CS767 Project Report: Automatic Division of Labour

By Daniel Flower

I. Introduction

"*A jack of all trades will never be rich*" – proverb

Multiagent systems have great potential in many areas, from searching the Web to creating a habitat on Mars. However, as the number of agents in a system grows it becomes increasingly difficult to control the agents centrally. Aside from the communication overhead and the difficulty in effectively controlling many agents to perform a task, central control is greatly impaired when the central controller either cannot access all the information available to the individual agents, or is unable to make accurate estimates of the global situation. It of course becomes impossible when not all agents can be controlled by the one controller, which occurs in cases where competing parties are present in the same environment. Other benefits include scalability and fault tolerance. It should be possible for many agents to stop working, and for the system to still function, albeit not so efficiently.

The observation that society is like one large multiagent system inspired the use of economic principles in controlling individual agents. Specifically, agents can be programmed to act in their own local self-interest in order to achieve global goals. This paradigm has been very effective and has led to a deeper use of economic theory in multiagent systems, such as market equilibrium, game theory, auctions etc.

Description of problem

An extremely important component of economic success is the division of labour. This is the idea that through specialisation and trade, society can produce a lot more goods than it could otherwise because each part of the production is being carried out by a specialist. This division of labour is equally important in a multiagent system. While it is possible to manually assign each agent to a specific task, it would be difficult to do this in a large system, and it would be difficult to calculate the optimal allocation. The goal of this project is therefore to create a system that automatically finds the near-optimal allocation of agents without central control, and to find which types of situations would benefit most from the division of labour.

This is an interesting problem because the use of economic theory in multiagent systems has been successful and looks to become more popular with ever increasingly complex systems that will be developed. The allocation of agents is just a small sub-problem in this paradigm, however it is an important problem because the effectiveness of an economy is greatly influenced by the allocation of its resources.

Example of problem

For this project, the example of a farm is used. Different goods may be produced, for example onions, potatoes and tomatoes. Different parties will be able to work on the farm, and each party may have multiple agents. Each party will have a certain amount of each product that they would like to obtain for the lowest cost possible. A party with one agent that would like some potatoes and some tomatoes may choose to first produce tomatoes, and then produce potatoes. However, the more experience an agent has producing a certain product, the more efficient it becomes. Therefore, it would be better for the agent to specialise in, say, potatoes, and then trade some of their potatoes for tomatoes. Or, if the agent calculates that it is more efficient to produce tomatoes, then it should specialise in tomatoes and then trade some for potatoes.

While this may seem an unrealistic example, the idea behind it is similar to many other multiagent systems. For example, a robot working on the Mars robotic outpost may choose to specialise in an area where there is demand, e.g. in gathering rocks. In disaster recovery, one agent may decide to only search for people and broadcast their findings, leaving the retrieval to another agent. In fact, in any system where an agent can choose between a set of roles, the situation is analogous to the farming example. Even in cases where agents cannot change their function (for example, in order to mine ore, an agent must have drilling equipment), a simulation could first be run where agents *can* perform any task in order to find a suitable allocation.

Evaluation

The success of the system comes in two parts: 1, the ability of parties to gain the products they require even when some agents fail working, and 2, for the system to work at a near-optimal level.

1) Firstly, each party must be able to obtain the goods they require without central control, and all products must be produced even if many agents break down. Therefore, each individual agent will need to choose what to produce on its own.
2) Secondly, the system should perform better than a system where resources were allocated centrally. Because there is no single way a centrally controlled system would approach the problem, it will be difficult to evaluate whether the system performs better than others. So instead, the performance of the system will be evaluated when certain parameters are changed in order to find those situations where the division of labour is most important.

Overview of Approach

The virtual farm will need to have virtual products that an agent can produce, and it should work whether there are 3 or 103 products to choose from. Each product has characteristics such as difficulty of processing, barrier to entry[1], cost to produce per unit time step, and market price per unit sold. Other than market price, which is calculated automatically, the actual values specified are not important, as this is a simulation only with no need to mimic a real farm (these numbers would need to be found experimentally, or perhaps automatically, in a real-life system).

An agent using these product data while taking into account its own skill in processing that product will be able to calculate the cost and reward of each option, and hence choose the option that is best for that particular agent. As more agents work in a certain area on the farm, the harder it is to produce those goods, which is a simulation of diseconomies of scale[2]. This helps to spread out the roles of the agents, and mimics real-life economics.

So, the agents are the suppliers in the market while the parties are the consumers. Demand arises from the wants of the parties. Each party is given a certain amount of money and the number of each product that they desire to purchase. In the case where there is only one party in the market, they will therefore be purchasing the products from their own agents. While this may seem unnecessary, it is done on the hypothesis that this is the best way to allocate their own agents. In order to simplify the project, if there are *n* time steps in the simulation, then each party will try to purchase TotalDemand/*n* units of each product that it demands. In other words, parties do not wait until they think they can get the best price, and instead just take the current market price. This is therefore equivalent to the situation where a party needs a certain amount of goods per time step, for example a restaurant that needs five boxes of eggs each day, no matter what the cost.

---

[1] This is an economic term describing the fixed cost in starting a new production.

[2] Note: Because the parties will not have any fixed costs, there will be no economies of scale.

When a party buys from one of its own agents, the money moves from the party to the agent, but because the party owns the agent the net cost is therefore zero. However, this fact will be ignored by the parties when deciding whom to buy from because they are better off buying the product at a lower price and having another party buy from their agent at the higher price (which may or may not happen).

At the end of each time step trading occurs. Each agent makes available the products it produced and the parties are then free to purchase as they wish. After trading has finished, for each product the market price is adjusted depending on whether the demand exceeded supply, supply exceeded demand, or supply equalled demand. This change in price affects the decisions made by the agents in the following time step and hence this is what controls the allocation of resources effectively even in cases where, for example, many of the agents are unexpectedly taken out of the system.

At the end of the simulation all the data is made available in order to evaluate the system.

II. Related Research

While there have been many studies in economically inspired multiagent systems, concentration on the division of labour has normally been studied in systems not using market principles, such as in ant colonies. Jones & Mataric [1] studied adaptive division of labour in robots, which were picking up different coloured pucks. If 70% of the pucks were blue and 30% red, then 70% of the robots should have the job of picking up the blue pucks, and 30% the red. The simulation was very simplistic in that the robots were not told how many pucks there were, nor how many other robots there were, and the robots were not allowed to communicate. The robots simply looked at their memories to see how many of the different coloured pucks they had seen and how many robots they had seen picking up the different pucks and based their

decision on which colour to pick up using a probabilistic function that was influenced by what they had seen. When they changed the ratio of blue to red pucks in the simulation, the ratio of robots picking up those colours changed accordingly even though they were not told that the ratio had changed. These promising results give reason to investigate the division of labour further in other types of simulations.

Another interesting study in the area of having agents specialise was done by Smith et al. in [3]. Here, they used genetic algorithms to evolve just 3 agents that lived in an economy. They found that if they started with "Jacks of all trades", they soon evolved into specialists, with each agent specialising into a distinct area from the others, out of the 3 areas that were available to them. This study shows that machine learning may play a part in the division of labour, however on the other hand machine learning can be resource intensive and may have problems adapting when the market changes. This gives reason to explore other ways to achieve division of labour.

Even in economic-based multiagent research that has ignored the division of labour there are important lessons to be learnt that are relevant to this project. For example, in [2], robots could trade goods for energy. If some robots became too rich, they would have no need to trade, and therefore other robots would not receive any energy and die out. By making resources scarcer, the robots were encouraged to trade more, and fewer robots died out. This shows that it is important to get the balance of the economy correct.

Finally, Mainland et al. [4] showed the potential in using market oriented programming when there are very many agents in the system. In the study, sensors successfully tracked objects moving through a field while using minimum energy. The sensors could perform different tasks such as scanning, sleeping or broadcasting and they were rewarded when their actions brought good results. A very good innovation in this example was the use of reinforcement learning to estimate the profit of executing each action, which is important because a market economy should be self-regulating, and the less manually set parameters the better. Also, when some sensors were

given a lot more energy than others, they tended to specialise more on tasks that required higher energy. Other than that, the division of labour was not mentioned and it would be interesting to see if a system such as that could benefit from specialisation.

III. Description of Approach

This project was implemented as a console application using C#.NET. The following table shows the object model that was created:

OBJECT	PROPERTIES	VALUES	NOTES
Farm	Parties	List of parties in the simulation	
	Products	List of products in the simulation	
	Number Of Time steps	Integer	The number of time steps that the simulation will run for.
Party	Agents	List of agent objects	
	Money	Integer. This may be negative if this party owes money.	Decremented with each purchase.
	Demands	An amount of each product demanded.	Decremented with each purchase until it reaches zero for each product. This is a subset of all available products.
	Name	String	Used for reporting purposes only.
Agent	List of skills	A number between zero and one for each type of product.	Incremented a small amount for the skill currently being worked on.
	Current	Product	The product that this agent is currently working on.

	Money earned	Integer	Incremented with each sale.
	Money spent	Integer	Incremented at each time step.
Product	Name	String	
	Market price	Integer	This is calculated automatically and fluctuates depending on supply and demand.
	Number of agents	Integer	This will keep track of the number of agents currently processing this product.
	Cost per time step	Integer	The variable cost of producing one type of this product per time step.
	Barrier to entry	Integer	The fixed cost of producing a product. This cost is only incurred when a new product is starting to be worked on by an agent.
	Difficulty	In integer between zero and one	While this difficulty number will remain constant, the actual difficulty in processing this product fluctuates depending on the number of other agents working on the same raw material and each agent's own skill in creating this product.

When the simulation is run, the following tasks are run *NumberOfTimesteps* times:

1) Each agent chooses what to work on
2) Each agent produces some products
3) Each party purchases products from the agents
4) The market price of each product is adjusted

Those four steps will now be explained in detail.

1) Each agent chooses what to work on

Each agent calculates the profit (i.e. revenue less expense) it will make by producing each product and chooses the product with the greatest profit, or chooses to work on nothing if no product will render a profit. Expense is

simply the fixed cost plus the variable cost. The fixed cost is 0 for the product currently being worked on and the “barrier to entry” of each other product. The variable cost is simply the value as specified on the product. The revenue is simply the market price multiplied by the estimated number of products that can be produced in the next time step. This estimate is calculated thusly:

```
100 * product.difficulty * skillLevel * crowdingFactor
```

Where crowdingFactor is:

```
Max( 0.05, 1.0 - (numberOfAgentsWorking / totalNumberOfParties ) )
```

Note that for crowdingFactor, the larger the number of agents working on the product, the smaller the value. Because the difficulty, skill level and crowding factors are all between zero and one, the number of products to be produced must be multiplied by a large number, in this case 100. Again, this number and indeed this whole function do not need mimic real life values for the purposes of this project. What is important is that these factors have an impact on the agent's decisions, and in a real application the calculation of the estimation of profit would need to be learned by the agent.

2) Each agent produces some products

This simply updates the money spent by the agent and the amount of the product that the agent has created, or does nothing if the agent is not working.

3) Each party purchases products from the agents

In the step the party goes through each product that it demands and looks for agents that have this product for sale. The party pays the agent money and the agent suppliers the party with the product.

4) The market price of each product is adjusted

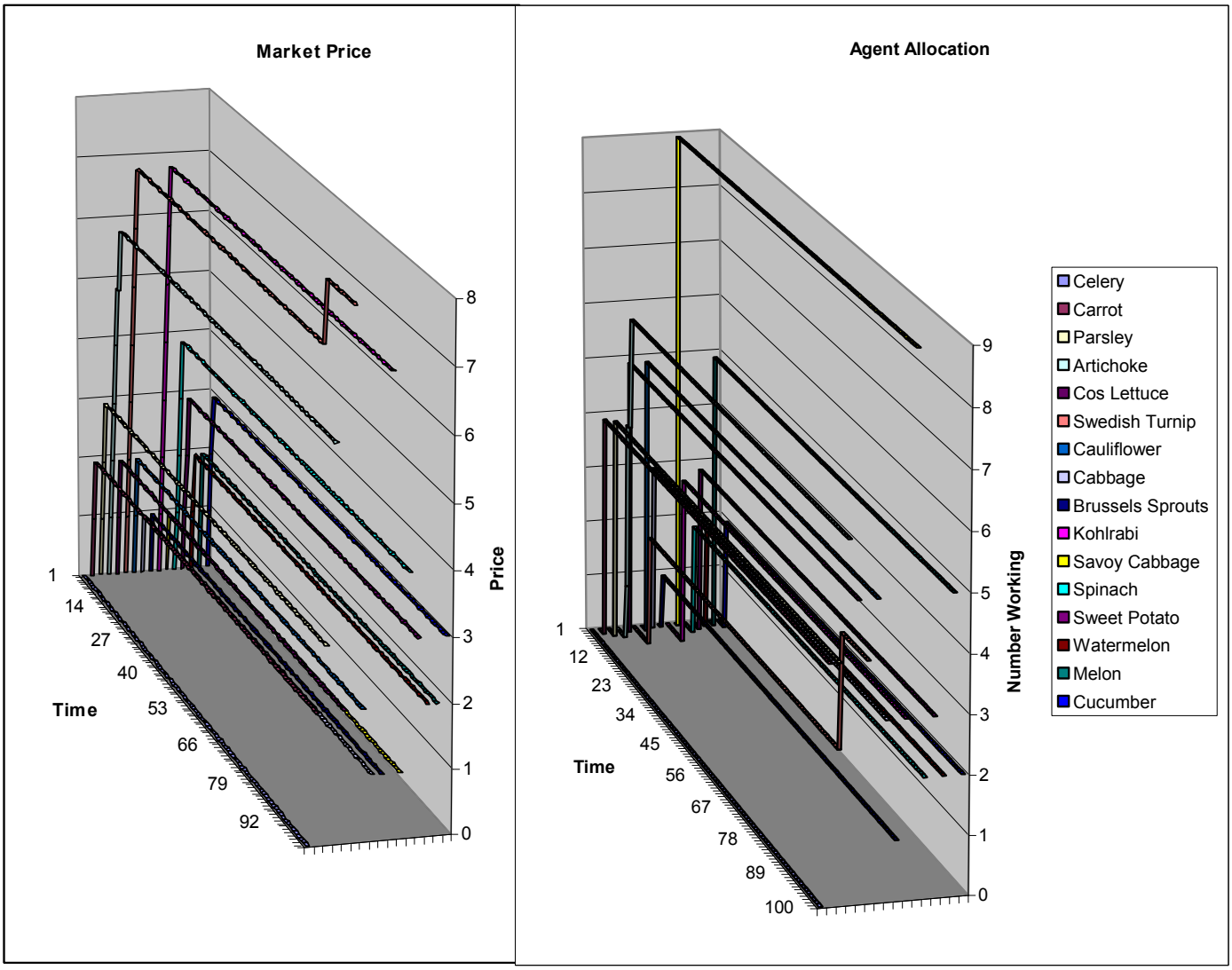
This is a crucial part of this project because this is what regulates the economy, i.e. this is what controls the allocation of the agents.
In the previous steps, the total amount of produced goods is recorded along with the total amount of goods demanded. If demand exceeded supply then the market price is increased by 1; if supply exceeded demand then the market price is decremented by 1; and there is no change if demand equalled supply.

The population of the farm was achieved with a combination of text files listing the products and agents to be included, and random values for certain values such as the skills of each agent in producing each product.

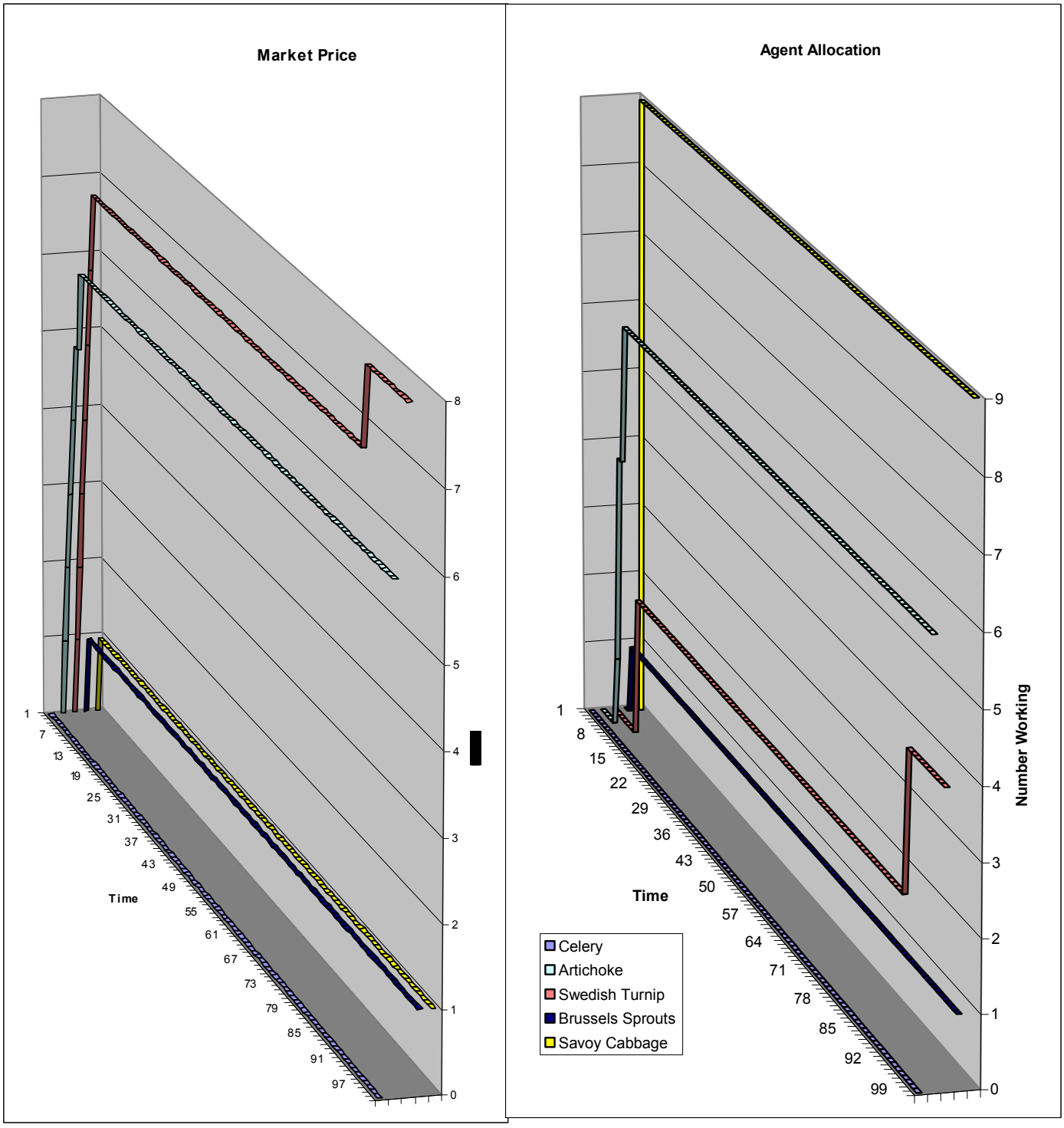
As you can see the implementation is very simple when you consider that these systems can effectively allocate thousands of agents to thousands of jobs even when different parties are competing with each other with the added bonus of fault tolerance. Just how effective the allocation is will be examined in the following section.

IV. Evaluation

The system was evaluated using a number of configurations in order to display different aspects of the system. The first evaluation is designed to show how the market becomes steady reasonably quickly. There were 16 parties with a total of 77 agents, with 16 products to produce. The following graphs show the market price of each product over time (left) and the allocation of agents (right). Note the speed in which the agents are allocated, and how steady that allocation remains when no unexpected events occur.

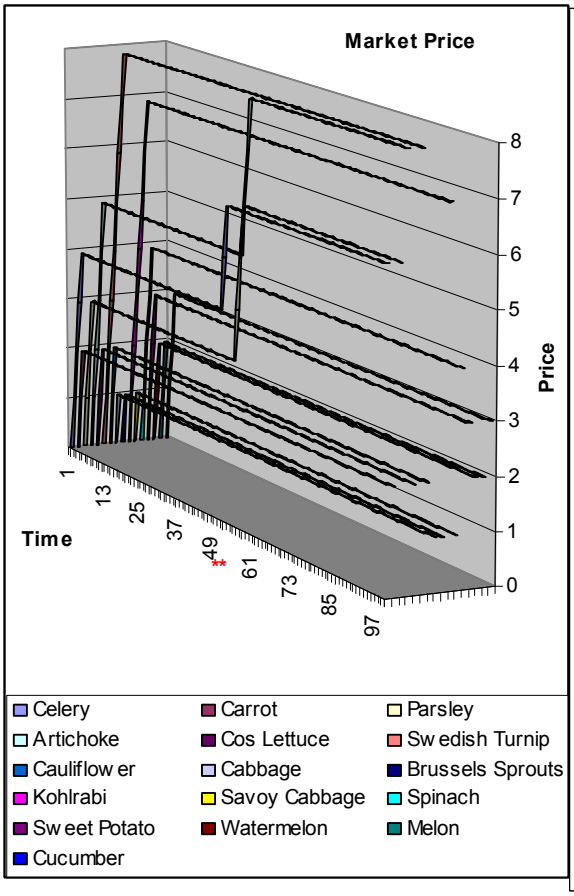


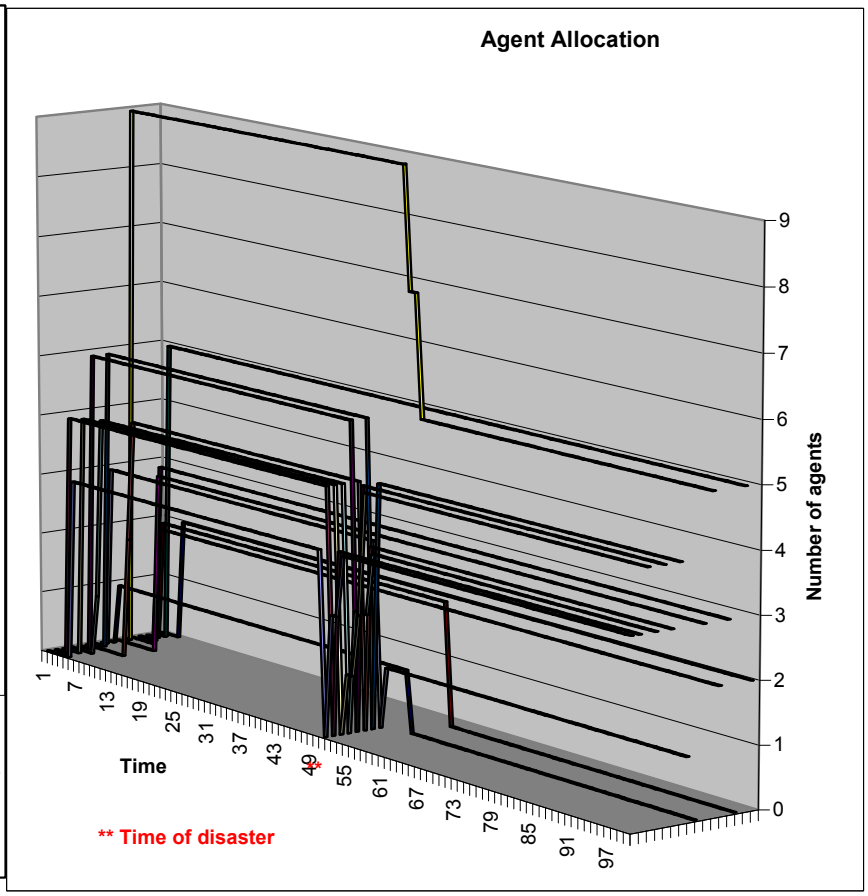
The graphs on the next page show a subset of the products shown above in order to show the relationship between market price and agent allocation. First look at the savoy cabbage (in yellow) and the Brussels sprouts (in dark blue). They both share the same market price, however nine times more agents are working on the cabbage. This is because cabbage is very easy to produce (in the simulation). In comparison, the Swedish turnip (in pink) had a very high market price but very few agents working on it. This is because it is very hard to produce, and the high price has occurred because there is more need for turnips than Brussels sprouts. The artichoke had high price and high allocation, showing that it is difficult to produce but there was very high demand for it, and there was no demand for celery and so none was made.



So the above graph on the right shows how agents are allocated in the given situation (i.e. many easy-to-make goods required, few difficult-to-make goods required, etc) and the graph on the left shows how the system achieved that allocation.

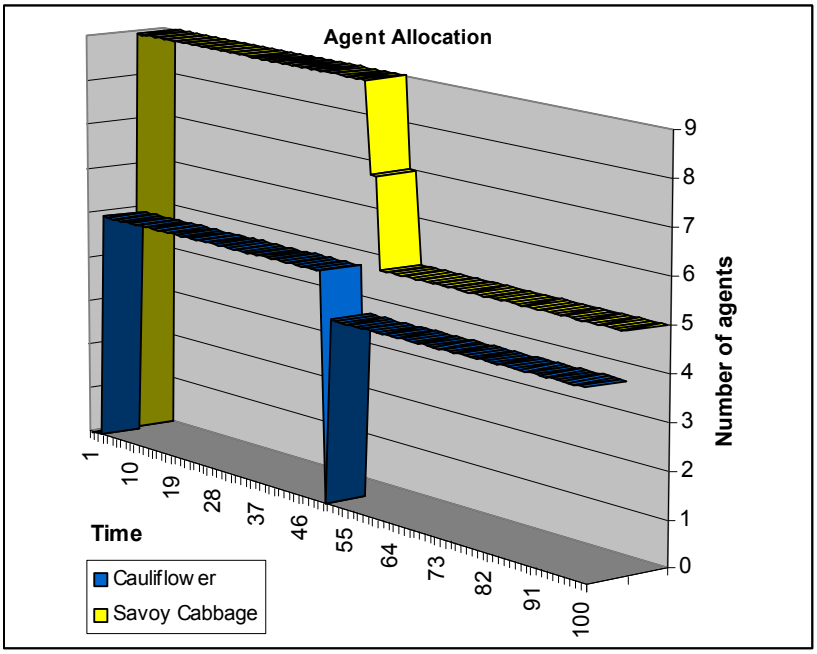
The next example illustrates the fault tolerance of the system. It simulates some disaster where the agents working on half of the products are destroyed. This occurs at time 50 out of 100 time steps.





Although it is difficult to clearly see what has happened, it is clear that immediately after the event the agents reallocated themselves, even though this behaviour was not explicitly programmed. You will note that for most of the products the price did not change because as soon as the destroyed agents disappeared, other agents realised the opportunity and immediately went to work on the affected products. The supply of the products they were previously working on decreased, which accounts for the and has increased market price in certain products.

The following graph looks at just two of the products to better illustrate how the allocation of agents changed, where all the agents producing cauliflower were destroyed: and



So the simulation achieved the goals of reaching a stable allocation which satisfied the parties in the simulation and being highly fault tolerant. Also note that because each agent makes its decisions individually, it would be highly scalable if each agent provided its own (even slow) processor. But does the division of labour improve performance over other systems?

Mathematically, assuming certain assumptions such as perfect knowledge of prices and products hold, as they do in this simulation, then clearly disabling the division of labour or disabling trading will lead to worse performance[3]. In an environment where the required assumptions do not hold, the performance of this system would be in question, and so the only fair way to say that this system was any better than another would be to compare them head-to-head.

[3] The proof of this is outside the scope of this report, but see any economics textbook to convince yourself of this.

However, it *is* possible to examine the performance of the system when changing the parameters that affect the division of labour. These results show where the division of labour would most benefit a system.

The system was such that every party was supplied with all the goods they needed, so the best way to study the performance of the system is to compare the cost in providing those goods when certain parameters are changed. Each result will also have a "career factor", which is the average number of different products each agent worked on, so the lower the career factor, the higher the division of labour.

The following table illustrates the results for a given typical configuration. You can ignore the actual meaning of the cost, using it only as an index to compare to the normal case.

Deviation from normal case	**Cost**	**Career factor**
None	31.64	1.27
No cross-party trade	33.20	2.49
No barriers to entry	29.83	4.25
Double barriers to entry	36.87	1.14
Double barriers to entry, no cross-party trade	44.04	1.81
No increase in skill	30.44	1.33
Double increase in skill	29.19	1.07
No variable costs	2.64	1.72
Double variable costs	68.90	1.36

Not surprisingly, when there is no cross party trade there is less specialisation because each party has to be self-sufficient. It is also clear that the barriers to entry have a big impact on the level of specialisation and cost, so in a multiagent system where barriers to entry do exist, it is vital that they are taken into account otherwise the system will not function efficiently (as is illustrated by the fact that on average each agent worked in over 4 different

areas when the barriers to entry were not considered compared to 1.14 areas when the barriers to entry were high).

As the level of increase in skill increases, the specialisation and hence efficiency also increases. So in any multiagent system where the agents can learn from experience, this increased skill should be taken into account when deciding what to work on. This would be cancelled out however in those systems where an increase in skill could instantly be shared with the other agents in the system, but would still be important where competing parties would not want to share their experiences.

Altering the variable cost had no visible trend as the division of labour increased and then decreased as variable cost increased. It would be expected that variable cost would have little effect on the division of labour, so the fact that with no variable costs the agents actually worked in more jobs seems surprising. However, this is an example of how in a market economy there are complex interdependencies that make the exact behaviour of the market difficult to predict. In this case, because there were no variable costs, more products were made more quickly so the agents had more time to work on the products that did not have such a high demand.

So while it was in no way proven that this system would perform better than other systems, it was found that that it was able to allocate many agents to satisfy many parties, it was fault tolerant, and we identified that in a system where barriers to entry or an increase in skill with experience exist, the system can be made more efficient by having specialisation. This system is an example of an easy and efficient way to achieve this specialisation.

V. Conclusions and Future Research

This project studied the allocation of agents in a multiagent system. While there are several ways of doing this, an effective allocation becomes difficult

as the number of agents increase, unexpected events such as faults in the agents or introduction of new resources occur, and when all agents cannot be controlled by a central program, such as when competitors are working within the same environment.

This problem was approached by using market-based principles. By introducing parties that demand goods and agents that can supply goods, the agents independently decide to produce the goods that bring them the most profit. Goods that are needed by many parties will have a high demand (and price) and hence more agents will produce them. On the other hand, if all the agents producing a certain good unexpectedly stop working then the supply of that good will be very low and so the market price will soar. Some agents will find it more profitable to start working on that good and so the system easily handles unexpected events and is self-regulating. All of this occurs without taking into account the global good of the system, and hence will work even when parties are competing against each other.

While this approach is not new (e.g. see [4]), the division of labour has often been ignored in market-based programming, despite its importance in markets. This project has looked at the benefits of the division of labour and the situations where the division of labour will bring the most benefit, namely when there are fixed costs involved in changing behaviour, and where there is improvement of efficiency from experience in working on a good. There are also other characteristics that promote the division of labour if they exist in a multiagent system, namely differing levels of skill between agents and the scarcity of resources, which were not examined in this project. Agents must also be able to at least approximately estimate the cost of producing each good, and how much of that good they are able to produce an order for them to make good decisions.

Future Work

A number of shortcuts were taken in this project, chiefly in populating the values in the simulations such as skill levels and costs of production. These were randomly generated, which was acceptable because the goal was not to accurately represent a farm, but rather to allocate resources taking into account these factors. When used as a real tool, these values must be accurate. The best way to determine these values would be for the system to learn them automatically, rather than having them manually entered. Therefore it would be important to use theories from machine learning concepts in order to determine the best way to learn these values so the agents could make well-informed decisions.

Another important aspect of market-oriented programming is the absence of any central control and information. In this project the market price was stored and decided centrally. A better method would be for each agent and party to keep their own price, and in order to find the selling price, they could query other agents. If this had been done in this project, the result would have been the same as keeping a global market price. This is because the price of a single type of product varies due to reasons such as the quality, the number of other competitors selling the product nearby, and the distance the product had to be shipped. None of these were relevant in the simulation, but where are any of these were present then the system would benefit if each agent kept its own preferred price and determined local market price by querying other agents. This would introduce further problems, such as trying to figure out if agents are telling the truth and how to determine market price based on what the other agents said. Bargaining, recommended retail prices and advertising may play a role in these systems also.

Agents could also stand to gain by being made more intelligent. This could include delaying purchasing when prices are falling, purchasing more before prices rise, and anticipating changes in the market in order to decide whether an agent should switch jobs or continue working as it has been.

There has been a trend to use ever more sophisticated economic theories in market-oriented programming, and applying theories relevant to the division of labour in order to increase the efficiency of a multiagent system and to solve problems that arise will be a continued effort in this field.

VI. References

[1] **Adaptive division of labor in large-scale minimalist multi-robot systems**, Jones, C.; Mataric, M.J., "Intelligent Robots and Systems", 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol.2, Iss., 27-31 Oct. 2003, Pages: 1969- 1974 vol.2

URL: ***http://ieeexplore.ieee.org.ezproxy.auckland.ac.nz/iel5/8832/27959/01248936.pdf?isnumber=27959&prod=STD&arnumber=1248936&arnumber=1248936&arSt=+1969&ared=+1974+vol.2&arAuthor=Jones%2C+C.%3B+Mataric%2C+M.J.***

[2] **A constitution and an economic model for the organisation and emergence of collective behaviour in a colony of robots**, Ali Cherif, A., (1994), Lab. d'Intelligence Artificielle, Univ. de Paris-Nord, St. Denis, France, This paper appears in: From Perception to Action Conference, 1994., Proceedings, pp: 334 - 337

[3] **Integrating economics and genetics models in information ecosystems**, Smith, R.E., Bonacine, C., Kearney, P., Eymann, T., (2000), Evolutionary Computation, 2000. Proceedings of the 2000 Congress on , Volume: 2 , 16-19 July 2000, pp:959 – 966

[4] **Using Virtual Markets to Program Global Behavior**, Mainland, G., Kang, L., Lahaie, S., Parkes, D. C., & Welsh, M., (2004), http://citeseer.ist.psu.edu/708474.html