	0.753057	0.943641	0.300406	0.193131	0.317603	1.283392
Total						
Count	60	60	60	60	60	
Sum	311.38	244.61	184.16	193.19	181.94	
Average	5.189667	4.076833	3.069333	3.219833	3.032333	
Variance	1.193396	0.798202	0.822345	2.455449	2.695052	

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	248.1031	2	124.0515	268.126	3.18E-66	3.027443
Columns	206.0184	4	51.50461	111.3225	5.14E-57	2.40332
Interaction	89.94067	8	11.24258	24.29981	2.26E-28	1.970961
Within	131.8585	285	0.462661			
Total	675.9207	299				

Figure 20: ANOVA results.

B. Results from Experiments

Basic Robot

Team Members	Total Food Collected	Total Energy Used	Engine Efficiency	Scanner Efficiency	Collection Efficiency
3	4.613	5105.274	68.60%	63.34%	66.62%
6	3.6595	4301.12	70.53%	80.28%	74.97%
9	2.265	6299.66	68.00%	69.86%	14.88%
12	1.3215	4893.115	56.41%	89.51%	91.44%
15	0.898	4459.62	73.08%	82.07%	18.18%

Contract Net Robot Without Adaptive Bidding

Team Members	Total Food Collected	Total Energy Used	Engine Efficiency	Scanner Efficiency	Collection Efficiency
3	4.7	4088.201	41.78%	75.13%	27.29%
6	3.802	4873.947	42.97%	66.14%	91.19%
9	3.472	4350.094	65.44%	88.53%	63.34%
12	3.4885	4691.518	73.52%	79.69%	35.06%
15	3.6955	4418.942	66.66%	64.53%	66.31%

Contract Net Robot With Adaptive Bidding

Hidden 1	Hidden 2	Output Nodes	Team Members	Total Food Collected	Total Energy Used	Engine Efficiency	Scanner Efficiency	Collection Efficiency
2	0	5	3	1.422684	5918.424	72.08%	68.75%	20.69%
3	0	5	3	1.678598	4720.122	67.66%	86.39%	36.98%
4	0	5	3	1.689634	5395.031	41.17%	80.40%	91.83%
5	0	5	3	1.625276	6709.63	57.93%	67.43%	33.51%
6	0	5	3	2.310009	5282.659	47.77%	70.30%	49.51%
7	0	5	3	2.25526	5127.758	54.75%	67.10%	46.56%
8	0	5	3	2.215182	5518.633	72.87%	84.22%	84.16%
9	0	5	3	1.615346	5469.922	78.24%	73.18%	36.21%
10	0	5	3	1.736228	6894.614	52.72%	70.04%	34.51%
11	0	5	3	1.601201	5269.264	46.14%	72.13%	58.41%
12	0	5	3	1.445183	6106.836	64.99%	81.48%	82.42%
13	0	5	3	1.463722	6018.31	43.88%	86.55%	70.22%
14	0	5	3	1.437936	5102.83	70.08%	83.69%	21.64%
15	0	5	3	1.476694	4555.23	49.90%	65.34%	13.20%

16	0	5	3	1.73481	6759.062	54.75%	66.11%	62.82%
2	0	1	3	1.445558	5345.426	67.05%	69.08%	36.47%
3	0	1	3	1.680133	4183.273	41.66%	79.97%	81.78%
4	0	1	3	1.912449	5231.29	42.14%	83.23%	58.82%
5	0	1	3	1.818005	4517.413	76.98%	67.09%	78.96%
6	0	1	3	2.369024	4859.484	57.40%	64.51%	57.55%
7	0	1	3	2.554391	5478.884	55.12%	87.03%	54.23%
8	0	1	3	1.967324	4050.412	67.17%	86.41%	60.81%
9	0	1	3	2.142679	4073.311	67.27%	63.61%	36.77%
10	0	1	3	1.937468	6694.193	65.03%	82.91%	56.57%
11	0	1	3	1.782682	4223.03	79.70%	73.34%	22.16%
12	0	1	3	1.530516	4382.913	43.83%	63.58%	52.48%
13	0	1	3	1.531306	5210.955	74.92%	72.44%	51.11%
14	0	1	3	1.496865	5593.001	61.69%	74.40%	87.82%
15	0	1	3	1.588271	6033.557	54.61%	69.21%	25.58%
16	0	1	3	1.90505	5711.934	59.63%	80.25%	12.08%
2	2	5	3	1.532877	4513.817	44.40%	83.99%	21.37%
2	3	5	3	1.617211	4675.941	59.20%	83.17%	69.95%
2	4	5	3	1.765867	5154.751	69.19%	65.22%	22.38%
2	5	5	3	1.681562	5043.938	51.49%	88.72%	13.85%
2	6	5	3	2.123494	4364.812	51.06%	67.46%	44.12%
2	7	5	3	2.328124	5270.344	49.45%	84.72%	14.80%
2	8	5	3	1.865478	4596.36	45.27%	76.87%	67.26%
2	9	5	3	1.85879	5147.5	49.91%	76.00%	49.43%
2	10	5	3	1.806631	5408.624	47.92%	61.17%	15.48%
3	2	5	3	1.736841	6841.851	58.65%	71.11%	19.72%
3	3	5	3	1.620929	6738.159	65.09%	61.56%	89.47%
3	4	5	3	1.953693	5137.482	55.79%	75.02%	11.18%
3	5	5	3	2.347523	4312.118	40.42%	88.65%	13.85%
3	6	5	3	3.117738	5432.584	74.89%	76.05%	71.07%
3	7	5	3	2.856911	4580.64	46.67%	73.12%	74.62%
3	8	5	3	2.110215	5322.426	53.19%	79.94%	72.02%
3	9	5	3	2.007643	5772.173	43.96%	89.97%	51.41%
3	10	5	3	1.98689	5993.654	50.78%	70.17%	85.21%
4	2	5	3	1.666635	4709.067	45.67%	85.54%	76.18%
4	3	5	3	1.863577	4806.632	46.03%	89.56%	25.67%
4	4	5	3	2.812202	6644.579	70.24%	76.95%	29.85%
4	5	5	3	3.320426	5504.385	53.44%	87.85%	53.40%
4	6	5	3	3.694918	4032.118	48.78%	81.06%	73.05%
4	7	5	3	4.419131	4655.591	41.82%	76.64%	92.62%
4	8	5	3	3.067812	5910.527	63.71%	69.02%	84.58%
4	9	5	3	2.601031	4924.997	43.49%	87.60%	27.28%
4	10	5	3	2.83049	6742.561	75.80%	79.05%	35.36%
5	2	5	3	1.996069	5567.885	63.62%	81.17%	37.32%
5	3	5	3	1.959799	6376.593	76.99%	78.00%	32.25%
5	4	5	3	3.659189	4661.783	66.41%	74.18%	40.77%
5	5	5	3	2.997616	6879.473	58.13%	85.96%	15.96%
5	6	5	3	3.053278	4021.061	71.32%	81.18%	11.16%
5	7	5	3	5.953181	4814.498	42.69%	62.61%	45.76%
5	8	5	3	3.859309	4530.872	65.94%	76.43%	24.12%

5	9	5	3	3.592787	5547.54	70.24%	84.26%	29.28%
5	10	5	3	2.284716	5782.96	68.40%	80.76%	44.15%
6	2	5	3	2.188332	4130.969	40.73%	85.09%	19.12%
6	3	5	3	3.235947	6238.624	78.90%	66.74%	74.85%
6	4	5	3	2.370279	4082.15	62.07%	64.66%	36.89%
6	5	5	3	2.945238	5179.074	49.38%	81.44%	36.16%
6	6	5	3	6.281329	4915.814	46.72%	88.82%	46.65%
6	7	5	3	3.633752	4674.806	77.89%	73.87%	62.75%
6	8	5	3	4.19	4359.424	49.53%	63.03%	42.58%
6	9	5	3	3.60934	6352.109	70.71%	82.51%	40.41%
6	10	5	3	3.690521	4369.944	65.12%	78.98%	34.61%
7	2	5	3	1.957466	6037.764	72.77%	67.90%	68.59%
7	3	5	3	2.443282	4796.252	54.44%	80.09%	38.01%
7	4	5	3	4.360019	6423.444	74.86%	66.10%	12.77%
7	5	5	3	4.039579	5232.504	47.89%	83.72%	12.74%
7	6	5	3	3.769747	6580.453	65.22%	83.31%	92.51%
7	7	5	3	4.503532	5553.953	79.40%	77.89%	10.31%
7	8	5	3	3.503225	5336.772	65.01%	76.52%	26.40%
7	9	5	3	5.618852	5897.497	42.77%	64.76%	36.88%
7	10	5	3	2.643647	6733.897	58.31%	77.94%	39.76%
8	2	5	3	2.194655	5207.199	73.49%	74.24%	35.09%
8	3	5	3	3.18174	4566.04	63.21%	88.98%	37.04%
8	4	5	3	3.056356	6718.715	65.63%	76.64%	65.65%
8	5	5	3	4.438313	6084.906	51.73%	65.48%	53.89%
8	6	5	3	3.793847	5133.3	67.23%	69.03%	14.10%
8	7	5	3	5.327585	6744.76	71.31%	66.23%	37.94%
8	8	5	3	5.808619	6223.026	40.67%	83.81%	92.41%
8	9	5	3	3.177059	5870.208	64.63%	87.27%	32.33%
8	10	5	3	3.385905	6882.408	59.35%	71.40%	23.40%
9	2	5	3	1.822001	5417.498	51.55%	80.02%	25.11%
9	3	5	3	2.206994	5684.024	59.79%	72.73%	23.04%
9	4	5	3	2.603156	5354.572	49.05%	69.19%	65.59%
9	5	5	3	2.633245	4262.022	59.61%	61.40%	66.61%
9	6	5	3	4.359454	4616.594	58.85%	78.49%	63.87%
9	7	5	3	4.544754	4346.056	52.83%	87.62%	28.14%
9	8	5	3	3.491971	6801.114	75.78%	74.18%	27.91%
9	9	5	3	3.902803	4849.413	48.94%	70.05%	91.92%
9	10	5	3	2.978183	4499.214	50.75%	75.05%	46.57%
10	2	5	3	1.558817	6715.542	68.80%	76.31%	79.02%
10	3	5	3	2.495308	4041.569	78.80%	88.97%	93.44%
10	4	5	3	3.121414	4438.652	51.37%	80.00%	63.56%
10	5	5	3	2.565176	6512.408	69.02%	74.36%	47.75%
10	6	5	3	3.522549	5726.464	41.12%	71.53%	55.07%
10	7	5	3	3.602468	4352.717	47.37%	89.11%	10.47%
10	8	5	3	3.950324	5762.794	60.81%	89.35%	26.86%
10	9	5	3	3.528199	5738.767	59.56%	65.76%	37.40%
10	10	5	3	2.787576	5870.231	54.05%	77.18%	35.95%
2	2	1	3	1.532811	4245.014	76.53%	74.08%	33.95%
2	3	1	3	1.62283	4426.51	53.72%	65.42%	32.75%
2	4	1	3	1.879006	5385.356	65.25%	69.43%	53.63%

2	5	1	3	2.285591	6561.907	71.99%	74.77%	32.92%
2	6	1	3	2.481559	5710.689	64.61%	69.69%	12.88%
2	7	1	3	2.129984	5916.139	71.43%	75.13%	38.18%
2	8	1	3	2.121613	4614.298	49.00%	83.87%	74.47%
2	9	1	3	2.067612	4771.93	47.86%	72.70%	76.27%
2	10	1	3	1.908923	6603.92	44.20%	68.29%	93.20%
3	2	1	3	1.611836	4428.335	54.29%	63.07%	78.89%
3	3	1	3	1.751271	4054.048	41.04%	79.98%	83.12%
3	4	1	3	2.391166	5629.294	60.76%	82.97%	52.67%
3	5	1	3	2.248687	4045.683	51.81%	61.67%	60.63%
3	6	1	3	3.155414	5563.264	42.53%	65.14%	11.03%
3	7	1	3	2.839712	5559.198	78.89%	60.42%	26.13%
3	8	1	3	2.860578	4239.473	44.38%	81.80%	66.00%
3	9	1	3	3.011341	4659.126	67.36%	85.05%	72.52%
3	10	1	3	2.504924	5439.401	51.18%	82.55%	79.52%
4	2	1	3	1.678975	6041.995	43.78%	82.39%	41.08%
4	3	1	3	2.114848	4712.127	45.96%	79.24%	34.70%
4	4	1	3	3.359466	5250.706	52.71%	81.79%	78.75%
4	5	1	3	3.357845	5031.034	55.75%	60.21%	13.13%
4	6	1	3	3.572333	5546.261	41.72%	86.82%	91.72%
4	7	1	3	5.511224	4385.808	62.61%	81.29%	56.31%
4	8	1	3	4.062096	4566.842	65.43%	77.18%	58.61%
4	9	1	3	3.157598	5055.519	69.73%	68.47%	83.05%
4	10	1	3	2.807292	6960.007	69.26%	68.80%	88.35%
5	2	1	3	1.85695	4120.434	48.45%	71.89%	58.78%
5	3	1	3	2.993311	4083.506	69.41%	62.84%	70.25%
5	4	1	3	2.68937	5416.351	43.01%	79.60%	78.41%
5	5	1	3	4.845338	5024.932	50.04%	83.18%	36.68%
5	6	1	3	6.126368	5608.991	42.69%	69.93%	86.28%
5	7	1	3	5.912364	4411.426	54.84%	80.11%	33.77%
5	8	1	3	3.473765	5607.75	48.71%	73.38%	36.91%
5	9	1	3	2.659304	4521.477	69.93%	60.90%	81.93%
5	10	1	3	3.451381	6104.733	57.96%	85.63%	40.55%
6	2	1	3	2.464318	4920.704	45.68%	82.25%	61.08%
6	3	1	3	2.805008	6331.511	44.22%	64.09%	86.20%
6	4	1	3	3.955509	4604.665	55.55%	65.27%	20.25%
6	5	1	3	5.875458	6678.889	68.74%	72.09%	80.73%
6	6	1	3	3.853449	6899.536	52.26%	79.06%	37.05%
6	7	1	3	5.396675	5685.968	74.76%	68.45%	49.15%
6	8	1	3	3.829749	6205.064	61.07%	61.98%	64.38%
6	9	1	3	5.722941	4151.146	63.24%	87.84%	61.37%
6	10	1	3	3.796697	6236.777	67.56%	63.97%	17.59%
7	2	1	3	2.594488	6215.948	49.53%	72.66%	14.58%
7	3	1	3	2.499595	4140.307	61.32%	89.49%	18.35%
7	4	1	3	5.162797	4946.052	77.01%	87.39%	21.97%
7	5	1	3	5.215101	4373.2	50.48%	70.80%	93.09%
7	6	1	3	5.631158	4143.843	48.37%	79.65%	49.40%
7	7	1	3	6.256824	4118.317	73.20%	82.23%	50.31%
7	8	1	3	6.156156	5465.945	59.96%	70.89%	19.18%
7	9	1	3	6.020715	5878.842	71.46%	68.16%	19.77%

7	10	1	3	5.264788	4291.783	43.95%	64.73%	58.67%
8	2	1	3	2.221594	5590.511	66.12%	61.40%	11.61%
8	3	1	3	2.254531	6060.213	75.50%	62.78%	53.35%
8	4	1	3	2.710164	5761.069	73.92%	72.95%	42.19%
8	5	1	3	4.449074	4902.805	51.49%	82.62%	64.26%
8	6	1	3	4.310199	5534.951	58.65%	70.65%	56.28%
8	7	1	3	3.796336	6408.49	59.27%	83.47%	15.10%
8	8	1	3	4.810498	6440.318	46.59%	86.16%	13.13%
8	9	1	3	5.037883	5625.115	62.72%	80.98%	91.95%
8	10	1	3	2.883529	4363.143	60.95%	82.03%	34.32%
9	2	1	3	1.936264	5901.888	46.94%	76.56%	77.64%
9	3	1	3	3.165957	6270.164	43.76%	77.88%	10.37%
9	4	1	3	2.820307	4586.55	67.59%	77.04%	42.69%
9	5	1	3	4.69621	6571.675	62.02%	89.02%	41.43%
9	6	1	3	4.155229	6083.671	45.18%	66.67%	40.60%
9	7	1	3	4.436129	4523.82	71.98%	74.42%	58.44%
9	8	1	3	4.903091	5893.243	58.34%	67.05%	25.03%
9	9	1	3	4.747729	4918.632	56.90%	87.57%	70.87%
9	10	1	3	4.034758	4408.949	50.58%	72.68%	22.71%
10	2	1	3	1.800443	6352.642	57.07%	69.64%	60.85%
10	3	1	3	1.861981	5344.671	74.34%	80.36%	76.46%
10	4	1	3	2.19123	5426.451	57.84%	70.48%	55.17%
10	5	1	3	4.186431	4624.447	46.57%	73.90%	63.61%
10	6	1	3	4.25975	4935.454	70.91%	87.19%	26.75%
10	7	1	3	5.505877	6132.627	63.62%	89.12%	81.58%
10	8	1	3	4.510597	4518.579	56.25%	85.77%	13.51%
10	9	1	3	2.890052	6399.691	75.19%	78.32%	38.73%
10	10	1	3	2.462912	6051.065	79.03%	69.03%	47.95%
2	0	5	6	1.502901	4517.825	78.32%	74.03%	89.08%
3	0	5	6	1.480263	4322.314	63.38%	80.12%	42.07%
4	0	5	6	1.703896	5030.176	59.95%	81.23%	90.94%
5	0	5	6	1.971315	6216.279	67.59%	71.57%	85.05%
6	0	5	6	1.769349	4067.894	63.93%	77.65%	31.30%
7	0	5	6	1.826108	6440.799	66.47%	87.99%	32.76%
8	0	5	6	1.862122	4811.902	70.75%	69.93%	49.98%
9	0	5	6	1.622607	6645.118	74.93%	73.64%	46.70%
10	0	5	6	1.645498	5155.075	60.49%	83.30%	70.15%
11	0	5	6	1.57689	5119.229	62.50%	77.40%	42.28%
12	0	5	6	1.462529	5751.989	63.54%	73.87%	94.33%
13	0	5	6	1.489732	4242.698	61.16%	65.66%	45.32%
14	0	5	6	1.44124	4472.238	72.72%	64.95%	55.14%
15	0	5	6	1.544278	4148.936	51.39%	82.70%	18.43%
16	0	5	6	1.732907	6436.568	48.47%	77.25%	86.06%
2	0	1	6	1.484287	4524.569	64.08%	78.34%	64.84%
3	0	1	6	1.700865	6127.923	76.19%	83.41%	89.09%
4	0	1	6	1.895384	6858.828	63.57%	84.15%	81.68%
5	0	1	6	1.878566	4233.972	79.12%	61.27%	10.15%
6	0	1	6	2.234422	4545.763	44.45%	65.35%	74.94%
7	0	1	6	1.772109	6223.723	43.79%	73.87%	12.99%
8	0	1	6	2.115247	5562.413	72.42%	79.93%	28.58%

9	0	1	6	1.786368	5352.386	48.34%	80.48%	53.69%
10	0	1	6	1.678041	6340.002	67.30%	86.96%	50.55%
11	0	1	6	1.627259	5613.452	51.21%	76.69%	38.81%
12	0	1	6	1.511742	4649.445	44.94%	80.71%	29.38%
13	0	1	6	1.504229	5469.521	78.54%	63.33%	12.64%
14	0	1	6	1.411449	5565.741	79.66%	76.31%	28.60%
15	0	1	6	1.520736	4224.617	53.89%	62.21%	85.64%
16	0	1	6	1.793216	5470.591	78.77%	60.78%	23.89%
2	2	5	6	1.481944	5627.076	51.34%	81.58%	16.44%
2	3	5	6	1.665202	5211.024	76.87%	72.07%	58.08%
2	4	5	6	1.662811	6690.123	77.22%	72.30%	71.37%
2	5	5	6	1.606208	5941.775	69.15%	66.02%	23.39%
2	6	5	6	1.764959	4371.001	43.50%	63.97%	79.83%
2	7	5	6	2.018467	4602.643	40.41%	84.33%	11.83%
2	8	5	6	1.885036	6649.572	46.46%	67.04%	75.77%
2	9	5	6	1.967684	5486.961	43.49%	80.29%	50.27%
2	10	5	6	1.645052	6487.552	76.39%	70.59%	71.72%
3	2	5	6	1.643394	5563.791	48.92%	86.54%	57.36%
3	3	5	6	1.564352	4702.991	78.12%	65.53%	92.70%
3	4	5	6	2.031496	5954.131	44.36%	69.17%	20.67%
3	5	5	6	2.224507	5755.754	43.77%	79.18%	56.49%
3	6	5	6	2.738968	5132.141	58.82%	71.51%	59.42%
3	7	5	6	2.960625	6424.88	40.70%	71.82%	94.57%
3	8	5	6	2.91342	4991.07	63.28%	76.45%	74.38%
3	9	5	6	1.958798	5585.984	61.79%	77.48%	79.79%
3	10	5	6	1.670919	6725.018	43.84%	61.63%	67.95%
4	2	5	6	1.530112	5241.269	76.44%	62.10%	73.48%
4	3	5	6	2.142941	5178.637	40.81%	87.01%	17.75%
4	4	5	6	1.949817	5882.202	57.33%	70.68%	83.55%
4	5	5	6	2.798493	5674.244	66.91%	69.14%	26.85%
4	6	5	6	2.898057	5664.18	62.31%	64.88%	43.37%
4	7	5	6	2.562862	4163.402	53.77%	60.21%	39.08%
4	8	5	6	3.589771	5167.329	65.92%	85.33%	94.52%
4	9	5	6	2.78716	6577.152	43.26%	78.02%	94.45%
4	10	5	6	1.957345	4252.522	45.17%	84.91%	26.21%
5	2	5	6	1.738953	6776.628	44.61%	83.54%	12.59%
5	3	5	6	2.485962	6649.747	55.09%	74.09%	37.96%
5	4	5	6	3.250969	4511.553	49.08%	73.26%	68.77%
5	5	5	6	2.769328	6005.77	64.93%	67.68%	46.19%
5	6	5	6	4.367646	4514.762	79.45%	76.89%	20.23%
5	7	5	6	2.758536	4802.532	54.84%	85.53%	38.50%
5	8	5	6	3.720732	4439.6	73.58%	71.94%	28.83%
5	9	5	6	3.921818	4091.024	47.88%	72.99%	17.81%
5	10	5	6	2.183604	6567.062	76.20%	75.36%	23.10%
6	2	5	6	1.709293	5981.456	68.89%	61.32%	33.74%
6	3	5	6	2.114298	4060.244	71.52%	86.60%	27.16%
6	4	5	6	2.671319	5294.941	60.11%	66.48%	39.36%
6	5	5	6	4.008982	6066.745	50.78%	87.32%	37.15%
6	6	5	6	5.236033	4151.412	79.16%	72.94%	36.26%
6	7	5	6	5.778789	4663.377	54.10%	76.76%	73.91%

6	8	5	6	5.008364	5555.839	74.78%	87.52%	58.63%
6	9	5	6	4.299171	5147.981	56.24%	61.87%	15.39%
6	10	5	6	3.703396	4259.178	60.74%	83.37%	42.46%
7	2	5	6	2.27981	5878.986	44.90%	85.92%	23.95%
7	3	5	6	2.931337	5300.657	55.53%	78.79%	61.21%
7	4	5	6	3.074107	5513.17	50.54%	89.69%	70.20%
7	5	5	6	4.66374	6620.1	53.45%	88.94%	10.19%
7	6	5	6	3.535939	6917.185	42.71%	78.68%	48.23%
7	7	5	6	4.513367	6341.948	46.13%	72.15%	78.22%
7	8	5	6	5.582876	5536.759	49.81%	62.25%	69.75%
7	9	5	6	4.939108	5808.531	58.30%	70.05%	60.14%
7	10	5	6	3.645587	4700.2	72.48%	76.96%	41.69%
8	2	5	6	1.679569	4944.026	76.17%	69.17%	56.02%
8	3	5	6	2.414137	4743.45	74.92%	78.66%	74.48%
8	4	5	6	3.161	6033.445	40.50%	81.49%	69.44%
8	5	5	6	4.448843	5295.843	43.93%	65.15%	63.53%
8	6	5	6	4.878418	5502.041	67.88%	88.89%	51.45%
8	7	5	6	3.722053	5420.565	51.47%	70.39%	91.28%
8	8	5	6	4.489271	4274.939	77.20%	85.53%	21.56%
8	9	5	6	3.104615	5316.604	77.64%	63.00%	45.77%
8	10	5	6	3.195211	6156.285	79.74%	79.74%	26.08%
9	2	5	6	1.650393	5806.043	45.64%	85.57%	63.08%
9	3	5	6	2.531862	4476.301	55.19%	78.15%	30.74%
9	4	5	6	2.762461	4032.66	49.66%	61.00%	71.21%
9	5	5	6	3.936487	5433.567	64.08%	78.00%	76.96%
9	6	5	6	3.449821	4966.99	41.28%	79.84%	31.97%
9	7	5	6	3.63324	5173.02	44.89%	89.93%	64.67%
9	8	5	6	3.066247	6929.448	53.22%	66.63%	44.52%
9	9	5	6	3.426119	4037.954	48.26%	65.16%	76.13%
9	10	5	6	3.152727	4765.455	61.16%	68.64%	56.61%
10	2	5	6	1.670496	6787.858	64.13%	65.66%	31.74%
10	3	5	6	2.276354	6019.232	55.04%	63.94%	43.68%
10	4	5	6	2.036209	5682.275	40.78%	78.71%	66.86%
10	5	5	6	2.552011	5798.213	70.43%	63.53%	77.75%
10	6	5	6	2.89789	5295.605	67.57%	89.96%	38.69%
10	7	5	6	3.645086	6760.794	50.52%	60.40%	64.17%
10	8	5	6	2.73396	4106.52	62.07%	69.22%	68.33%
10	9	5	6	2.289473	4092.11	45.18%	68.77%	68.79%
10	10	5	6	2.709785	4343.81	51.29%	85.56%	77.15%
2	2	1	6	1.498742	5958.741	66.34%	73.59%	76.44%
2	3	1	6	1.656087	4356.894	64.88%	68.13%	72.62%
2	4	1	6	1.807922	6365.119	66.85%	72.61%	93.82%
2	5	1	6	1.807098	6326.375	45.53%	63.19%	60.09%
2	6	1	6	1.844193	4666.972	46.21%	75.56%	46.82%
2	7	1	6	1.769649	5438.584	48.60%	88.10%	17.63%
2	8	1	6	2.317631	4400.602	77.47%	66.38%	93.24%
2	9	1	6	1.60444	6932.068	46.43%	70.19%	14.35%
2	10	1	6	1.72787	5011.448	46.75%	85.05%	39.13%
3	2	1	6	1.575675	5726.392	53.27%	69.82%	20.71%
3	3	1	6	1.992198	4518.139	58.25%	62.03%	66.87%

3	4	1	6	2.148341	5390.312	45.70%	88.84%	48.74%
3	5	1	6	1.952561	6517.027	72.88%	80.37%	66.24%
3	6	1	6	2.366209	5314.402	59.59%	75.96%	28.10%
3	7	1	6	3.422991	6429.982	57.65%	85.79%	30.04%
3	8	1	6	2.157203	5657.645	45.97%	76.70%	88.72%
3	9	1	6	2.440292	6577.286	55.71%	60.72%	10.31%
3	10	1	6	2.036799	6690.604	75.93%	72.66%	90.52%
4	2	1	6	1.848747	4602.518	63.15%	79.18%	74.40%
4	3	1	6	1.936501	5780.072	75.21%	63.58%	27.98%
4	4	1	6	2.422721	5323.555	55.17%	71.09%	16.09%
4	5	1	6	3.356511	4776.436	55.05%	66.66%	25.27%
4	6	1	6	4.178372	4993.724	63.48%	60.83%	16.97%
4	7	1	6	4.623352	5453.376	77.18%	77.45%	52.39%
4	8	1	6	4.125398	6033.737	55.14%	66.10%	80.55%
4	9	1	6	2.960556	5668.348	75.17%	83.36%	90.15%
4	10	1	6	2.638239	6262.695	57.60%	69.47%	31.40%
5	2	1	6	1.918406	5709.708	55.66%	80.05%	78.23%
5	3	1	6	2.858526	6507.726	46.93%	73.40%	15.07%
5	4	1	6	2.557928	5706.598	78.39%	89.80%	57.15%
5	5	1	6	2.652277	6455.714	45.61%	60.25%	45.09%
5	6	1	6	3.851349	5145.151	51.06%	67.25%	29.99%
5	7	1	6	5.508789	5775.314	57.59%	63.17%	24.94%
5	8	1	6	3.684614	4770.108	50.62%	78.90%	78.55%
5	9	1	6	3.129154	4965.346	71.91%	65.45%	22.37%
5	10	1	6	3.667746	5766.866	58.51%	66.33%	72.61%
6	2	1	6	2.1776	6263.875	64.85%	74.25%	25.89%
6	3	1	6	3.035035	6071.416	63.58%	83.75%	22.81%
6	4	1	6	3.291578	6991.891	64.71%	83.71%	92.79%
6	5	1	6	4.005233	6477.27	68.55%	66.23%	59.13%
6	6	1	6	3.110779	4432.144	42.41%	87.60%	16.82%
6	7	1	6	4.805558	6819.225	67.24%	63.14%	45.80%
6	8	1	6	5.908424	4237.751	70.14%	62.13%	92.96%
6	9	1	6	4.312219	5471.974	48.50%	73.85%	23.67%
6	10	1	6	3.90944	6341.226	65.92%	79.47%	64.00%
7	2	1	6	2.002056	4605.258	41.90%	61.74%	48.85%
7	3	1	6	2.696288	5165.851	64.97%	77.63%	69.13%
7	4	1	6	3.842941	5599.208	48.79%	63.78%	48.99%
7	5	1	6	5.878146	5612.16	55.03%	74.33%	85.85%
7	6	1	6	4.256724	4746.666	50.05%	63.21%	16.86%
7	7	1	6	4.769	6071.226	74.64%	78.58%	33.33%
7	8	1	6	4.938204	4221.843	58.96%	67.92%	61.67%
7	9	1	6	4.756881	6290.913	65.65%	66.22%	58.55%
7	10	1	6	4.0614	5181.815	75.24%	83.94%	48.08%
8	2	1	6	2.273352	5373.272	60.65%	82.06%	83.45%
8	3	1	6	2.95901	5764.62	61.30%	66.01%	66.24%
8	4	1	6	2.518348	5803.195	73.68%	84.86%	31.55%
8	5	1	6	4.698547	6670.115	65.23%	83.75%	88.91%
8	6	1	6	3.370118	4306.961	58.42%	76.38%	94.75%
8	7	1	6	4.879721	5614.797	65.06%	63.62%	72.50%
8	8	1	6	5.326444	5108.279	71.30%	68.99%	83.39%

8	9	1	6	3.89778	6983.318	59.13%	63.93%	37.82%
8	10	1	6	3.454647	4792.563	78.56%	89.05%	64.53%
9	2	1	6	1.655278	6528.551	48.52%	83.55%	75.28%
9	3	1	6	2.391098	6214.28	51.42%	84.56%	40.22%
9	4	1	6	2.668013	6861.508	54.28%	68.70%	58.88%
9	5	1	6	2.503588	6829.477	50.37%	88.12%	63.62%
9	6	1	6	3.35136	6833.749	75.56%	82.08%	53.89%
9	7	1	6	6.041497	5072.046	47.90%	85.12%	16.02%
9	8	1	6	4.899552	4412.204	48.30%	77.24%	25.63%
9	9	1	6	3.93171	6634.197	58.04%	69.27%	16.82%
9	10	1	6	3.00713	6398.21	63.73%	78.82%	89.13%
10	2	1	6	1.647118	5176.472	54.69%	70.11%	38.34%
10	3	1	6	1.884813	6581.936	72.25%	62.61%	14.88%
10	4	1	6	2.113098	4677.852	65.57%	69.18%	68.48%
10	5	1	6	2.819639	5367.703	74.11%	80.34%	32.00%
10	6	1	6	3.663282	6539.227	74.56%	64.81%	26.74%
10	7	1	6	2.589583	4626.111	42.18%	74.88%	69.92%
10	8	1	6	3.516732	5737.995	73.36%	83.17%	21.70%
10	9	1	6	3.510195	5698.04	50.03%	73.66%	81.10%
10	10	1	6	2.001689	5896.556	49.91%	84.00%	86.68%
2	0	5	9	1.470398	5967.18	65.03%	65.74%	92.98%
3	0	5	9	1.458381	6334.541	52.16%	75.70%	25.71%
4	0	5	9	1.579281	6323.266	49.86%	63.41%	12.75%
5	0	5	9	1.65666	6939.788	42.34%	62.44%	31.83%
6	0	5	9	1.845856	6506.601	46.31%	61.59%	10.16%
7	0	5	9	1.892163	5560.834	46.78%	63.16%	36.87%
8	0	5	9	1.718398	4959.056	76.28%	87.90%	49.61%
9	0	5	9	1.752482	6137.904	59.37%	74.48%	24.92%
10	0	5	9	1.600783	4508.639	72.63%	87.80%	38.73%
11	0	5	9	1.57492	6301.059	72.61%	71.21%	13.22%
12	0	5	9	1.458094	5589.104	58.90%	88.46%	11.00%
13	0	5	9	1.439573	4326.624	53.00%	81.76%	93.41%
14	0	5	9	1.457544	6027.43	66.17%	60.91%	10.89%
15	0	5	9	1.546625	5433.381	54.91%	61.84%	29.92%
16	0	5	9	1.743359	6981.373	46.43%	68.04%	49.29%
2	0	1	9	1.446055	6674.621	49.47%	65.73%	26.73%
3	0	1	9	1.542618	5266.995	74.35%	61.61%	11.92%
4	0	1	9	1.555943	6214.658	50.34%	88.99%	43.26%
5	0	1	9	1.936393	6090.879	52.40%	74.54%	21.61%
6	0	1	9	1.786867	5715.277	48.32%	68.60%	36.00%
7	0	1	9	1.713814	4378.92	79.35%	68.07%	81.60%
8	0	1	9	1.955559	4246.458	56.41%	83.19%	27.95%
9	0	1	9	1.959795	5139.966	74.72%	75.71%	79.79%
10	0	1	9	1.820412	4792.566	76.81%	85.28%	18.56%
11	0	1	9	1.501728	5746.297	62.13%	70.29%	89.76%
12	0	1	9	1.402993	5786.922	43.80%	78.76%	23.96%
13	0	1	9	1.34	6044.295	40.65%	87.97%	60.72%
14	0	1	9	1.498826	5467.604	61.69%	73.20%	14.59%
15	0	1	9	1.616225	6504.087	41.69%	84.49%	29.11%
16	0	1	9	1.679043	5986.614	43.32%	85.49%	49.89%

2	2	5	9	1.388609	4453.329	48.49%	84.50%	76.26%
2	3	5	9	1.532164	6175.98	64.92%	65.94%	53.79%
2	4	5	9	1.646674	5161.611	59.05%	69.84%	76.42%
2	5	5	9	1.881804	6414.592	62.30%	78.78%	40.00%
2	6	5	9	1.924572	5884.195	64.54%	71.75%	16.80%
2	7	5	9	1.697249	5814.249	70.47%	78.56%	62.13%
2	8	5	9	1.942533	5617.544	47.40%	80.49%	67.81%
2	9	5	9	1.532501	6479.691	70.27%	72.36%	91.21%
2	10	5	9	1.59616	4875.837	48.19%	67.41%	81.34%
3	2	5	9	1.562464	4436.814	65.26%	63.35%	63.26%
3	3	5	9	1.841919	6655.252	61.22%	62.62%	11.85%
3	4	5	9	1.715058	5503.942	43.55%	83.99%	25.39%
3	5	5	9	1.841928	6127.968	53.10%	60.24%	80.09%
3	6	5	9	2.173325	5042.482	78.47%	74.43%	54.12%
3	7	5	9	2.141871	6612.804	42.59%	64.46%	91.70%
3	8	5	9	2.308271	6098.778	54.43%	89.56%	45.82%
3	9	5	9	1.757288	5053.466	41.26%	71.32%	54.35%
3	10	5	9	2.142197	4110.018	64.87%	73.85%	72.41%
4	2	5	9	1.663189	6715.799	43.25%	60.98%	64.38%
4	3	5	9	1.923499	5942.77	60.84%	74.35%	34.10%
4	4	5	9	1.908494	5299.816	63.66%	65.13%	21.09%
4	5	5	9	2.759087	4748.913	63.12%	69.90%	13.51%
4	6	5	9	3.153813	6024.227	56.06%	62.35%	30.70%
4	7	5	9	2.78952	4980.458	44.38%	67.34%	85.43%
4	8	5	9	2.720138	4688.9	61.42%	88.30%	55.27%
4	9	5	9	2.194889	6279.317	45.22%	64.69%	45.62%
4	10	5	9	1.889191	4333.634	75.28%	81.98%	90.53%
5	2	5	9	1.54504	4051.81	52.79%	70.88%	27.10%
5	3	5	9	2.219495	5616.095	49.31%	67.50%	51.39%
5	4	5	9	2.567645	5521.584	70.40%	60.96%	59.29%
5	5	5	9	2.552746	5325.153	73.51%	69.82%	51.90%
5	6	5	9	2.74955	4312.836	51.60%	68.26%	29.85%
5	7	5	9	4.534263	6646.802	52.88%	68.74%	42.12%
5	8	5	9	3.87982	6430.407	62.82%	64.30%	71.83%
5	9	5	9	2.392048	4203.7	44.22%	81.43%	10.72%
5	10	5	9	2.374946	6616.543	57.35%	65.36%	13.41%
6	2	5	9	1.958114	4750.006	49.29%	75.53%	64.73%
6	3	5	9	2.00949	5551.089	50.67%	82.88%	21.81%
6	4	5	9	2.997588	6216.758	60.06%	74.27%	26.56%
6	5	5	9	3.276543	5227.833	78.93%	77.43%	39.03%
6	6	5	9	2.682481	5232.628	47.19%	72.90%	94.07%
6	7	5	9	4.222653	6649.822	60.72%	84.30%	77.26%
6	8	5	9	4.805378	5953.271	71.89%	60.08%	88.59%
6	9	5	9	3.821058	5675.936	42.35%	69.27%	25.61%
6	10	5	9	3.365291	6620.018	41.27%	69.55%	38.28%
7	2	5	9	1.948677	5240.209	65.95%	73.92%	88.01%
7	3	5	9	2.305366	5341.629	69.27%	86.41%	92.62%
7	4	5	9	3.444306	6982.858	68.61%	63.53%	21.00%
7	5	5	9	4.172763	6137.161	76.06%	85.33%	89.82%
7	6	5	9	4.664814	6078.318	63.27%	69.20%	71.54%

7	7	5	9	5.677214	6545.093	61.92%	64.96%	35.66%
7	8	5	9	3.214948	6022.6	76.80%	81.46%	59.32%
7	9	5	9	4.44247	4166.838	71.95%	61.67%	84.05%
7	10	5	9	3.238826	4666.427	73.82%	70.22%	31.63%
8	2	5	9	1.90784	4253.507	59.86%	76.25%	43.47%
8	3	5	9	2.074269	4362.776	40.26%	89.45%	23.42%
8	4	5	9	2.189113	6822.114	54.09%	89.28%	67.89%
8	5	5	9	2.548589	6580.563	78.29%	78.21%	52.24%
8	6	5	9	4.263323	4164.123	76.35%	85.43%	87.36%
8	7	5	9	4.556328	5734.651	57.60%	83.90%	55.63%
8	8	5	9	3.541259	5817.001	79.28%	82.84%	61.04%
8	9	5	9	2.316194	6074.077	40.35%	72.40%	68.11%
8	10	5	9	2.203313	5788.082	67.09%	80.75%	76.93%
9	2	5	9	1.646402	4600.971	58.06%	77.78%	26.60%
9	3	5	9	2.380421	4124.854	41.98%	77.12%	21.67%
9	4	5	9	3.017223	5407.847	72.42%	87.07%	79.59%
9	5	5	9	3.20817	4562.82	76.05%	80.03%	19.24%
9	6	5	9	2.999579	5940.213	76.14%	80.69%	59.81%
9	7	5	9	2.469813	5337.311	45.91%	68.88%	74.63%
9	8	5	9	3.790822	4471.09	66.17%	72.19%	50.53%
9	9	5	9	3.010036	6129.895	46.24%	63.83%	94.08%
9	10	5	9	2.595704	4073.156	56.04%	78.05%	88.82%
10	2	5	9	1.630792	4467.711	60.67%	80.73%	48.04%
10	3	5	9	1.837513	4069.679	67.04%	76.77%	36.83%
10	4	5	9	2.370662	4048.871	43.74%	72.95%	15.94%
10	5	5	9	2.574908	6061.672	50.51%	71.29%	28.77%
10	6	5	9	2.628592	5028.706	42.02%	71.60%	78.23%
10	7	5	9	2.891399	6665.027	46.89%	65.62%	87.81%
10	8	5	9	3.342442	5773.208	72.96%	78.87%	10.25%
10	9	5	9	2.100795	4845.541	62.94%	87.77%	53.56%
10	10	5	9	2.293439	4731.201	63.76%	75.49%	55.16%
2	2	1	9	1.44005	5603.819	67.27%	62.22%	58.20%
2	3	1	9	1.47649	6939.578	41.21%	71.62%	13.05%
2	4	1	9	1.65281	6914.243	60.75%	79.55%	44.01%
2	5	1	9	1.904181	5201.031	76.65%	84.37%	40.92%
2	6	1	9	1.720648	4415.191	51.42%	77.48%	35.64%
2	7	1	9	2.012478	5869.329	53.43%	89.45%	27.75%
2	8	1	9	1.696066	4674.533	40.51%	86.29%	58.50%
2	9	1	9	1.705685	6353.982	41.51%	82.68%	24.18%
2	10	1	9	1.807268	4816.942	69.07%	66.75%	88.92%
3	2	1	9	1.489144	6919.227	62.79%	63.55%	76.19%
3	3	1	9	1.946382	6229.532	55.05%	87.62%	32.96%
3	4	1	9	2.136577	5232.241	75.86%	80.44%	52.54%
3	5	1	9	1.916195	6978.441	68.28%	60.52%	86.80%
3	6	1	9	2.705961	4545.535	43.97%	61.56%	17.44%
3	7	1	9	2.637409	4686.406	51.24%	82.20%	55.77%
3	8	1	9	2.849644	6407.943	70.15%	66.20%	39.18%
3	9	1	9	2.226759	5698.911	77.66%	61.61%	46.31%
3	10	1	9	2.145809	4520.052	55.37%	70.67%	58.34%
4	2	1	9	1.54735	4322.637	75.34%	87.65%	25.20%

4	3	1	9	2.024365	5982.945	76.95%	68.44%	68.08%
4	4	1	9	2.227167	6515.336	41.98%	78.21%	57.34%
4	5	1	9	2.50474	6069.444	75.21%	80.57%	47.54%
4	6	1	9	3.492022	5594.368	48.83%	67.53%	26.90%
4	7	1	9	2.368544	4938.639	62.38%	86.04%	71.51%
4	8	1	9	3.069628	5560.84	41.76%	82.77%	22.89%
4	9	1	9	2.090296	6904.337	51.30%	64.71%	36.27%
4	10	1	9	2.294406	4024.409	42.15%	83.10%	85.72%
5	2	1	9	1.986648	5493.731	65.62%	89.89%	72.46%
5	3	1	9	2.202327	5847.405	53.39%	64.19%	53.13%
5	4	1	9	2.202953	5330.35	56.94%	64.29%	22.56%
5	5	1	9	2.283875	4989.036	76.27%	65.39%	36.44%
5	6	1	9	4.512492	5175.216	70.78%	74.78%	54.54%
5	7	1	9	4.974639	5008.025	62.89%	76.03%	89.08%
5	8	1	9	3.669113	5095.622	43.06%	84.55%	21.95%
5	9	1	9	3.897771	4298.108	71.69%	89.77%	71.69%
5	10	1	9	3.022862	5481.231	60.00%	85.37%	42.44%
6	2	1	9	1.733166	6536.804	60.20%	60.31%	60.02%
6	3	1	9	2.178169	4508.171	74.06%	84.54%	41.14%
6	4	1	9	3.033558	6078.618	68.96%	60.83%	93.74%
6	5	1	9	2.950887	4707.058	56.11%	82.66%	32.67%
6	6	1	9	4.122914	5839.906	46.32%	62.70%	82.23%
6	7	1	9	3.353638	5622.444	50.02%	81.13%	70.89%
6	8	1	9	4.316667	6250.5	60.25%	71.51%	68.57%
6	9	1	9	3.572484	5300.813	73.98%	83.34%	50.45%
6	10	1	9	3.471478	5268.283	41.79%	61.74%	76.41%
7	2	1	9	1.747459	6295.851	59.18%	63.81%	65.30%
7	3	1	9	2.039783	5231.799	73.44%	62.06%	57.20%
7	4	1	9	2.690576	5301.655	64.56%	80.46%	40.76%
7	5	1	9	3.286452	4217.749	63.76%	60.12%	16.07%
7	6	1	9	4.12126	6932.504	45.82%	72.05%	20.43%
7	7	1	9	4.704548	5466.429	77.50%	66.61%	14.29%
7	8	1	9	4.143885	6567.704	42.15%	84.57%	14.01%
7	9	1	9	4.894174	5262.231	50.75%	62.87%	16.33%
7	10	1	9	3.914561	6303.185	70.49%	88.78%	56.33%
8	2	1	9	2.098345	6860.533	64.73%	76.36%	36.61%
8	3	1	9	2.145338	4382.618	49.68%	85.66%	83.75%
8	4	1	9	3.152488	5799.8	64.01%	69.19%	62.08%
8	5	1	9	2.983145	5671.435	75.10%	71.60%	18.22%
8	6	1	9	2.953507	5497.164	70.86%	81.26%	56.41%
8	7	1	9	3.53253	5455.086	73.57%	75.46%	43.45%
8	8	1	9	5.271942	4965.704	43.83%	63.04%	37.18%
8	9	1	9	4.334111	4424.109	75.72%	72.92%	57.36%
8	10	1	9	2.233054	5322.095	62.50%	80.59%	54.64%
9	2	1	9	1.590563	6175.402	46.76%	88.59%	68.69%
9	3	1	9	2.34707	6782.044	43.39%	79.54%	87.08%
9	4	1	9	2.585546	5997.113	63.77%	64.64%	23.19%
9	5	1	9	2.296152	4584.697	77.18%	89.54%	86.34%
9	6	1	9	3.648157	5350.003	45.76%	60.42%	71.28%
9	7	1	9	5.226791	4934.012	57.36%	83.34%	68.53%

9	8	1	9	4.154711	6636.882	49.65%	72.77%	12.71%
9	9	1	9	3.230836	6188.947	79.62%	72.47%	10.95%
9	10	1	9	2.996458	4363.417	43.78%	87.57%	44.88%
10	2	1	9	1.638016	4807.614	42.14%	84.02%	59.36%
10	3	1	9	1.889867	6435.647	65.09%	61.05%	85.10%
10	4	1	9	2.594805	5406.404	77.87%	67.82%	88.04%
10	5	1	9	2.303616	5407.023	71.37%	87.32%	45.19%
10	6	1	9	3.6302	6835.23	64.23%	68.23%	18.89%
10	7	1	9	2.748885	4879.371	50.97%	71.28%	93.59%
10	8	1	9	3.174033	4361.048	59.26%	78.06%	22.40%
10	9	1	9	2.481308	4390.368	74.91%	85.84%	14.95%
10	10	1	9	2.391517	4536.488	49.05%	85.14%	54.86%
2	0	5	12	1.444683	5266.409	55.82%	71.89%	30.28%
3	0	5	12	1.551803	6762.737	67.03%	79.05%	25.22%
4	0	5	12	1.698445	4878.777	77.26%	76.93%	91.43%
5	0	5	12	1.614845	4696.946	70.81%	63.02%	90.01%
6	0	5	12	1.870514	6449.027	63.84%	79.00%	17.51%
7	0	5	12	2.027127	6003.186	63.14%	67.10%	45.36%
8	0	5	12	1.575579	6013.231	62.52%	72.47%	59.57%
9	0	5	12	1.566706	5599.044	49.72%	61.63%	18.18%
10	0	5	12	1.604523	4059.134	52.98%	83.62%	72.81%
11	0	5	12	1.47422	5657.56	46.17%	72.46%	71.53%
12	0	5	12	1.431061	4047.131	46.19%	68.69%	73.22%
13	0	5	12	1.426207	5581.364	40.71%	73.20%	65.34%
14	0	5	12	1.423424	4535.267	75.83%	70.15%	36.55%
15	0	5	12	1.490122	6983.943	46.31%	77.32%	55.89%
16	0	5	12	1.65205	6397.789	62.48%	64.58%	87.97%
2	0	1	12	1.398452	5582.897	62.71%	89.25%	33.70%
3	0	1	12	1.628868	6644.243	76.02%	72.28%	32.95%
4	0	1	12	1.521899	6362.561	69.87%	62.80%	45.76%
5	0	1	12	1.671092	5986.732	66.36%	82.76%	88.89%
6	0	1	12	1.949077	4533.409	76.67%	64.16%	68.25%
7	0	1	12	1.849528	6932.616	43.62%	69.67%	14.97%
8	0	1	12	1.68746	4914.744	67.25%	86.47%	18.27%
9	0	1	12	1.686768	6120.868	62.71%	81.55%	11.02%
10	0	1	12	1.616305	5949.02	47.35%	68.63%	46.48%
11	0	1	12	1.464916	4249.275	66.14%	70.09%	18.11%
12	0	1	12	1.421623	6985.78	51.99%	86.59%	14.53%
13	0	1	12	1.490679	6466.857	53.38%	75.94%	41.09%
14	0	1	12	1.40338	5141.888	46.95%	72.40%	87.06%
15	0	1	12	1.600232	6461.933	53.09%	77.73%	38.81%
16	0	1	12	1.674389	5636.615	79.91%	78.75%	31.54%
2	2	5	12	1.461085	6127.994	46.86%	76.51%	42.54%
2	3	5	12	1.476478	5945.504	77.23%	69.13%	51.74%
2	4	5	12	1.682663	4329.389	64.29%	67.75%	60.73%
2	5	5	12	1.750042	5815.515	69.83%	65.76%	57.61%
2	6	5	12	1.862163	5427.843	65.04%	62.74%	20.42%
2	7	5	12	1.843422	6600.251	62.79%	65.40%	55.03%
2	8	5	12	1.66803	5964.309	42.38%	87.41%	60.24%
2	9	5	12	1.780884	5559.62	48.44%	68.28%	30.79%

2	10	5	12	1.470029	5161.468	65.38%	78.89%	52.03%
3	2	5	12	1.46173	6768.581	61.93%	69.95%	93.60%
3	3	5	12	1.679184	5255.109	67.03%	85.59%	15.71%
3	4	5	12	1.64048	6887.217	79.65%	70.90%	85.66%
3	5	5	12	2.289023	4915.424	72.17%	88.02%	65.18%
3	6	5	12	2.285233	4869.571	71.00%	74.14%	56.52%
3	7	5	12	1.911641	4420.479	63.20%	61.42%	73.07%
3	8	5	12	1.797131	5804.25	71.44%	60.48%	10.08%
3	9	5	12	2.341177	5614.676	45.55%	83.10%	77.93%
3	10	5	12	2.047072	5446.076	69.55%	75.52%	37.49%
4	2	5	12	1.500156	6077	53.88%	81.24%	22.45%
4	3	5	12	1.670515	6046.039	78.16%	77.15%	30.98%
4	4	5	12	2.427568	6209.912	55.28%	85.41%	36.15%
4	5	5	12	2.319917	4735.997	64.81%	74.20%	42.15%
4	6	5	12	2.090049	5916.344	48.37%	68.58%	58.10%
4	7	5	12	2.873206	4543.078	50.27%	85.94%	74.31%
4	8	5	12	2.531465	5602.4	62.76%	71.32%	79.16%
4	9	5	12	2.14371	5823.899	59.45%	70.90%	12.53%
4	10	5	12	1.798077	4684.686	78.60%	64.33%	42.24%
5	2	5	12	1.821674	6674.937	69.31%	76.87%	46.54%
5	3	5	12	2.296222	5556.68	59.22%	60.06%	30.15%
5	4	5	12	2.5217	5475.911	45.52%	71.66%	65.86%
5	5	5	12	2.775432	4002.783	46.88%	75.46%	42.76%
5	6	5	12	2.283073	5803.842	51.28%	79.56%	69.49%
5	7	5	12	2.707792	4495.964	45.88%	78.57%	14.60%
5	8	5	12	3.23917	6634.232	53.34%	68.12%	47.83%
5	9	5	12	2.977865	6410.93	74.07%	88.49%	56.55%
5	10	5	12	2.248432	5011.755	49.71%	85.31%	13.75%
6	2	5	12	1.746739	4450.031	70.93%	89.04%	35.38%
6	3	5	12	2.195745	6899.177	43.51%	80.11%	29.71%
6	4	5	12	2.363269	4472.068	50.60%	77.13%	30.25%
6	5	5	12	2.779999	4074.781	65.14%	74.10%	41.58%
6	6	5	12	4.429271	6752.71	53.38%	66.08%	92.60%
6	7	5	12	3.258756	6791.01	40.96%	61.52%	24.75%
6	8	5	12	2.866703	6714.475	74.69%	84.12%	10.21%
6	9	5	12	2.201262	5621.421	72.47%	79.08%	11.32%
6	10	5	12	2.916968	6187.463	40.75%	76.82%	51.48%
7	2	5	12	1.723638	4569.209	78.57%	62.89%	51.47%
7	3	5	12	2.531312	6316.625	48.72%	61.21%	22.75%
7	4	5	12	2.611385	5029.259	70.05%	78.85%	66.59%
7	5	5	12	4.124759	5323.544	48.85%	74.88%	54.79%
7	6	5	12	2.6885	4664.856	55.91%	76.84%	81.02%
7	7	5	12	3.355224	6304.583	79.62%	79.86%	60.58%
7	8	5	12	4.966053	6525.59	71.36%	87.40%	28.51%
7	9	5	12	3.219768	4256.279	60.34%	74.69%	85.40%
7	10	5	12	2.379206	4557.264	79.08%	84.87%	26.27%
8	2	5	12	1.885549	6650.132	67.64%	77.94%	89.96%
8	3	5	12	2.269166	4674.195	56.81%	73.76%	62.74%
8	4	5	12	3.052568	5725.672	76.22%	69.03%	31.69%
8	5	5	12	3.706105	5364.621	56.85%	88.40%	18.63%

8	6	5	12	3.588514	4268.031	44.88%	65.96%	18.52%
8	7	5	12	4.865904	5490.392	54.34%	66.68%	85.48%
8	8	5	12	2.642822	5757.749	42.62%	69.48%	82.17%
8	9	5	12	2.598039	4040.755	44.21%	75.09%	91.76%
8	10	5	12	2.266596	4573.233	77.75%	61.82%	34.45%
9	2	5	12	1.625671	6896.652	74.35%	61.29%	13.97%
9	3	5	12	1.773344	5717.721	64.69%	60.85%	24.03%
9	4	5	12	2.355656	6607.35	49.37%	67.89%	85.53%
9	5	5	12	2.361061	4340.556	59.82%	67.54%	57.17%
9	6	5	12	3.710294	5928.141	76.32%	74.23%	22.69%
9	7	5	12	3.669359	5720.435	57.11%	76.45%	79.95%
9	8	5	12	2.537511	4838.192	78.67%	60.33%	11.33%
9	9	5	12	2.102292	4308.012	47.83%	64.57%	34.54%
9	10	5	12	2.088813	5150.571	56.88%	82.88%	19.05%
10	2	5	12	1.554032	5571.286	65.55%	77.49%	78.36%
10	3	5	12	1.992077	4451.994	49.53%	64.65%	81.74%
10	4	5	12	1.859684	5777.049	52.95%	85.80%	45.01%
10	5	5	12	2.343344	5601.673	75.11%	75.85%	14.14%
10	6	5	12	2.935564	5134.321	75.41%	60.74%	50.22%
10	7	5	12	2.963336	6163.696	66.09%	71.55%	18.24%
10	8	5	12	3.183641	5592.243	43.66%	78.96%	80.68%
10	9	5	12	2.71943	6143.487	40.02%	63.63%	14.52%
10	10	5	12	2.335599	6931.379	53.44%	88.03%	14.71%
2	2	1	12	1.44098	5900.571	47.16%	82.58%	32.28%
2	3	1	12	1.572299	4136.444	75.90%	84.13%	36.91%
2	4	1	12	1.780075	6396.793	69.15%	76.55%	43.33%
2	5	1	12	1.645149	5134.905	63.58%	80.45%	60.31%
2	6	1	12	1.622711	5518.909	74.26%	89.57%	54.92%
2	7	1	12	1.917058	6665.21	56.48%	68.60%	15.43%
2	8	1	12	1.595842	4749.479	55.65%	65.66%	92.62%
2	9	1	12	1.770829	6196.883	47.14%	83.21%	42.91%
2	10	1	12	1.509375	5864.459	46.73%	67.49%	72.83%
3	2	1	12	1.631281	4588.174	79.59%	80.51%	60.29%
3	3	1	12	1.798976	4024.825	46.76%	74.79%	59.97%
3	4	1	12	1.979582	6333.713	66.23%	86.66%	77.24%
3	5	1	12	2.451506	4326.176	60.84%	67.05%	32.72%
3	6	1	12	2.75454	6734.684	58.69%	82.11%	94.43%
3	7	1	12	2.224356	4622.615	63.61%	70.27%	40.15%
3	8	1	12	1.946363	5487.337	52.21%	69.54%	94.00%
3	9	1	12	2.482791	6288.9	56.44%	70.97%	70.01%
3	10	1	12	1.697999	6150.196	53.01%	80.89%	67.74%
4	2	1	12	1.611787	6355.028	52.26%	85.01%	70.67%
4	3	1	12	1.94903	6268.057	57.61%	87.09%	18.00%
4	4	1	12	2.401134	4429.923	59.20%	75.54%	77.82%
4	5	1	12	2.981056	6246.369	44.31%	70.59%	25.27%
4	6	1	12	2.595211	6715.425	67.72%	79.28%	84.39%
4	7	1	12	3.088217	6001.075	61.06%	72.27%	30.38%
4	8	1	12	3.03231	6796.512	42.19%	77.75%	44.56%
4	9	1	12	2.931976	4056.856	65.15%	72.72%	13.75%
4	10	1	12	2.546533	6010.575	61.56%	89.00%	25.44%

5	2	1	12	1.726103	6335.981	73.68%	69.07%	73.18%
5	3	1	12	2.300157	5636.654	43.66%	75.88%	89.44%
5	4	1	12	2.026663	6606.366	72.68%	68.41%	31.12%
5	5	1	12	2.659066	6854.091	68.79%	69.78%	93.18%
5	6	1	12	3.341814	6045.111	60.15%	85.52%	68.59%
5	7	1	12	2.738453	5600.708	56.53%	89.79%	18.68%
5	8	1	12	2.39714	6354.679	79.57%	85.17%	38.94%
5	9	1	12	3.30175	6561.282	76.54%	69.45%	46.14%
5	10	1	12	2.350145	5099.969	73.89%	67.28%	69.37%
6	2	1	12	2.018319	6918.759	43.49%	87.43%	12.59%
6	3	1	12	2.196444	6399.957	67.63%	73.20%	23.97%
6	4	1	12	3.213606	4837.495	63.51%	71.40%	73.90%
6	5	1	12	4.07926	4881.755	77.55%	72.42%	46.29%
6	6	1	12	3.641704	6967.972	44.76%	82.70%	77.57%
6	7	1	12	4.156522	5408.532	69.72%	60.37%	74.78%
6	8	1	12	4.245297	6971.325	55.59%	65.47%	20.95%
6	9	1	12	2.669138	6250.781	76.95%	81.81%	10.79%
6	10	1	12	3.224888	6134.098	48.48%	86.34%	54.62%
7	2	1	12	1.668945	4474.442	50.34%	82.19%	88.90%
7	3	1	12	2.939065	6330.647	54.21%	77.99%	26.00%
7	4	1	12	2.697762	6128.666	42.09%	73.05%	12.24%
7	5	1	12	3.693867	6800.517	72.28%	85.02%	48.87%
7	6	1	12	4.665078	5079.688	48.97%	82.47%	17.66%
7	7	1	12	4.849512	5577.317	45.89%	79.75%	81.98%
7	8	1	12	4.883398	5962.066	59.57%	63.15%	57.27%
7	9	1	12	4.218246	5455.266	46.61%	77.87%	85.32%
7	10	1	12	2.310553	5273.045	45.52%	81.18%	25.02%
8	2	1	12	1.972163	4527.72	74.03%	88.72%	35.83%
8	3	1	12	2.09285	5777.249	55.65%	65.38%	91.05%
8	4	1	12	2.774443	4739.677	79.25%	86.33%	26.71%
8	5	1	12	3.169075	6448.574	73.99%	85.97%	43.62%
8	6	1	12	3.881448	4987.444	67.86%	77.57%	56.52%
8	7	1	12	4.96209	4478.821	70.20%	62.95%	14.75%
8	8	1	12	4.712593	4397.27	45.58%	75.40%	92.22%
8	9	1	12	3.204719	5453.079	60.01%	64.58%	78.24%
8	10	1	12	2.571654	4736.078	55.13%	69.20%	66.63%
9	2	1	12	1.803217	6178.779	48.52%	82.83%	91.23%
9	3	1	12	2.025833	4759.81	49.00%	72.60%	69.23%
9	4	1	12	2.181666	6426.636	53.13%	80.64%	17.77%
9	5	1	12	3.674337	6870.869	50.65%	78.22%	11.61%
9	6	1	12	2.663957	4186.004	77.50%	75.54%	17.22%
9	7	1	12	3.182558	5143.321	63.99%	63.24%	73.95%
9	8	1	12	4.325952	5900.361	62.49%	67.07%	92.00%
9	9	1	12	3.277415	4334.521	70.95%	65.86%	43.09%
9	10	1	12	3.100501	6450.437	69.23%	81.78%	91.75%
10	2	1	12	1.60302	5010.492	79.86%	62.78%	56.90%
10	3	1	12	1.876462	5298.158	64.20%	82.86%	21.49%
10	4	1	12	2.515039	4039.489	49.25%	64.04%	91.14%
10	5	1	12	2.140738	4624.015	77.87%	85.52%	57.44%
10	6	1	12	2.417264	6211.578	70.65%	76.75%	14.92%

10	7	1	12	3.651545	5514.601	65.91%	78.91%	63.89%
10	8	1	12	3.441627	4414.092	51.96%	87.57%	13.11%
10	9	1	12	2.50026	5632.385	78.30%	76.82%	49.30%
10	10	1	12	2.2345	6486.48	55.46%	68.36%	24.34%
2	0	5	15	1.451047	4518.945	65.51%	64.80%	92.31%
3	0	5	15	1.516494	6103.062	45.51%	68.81%	73.04%
4	0	5	15	1.643859	5533.69	54.91%	78.91%	47.30%
5	0	5	15	1.524687	6578.18	52.45%	78.97%	40.65%
6	0	5	15	1.645392	5803.143	61.45%	88.18%	13.45%
7	0	5	15	1.877991	6703.71	47.56%	61.15%	69.68%
8	0	5	15	1.916631	4612.173	78.65%	71.93%	57.77%
9	0	5	15	1.529694	6108.471	41.22%	64.25%	69.31%
10	0	5	15	1.614828	6138.235	60.56%	64.83%	41.11%
11	0	5	15	1.569814	6906.445	47.03%	69.63%	37.01%
12	0	5	15	1.39364	4232.721	79.11%	87.21%	18.74%
13	0	5	15	1.400354	5365.075	49.71%	79.50%	76.93%
14	0	5	15	1.417694	5343.561	71.91%	68.54%	25.16%
15	0	5	15	1.559777	4843.945	57.69%	77.91%	53.41%
16	0	5	15	1.663665	6066.191	76.28%	78.51%	23.86%
2	0	1	15	1.430455	6006.726	46.33%	86.74%	24.77%
3	0	1	15	1.503258	5876.013	72.63%	65.64%	20.93%
4	0	1	15	1.766049	6859.353	44.12%	60.59%	29.06%
5	0	1	15	1.745701	6592.176	73.82%	66.70%	17.81%
6	0	1	15	1.680288	4442.077	46.39%	65.57%	50.75%
7	0	1	15	1.924106	6274.987	65.87%	63.52%	69.17%
8	0	1	15	2.00654	6186.488	48.26%	75.38%	67.39%
9	0	1	15	1.837865	4096.082	67.33%	64.65%	16.38%
10	0	1	15	1.742975	5136.5	77.62%	85.79%	86.50%
11	0	1	15	1.530195	4813.144	42.82%	73.63%	74.09%
12	0	1	15	1.465704	5831.784	40.26%	65.12%	11.96%
13	0	1	15	1.412269	5506.769	73.20%	73.93%	39.33%
14	0	1	15	1.44358	4986.739	43.01%	79.59%	32.81%
15	0	1	15	1.613151	5118.231	50.00%	89.14%	70.01%
16	0	1	15	1.485731	6562.797	52.55%	61.94%	82.21%
2	2	5	15	1.417893	6453.927	61.34%	84.11%	65.05%
2	3	5	15	1.485769	5412.95	52.87%	62.29%	78.77%
2	4	5	15	1.675856	6161.698	51.08%	87.26%	28.37%
2	5	5	15	1.597451	6144.602	67.28%	88.73%	67.75%
2	6	5	15	1.727192	6533.707	48.13%	74.44%	10.48%
2	7	5	15	1.654999	4554.426	41.03%	89.07%	34.50%
2	8	5	15	1.73914	5886.928	53.03%	68.95%	60.55%
2	9	5	15	1.644743	5119.13	48.74%	81.06%	24.27%
2	10	5	15	1.563236	5426.065	49.06%	62.00%	45.98%
3	2	5	15	1.487509	6530.966	54.92%	68.30%	86.29%
3	3	5	15	1.574902	6246.419	49.64%	69.86%	21.16%
3	4	5	15	1.953169	4470.392	75.38%	68.28%	46.76%
3	5	5	15	1.718736	5129.472	55.55%	63.74%	81.41%
3	6	5	15	1.770977	6873.323	44.85%	71.86%	43.49%
3	7	5	15	2.458832	6967.359	77.04%	70.37%	59.75%
3	8	5	15	2.399697	6458.704	43.75%	65.08%	79.84%

3	9	5	15	1.793454	5098.182	55.79%	78.92%	18.73%
3	10	5	15	1.998945	6021.914	40.68%	83.39%	60.66%
4	2	5	15	1.547875	6765.329	75.76%	87.47%	62.54%
4	3	5	15	2.057877	4480.917	71.39%	73.00%	19.30%
4	4	5	15	1.740962	4022.95	66.97%	86.61%	76.71%
4	5	5	15	1.836519	4791.481	67.43%	63.28%	34.93%
4	6	5	15	3.09664	4182.048	70.70%	60.77%	89.97%
4	7	5	15	2.385188	5106.934	63.86%	79.29%	94.88%
4	8	5	15	2.598308	4487.447	44.52%	82.87%	24.14%
4	9	5	15	2.726485	5199.043	41.95%	61.76%	40.55%
4	10	5	15	2.420044	4646.914	65.67%	79.84%	62.00%
5	2	5	15	1.621348	5459.557	40.61%	85.87%	81.42%
5	3	5	15	1.808544	4114.696	75.12%	68.14%	78.58%
5	4	5	15	2.036213	6584.107	63.90%	64.66%	15.75%
5	5	5	15	2.204156	5035.391	58.47%	88.10%	92.80%
5	6	5	15	2.380643	5845.312	49.38%	82.62%	71.47%
5	7	5	15	2.73583	5758.819	77.29%	81.55%	16.13%
5	8	5	15	3.557184	6604.702	44.05%	76.27%	93.38%
5	9	5	15	2.873829	6030.541	61.82%	67.20%	91.00%
5	10	5	15	2.077948	6738.589	69.25%	79.65%	82.42%
6	2	5	15	1.708306	6635.093	70.41%	85.21%	81.91%
6	3	5	15	2.366666	5002.38	70.05%	81.56%	72.52%
6	4	5	15	2.511025	4362.821	42.78%	62.75%	10.40%
6	5	5	15	2.790344	6854.935	66.57%	73.58%	16.06%
6	6	5	15	3.753715	4992.582	41.56%	69.31%	89.79%
6	7	5	15	3.651676	4961.57	62.70%	79.53%	26.10%
6	8	5	15	4.345792	6424.133	55.22%	81.35%	78.15%
6	9	5	15	2.772453	5728.403	57.14%	77.81%	83.86%
6	10	5	15	2.320181	6797.161	76.30%	72.61%	10.54%
7	2	5	15	1.993595	5460.887	71.61%	76.01%	20.15%
7	3	5	15	2.04045	6198.84	40.83%	75.75%	11.68%
7	4	5	15	3.31204	4122.541	77.85%	83.53%	20.01%
7	5	5	15	2.946222	4073.789	59.13%	60.21%	68.28%
7	6	5	15	3.816297	5202.691	55.36%	66.10%	21.95%
7	7	5	15	2.928497	4988.913	44.72%	62.16%	39.74%
7	8	5	15	3.458696	5038.305	79.58%	88.10%	45.22%
7	9	5	15	3.848755	5459.957	60.89%	87.42%	89.79%
7	10	5	15	2.59936	4513.286	66.41%	72.15%	94.41%
8	2	5	15	1.665772	5290.649	48.14%	87.15%	40.46%
8	3	5	15	1.742765	4562.857	74.32%	66.33%	82.94%
8	4	5	15	2.713494	5167.831	56.04%	66.66%	16.79%
8	5	5	15	2.555775	6474.339	68.60%	88.80%	53.54%
8	6	5	15	3.397918	4759.359	46.57%	71.83%	74.57%
8	7	5	15	3.616731	5829.502	49.93%	82.97%	71.56%
8	8	5	15	3.773401	4686.66	41.10%	89.04%	20.32%
8	9	5	15	3.131951	6388.616	69.90%	83.19%	40.27%
8	10	5	15	2.648765	5188.268	69.55%	80.70%	37.24%
9	2	5	15	1.669855	6591.861	62.99%	63.10%	52.45%
9	3	5	15	2.036652	4473.353	72.65%	64.05%	15.01%
9	4	5	15	2.118813	6101.477	79.28%	70.59%	34.31%

9	5	5	15	2.263822	4773.395	53.98%	64.19%	44.69%
9	6	5	15	2.195191	6917.068	75.94%	73.35%	28.56%
9	7	5	15	3.095842	4613.857	46.21%	73.52%	39.66%
9	8	5	15	2.499174	5703.016	59.00%	77.50%	38.22%
9	9	5	15	2.573533	5004.127	50.45%	78.60%	61.32%
9	10	5	15	2.631631	5220.183	67.97%	62.00%	39.21%
10	2	5	15	1.519015	6547.37	71.72%	76.18%	66.20%
10	3	5	15	1.852851	4245.22	73.72%	84.34%	43.94%
10	4	5	15	1.854245	6387.556	61.86%	66.93%	33.53%
10	5	5	15	2.409902	6867.012	53.69%	83.06%	62.60%
10	6	5	15	2.26528	4300.26	78.14%	78.74%	56.21%
10	7	5	15	2.310331	4578.696	69.44%	87.55%	40.53%
10	8	5	15	3.007298	5865.453	65.15%	77.30%	78.94%
10	9	5	15	2.109392	4473.011	53.97%	60.64%	74.60%
10	10	5	15	2.300005	6067.47	42.01%	83.37%	46.59%
2	2	1	15	1.391371	6948.3	58.58%	74.17%	63.80%
2	3	1	15	1.440007	6234.53	42.03%	80.67%	62.94%
2	4	1	15	1.586105	6837.434	63.75%	73.35%	53.08%
2	5	1	15	1.726088	4789.568	57.35%	72.42%	10.37%
2	6	1	15	1.788451	4353.592	50.37%	77.89%	59.46%
2	7	1	15	1.777599	5315.479	76.57%	70.84%	89.60%
2	8	1	15	1.63791	6451.701	76.25%	68.07%	61.46%
2	9	1	15	1.688502	4672.997	62.01%	72.61%	18.88%
2	10	1	15	1.526347	5056.96	44.60%	69.22%	82.42%
3	2	1	15	1.495018	4171.33	58.62%	82.84%	11.02%
3	3	1	15	1.676003	6105.157	63.74%	85.63%	80.76%
3	4	1	15	1.759086	6001.089	58.28%	85.96%	35.89%
3	5	1	15	2.114104	5502.014	66.07%	79.64%	33.66%
3	6	1	15	2.288562	6041.351	56.16%	79.98%	26.23%
3	7	1	15	2.113852	6157.102	42.69%	75.78%	61.23%
3	8	1	15	2.072897	6094.63	57.37%	65.22%	80.97%
3	9	1	15	1.993588	6983.832	75.98%	89.45%	71.40%
3	10	1	15	2.103953	4354.23	64.56%	82.44%	90.78%
4	2	1	15	1.51955	5638.843	79.53%	74.02%	68.64%
4	3	1	15	2.107781	6506.129	56.14%	85.48%	29.03%
4	4	1	15	2.076699	4910.513	56.54%	62.48%	65.62%
4	5	1	15	2.570927	5710.182	48.08%	77.08%	57.89%
4	6	1	15	3.231496	6052.232	63.95%	80.27%	28.56%
4	7	1	15	2.805835	4180.19	73.48%	65.28%	70.80%
4	8	1	15	3.099071	4462.416	59.10%	83.82%	51.75%
4	9	1	15	2.275345	4693.499	67.31%	87.71%	42.76%
4	10	1	15	2.530579	4301.638	60.83%	75.21%	59.17%
5	2	1	15	1.877292	6106.755	52.38%	77.06%	69.51%
5	3	1	15	2.140571	6941.89	68.03%	63.89%	61.35%
5	4	1	15	1.987896	4181.908	49.18%	65.37%	68.36%
5	5	1	15	2.579191	6154.421	74.62%	88.16%	21.27%
5	6	1	15	2.719416	4037.005	54.44%	86.39%	67.76%
5	7	1	15	4.429687	6590.958	64.59%	71.46%	66.19%
5	8	1	15	4.15921	6602.342	60.14%	71.88%	45.25%
5	9	1	15	3.089592	6583.242	75.01%	76.73%	57.53%

5	10	1	15	1.959429	6356.514	78.40%	84.76%	78.76%
6	2	1	15	1.591934	4210.679	67.31%	77.87%	23.46%
6	3	1	15	2.576848	5030.794	52.22%	71.81%	65.76%
6	4	1	15	2.575495	4064.814	63.63%	80.86%	52.70%
6	5	1	15	4.162169	5301.122	49.77%	78.52%	79.71%
6	6	1	15	4.724855	6291.76	65.12%	67.07%	57.87%
6	7	1	15	4.671141	5749.136	56.79%	86.69%	28.90%
6	8	1	15	4.550802	6068.363	72.96%	74.54%	67.66%
6	9	1	15	3.02334	5799.035	78.18%	61.30%	38.17%
6	10	1	15	2.76411	6923.383	72.05%	86.54%	86.40%
7	2	1	15	1.742515	4397.696	68.80%	83.23%	71.39%
7	3	1	15	3.055689	5735.7	50.38%	83.07%	80.07%
7	4	1	15	3.678269	6309.704	64.32%	62.95%	77.46%
7	5	1	15	4.630676	6834.165	55.74%	69.36%	69.58%
7	6	1	15	4.386768	5329.003	50.79%	84.77%	52.56%
7	7	1	15	4.503542	5670.9	58.70%	70.52%	45.59%
7	8	1	15	4.430773	6354.55	54.46%	80.90%	76.46%
7	9	1	15	4.723682	6608.616	77.08%	62.61%	78.98%
7	10	1	15	3.735532	5282.091	73.72%	69.65%	44.74%
8	2	1	15	1.988455	4140.074	58.87%	67.30%	68.83%
8	3	1	15	1.825073	4263.28	71.22%	76.98%	28.73%
8	4	1	15	2.265582	5183.962	72.73%	61.14%	82.47%
8	5	1	15	3.861518	4222.482	68.39%	80.71%	45.27%
8	6	1	15	4.564475	6262.276	66.14%	82.90%	36.35%
8	7	1	15	2.82743	6737.864	63.84%	69.23%	67.79%
8	8	1	15	4.843589	6974.907	50.74%	80.96%	85.24%
8	9	1	15	3.066159	5812.053	68.20%	66.54%	65.56%
8	10	1	15	3.466196	4458.771	74.28%	60.37%	58.67%
9	2	1	15	1.625631	6850.159	54.34%	67.78%	78.01%
9	3	1	15	2.227873	5593.102	70.00%	82.37%	93.50%
9	4	1	15	2.978891	6052.371	56.70%	81.35%	37.57%
9	5	1	15	3.284593	6039.234	54.74%	73.39%	42.53%
9	6	1	15	3.700492	5816.189	66.22%	76.32%	30.82%
9	7	1	15	4.369513	4899.545	60.92%	82.83%	45.32%
9	8	1	15	3.2826	6095.797	74.31%	77.78%	94.23%
9	9	1	15	3.456946	4251.738	75.00%	67.62%	43.14%
9	10	1	15	2.542283	6893.444	57.07%	78.98%	49.45%
10	2	1	15	1.530608	6694.098	77.81%	82.24%	43.88%
10	3	1	15	2.056249	6579.357	70.55%	87.68%	65.52%
10	4	1	15	2.525998	5379.282	68.92%	87.10%	57.16%
10	5	1	15	2.447039	4063.032	48.67%	72.46%	24.90%
10	6	1	15	3.178444	6201.557	62.83%	62.46%	29.95%
10	7	1	15	3.215884	5428.336	74.51%	83.07%	58.71%
10	8	1	15	2.633993	5900.22	64.15%	60.83%	22.97%
10	9	1	15	2.394674	6034.6	52.91%	87.84%	92.24%
10	10	1	15	2.560461	4689.598	45.17%	76.38%	55.79%

Figure 21: Numerical results obtained from experiments.