An Investigation Using Computer Models Into the Relationships Between Task Uncertainty, Complexity and Organisation

Daniel Mack
MSc Computer Studies (Artificial Intelligence) 1990-91
University of Essex

ABSTRACT

This dissertation describes the design of the Organizations Test Program (OTP), a program which allows the simulation of simple hierarchical organisations using the methods of Distributed Artificial Intelligence (DAI). The author believes this is the first such use of simulation in the study of the theory of organisations. Some experiments using the OTP are described, and their results explained with reference to Galbraith's [1] theory of human organisations. Despite the limitations of the OTP, the results are found to be in accordance with Galbraith's predictions. The usefulness of DAI simulations is successfully demonstrated, and some concepts are introduced which may be of use in the development of the theory.

November 14, 1991
Acknowledgements

I would like to acknowledge the assistance of the following people:

Professor Jim Doran, my supervisor for the MSc, for his help throughout the project, and in particular for helping me refine the problem to be solved to a degree where it was practicable to achieve something.

Professor Nigel Gilbert, of the Department of Sociology at the University of Surrey, for supplying several versions of the NOOP object oriented Prolog programming environment.
An Investigation Using Computer Models Into the Relationships Between Task Uncertainty, Complexity and Organisation

Daniel Mack

MSc Computer Studies (Artificial Intelligence) 1990-91
University of Essex

1. Introduction

This dissertation describes the building of a computer program (the OTP) which enables the modelling of certain kinds of multi-agent organisation, and some experiments which were performed using the program. The results are related to existing theories of organisations, with particular emphasis on Galbraith [1].

Section 2 outlines the theoretical background to the research. Where appropriate, management science terminology is recast into words more accessible to computer scientists.

Section 3 relates the functionality of the OTP to the theoretical background.

Section 4 describes the major design features of the OTP, and the procedures adopted for testing it.

Section 5 describes the experiments so far performed, and presents their results.

Section 6 presents my conclusions, and briefly outlines directions for future research.
2. Background: Theories of Organisations

2.1. Introduction

A motivational discussion of reasons for studying the theory of organisations is presented. A brief summary of Galbraith’s theory is then given, followed by a discussion based on a comparison of Galbraith [1] with other theories of human and computer organisation [2, 3, 5, 13], and one previous attempt to build a model of such a theory [4]. Some comments are then made, concerning limitations inherent in the perspectives from which these theories were formulated.

I concentrate wholly on those sections of the theories which attempt to relate formal organisational structure to the nature of the task(s) performed by the organisation. As Galbraith’s view (and also Minzberg’s [21]) is that the informal relations within an organisation complement the formal structure, without causing significant distortion, this can be regarded as a useful abstraction for testing his theory.

2.2. Motivation

Why study the theory of organisations? No doubt every author has his/her own answer, but I believe that one sensible answer is to help us design and implement organisations which better satisfy our needs. There is vast scope in this definition for its politicisation: whose needs are to be met, for example, and what constitutes a need? An ideal organisation theory may describe how organisations can be created to best meet individuals’ own conceptions of their needs; but doubtless the same theory could be perverted into creating organisations that meet the needs of only a minority “ruling class”.¹

This is the position regarding theories of human organisations: how best can it be interpreted with regard to computer organisations? I believe that the main difference here is that we want computer organisations to best satisfy our needs (not theirs). The same applies to computers embedded within human organisations, a position which is becoming more common, and is summarised by the notion of Computer Supported Cooperative Work [18]. How are we to ensure that our needs are given fair consideration in the hybrid organisations of the future?

The need to answer this question, among many, provides a powerful motivation for the study of organisation theory, and, I hope, a useful (if distant) backdrop to the rest of the dissertation.

2.3. Galbraith’s Theory

Galbraith’s theory is that the optimal structure for an organisation is dependent on the chosen division of labour and on the uncertainty inherent in the task of the organisation. Increased uncertainty causes increased need for information to be passed around the organisation in order that decisions can be made as to how the organisation responds to the uncertainty. The extra information passing influences the optimal organisational structure because of the need to avoid overloading communications paths.

He suggests that small increases in uncertainty can be handled by reducing the span of control of managers, and by creating a staff specialist role, independent of line management.

In addition, he postulates that only five different mechanisms (organizing modes) are available to an organisation for coping with high levels of uncertainty:

- Actively changing the organisation’s operating environment
- Creation of "self-contained" subtasks
- Creation of lateral communication paths between organisation members
- Investment in vertical management information systems
- Creation of slack resources

Galbraith presents a detailed discussion of each of these mechanisms, which I shall not repeat here, other than to mention that the creation of slack resources is in some sense the default. Slack resources are additional resources required to handle uncertainty, such as additional man-hours of work. If none of the other organizing modes are adopted by management in response to increased uncertainty, then slack resources will be created (i.e. the job will take longer than expected).

¹ It should be noted that we are a long way from having a theory which is sufficiently useful to create either of these scenarios.
2.4. Discussion and Related Work

2.4.1. Related Work

A comparison of Galbraith with other organisation theorists [2, 5, 13] yields few differences. Mintzberg [2] extends the theory to deal with very small organisations, of the form referred to by Bourne [15] as organic. Williamson [13] and Malone [4, 5] are concerned with the economic conditions under which it becomes profitable to form a hierarchical organisation; this is the stage at which Galbraith’s theory becomes applicable.

Van Dyke Parunak [19] explores the same area as Williamson and Malone from the point of view of the contract net protocol [20].

Fox [3] provides a computer science oriented account of organisation theories, together with some criticisms: much of the following discussion is reliant on his work.

2.4.2. Division of Labour

All the theories [1, 2, 3] depend upon a notion of agents with cognitive limitations. The limitations of computers are well known, and are explained in complexity theory as bounds on the time taken and memory space required to execute an algorithm. Simon [6] presents psychological evidence for an essentially computational theory of human cognition, subject to the same limitations.

Fox [3] introduces a notion of the perceived complexity of a task as “excessive demands on rationality”. I interpret this to mean the degree to which the bounds of rationality of any agent are exceeded during performance of the task. In his view, the division of labour adopted by an organisation is directly related to the complexity of the overall organisational task: this is a point not discussed by Galbraith.

However, I do not feel that Fox makes fully explicit the connection with computational complexity theory. In fact, the choice of organisation influences which agents will have excessive demands placed on their rationality, and organisation itself is part of the (parallel) algorithm which has been chosen to perform the organisation’s overall task. Computational complexity depends on both the task to be performed, and the choice of algorithm. I suggest that it is the computational complexity which decides the optimal division of labour for a task, as it is a true measure of the resources required to perform the task.² It is not clear whether this notion of division of labour corresponds accurately with Galbraith’s.

2.4.3. Uncertainty

Galbraith intends uncertainty to relate to the commonsense notion of an inability to predict what will happen in the world. He defines the uncertainty of a task as being the difference between the information required to perform the task, and the information already available. He fails to distinguish between the information available to one individual within the organisation (and the uncertainty of the atomic task which that individual is required to perform), and the information available to the organisation as a whole (and its overall task). This is probably due to his intention to formulate a macro-level theory of organisations without detailed reference to micro-level processes.

Fox [3] provides arguments for a considerable expansion of Galbraith’s notion of uncertainty. He suggests that it can arise from within the organisation: people do not always behave as required; plans made by managers are not always good enough; and information received may not be fully believed. It is clear that correction of these unwanted circumstances will require extra resources from somewhere within the organisation; the uncertainty immediately creates a need for information concerning these additional resources. Thus it is possible to fit these sources of uncertainty into the Galbraith framework.

Van Dyke Parunak [19] follows Davis and Smith [20] in defining three kinds of ignorance present in contract net based “organisations”. These kinds of ignorance are clearly an information deficit of the kind described by Galbraith.

It is instructive to compare Galbraith’s view of uncertainty as equivalent to information deficit with the views of uncertainty adopted by other artificial intelligence researchers. An excellent summary of this work is presented in chapter 2 of Cohen [7].

AI researchers have concentrated on defining the uncertainty of a belief of an agent. This is not the same as the uncertainty associated with a task, but the two notions are related: task uncertainty can be regarded as uncertainty in the knowledge about the task of the agents who are to perform the task. Thus the provision of the missing information described by Galbraith can be regarded as increasing the certainty of the agent’s knowledge of the task.

² Note that this depends on the algorithms employed by individual agents: I will return to this point when describing the experiments performed.
There is an implied assumption in Galbraith's theory that uncertainty arises only from lack of knowledge: the lack is remedied by the supply of information to the agents. The assumption ignores the possibility that beliefs about the task may be held with different strengths of conviction by the agents (and, indeed, that the agents may harbour false beliefs). However, it is consistent with all views of uncertainty put forward by Cohen that additional information can reduce the uncertainty of an agent's beliefs, so it is not immediately apparent that this should be regarded as a shortcoming of Galbraith's theory.

The interesting issue here seems to be that different amounts of information may be required to make different beliefs equally certain. This throws doubt on Galbraith's notion that uncertainty is equivalent to the information required before the task can be performed. It suggests that Galbraith's theory might be restated more safely, so that it relates organisational structure to information deficit without making any claims about uncertainty.

It is, however, important to note that none of the artificial intelligence models of uncertainty are making strong claims about their degree of similarity to human reasoning processes. Cohen [7] presents some evidence that humans reason qualitatively (as well as quantitatively) about uncertainty; Kahneman et al. [8] present psychological evidence that human heuristics for handling uncertainty are not optimal. Given the current state of our knowledge of this area, Galbraith's model of uncertainty seems a reasonable first approximation.

2.4.4. Relationship Between Uncertainty and Division of Labour

We have seen that whichever model of uncertainty is adopted, its introduction into an organisation causes an increase in the amount of information passing, and an increase in the amount of decision making required before the task can be performed. A major consequence of uncertainty is an increase (at a macroscopic level) in the complexity of the overall task to be performed by the organisation. Returning to our earlier discussion of computational complexity: the algorithm for dealing with uncertainty is part of the overall algorithm for performing a task.

Thus we see that the algorithm for performing a task depends on the division of labour; this influences the uncertainty in the task, which itself affects the overall algorithm; and so on. The problem of finding the optimal organisational design is given a recursive definition.

This also leads us to consider the effectiveness of the communication protocol used between the members of the organisation. The contract net protocol [20] is an attempt to abstract those features of human communication most relevant to the formation of organisations. However, it concentrates on the establishment and dissolution of temporary contractual relationships between agents, without making any comment about communications once an organisational relationship is in place.

Van Dyke Parunak [19] goes a step towards considering a more permanent organisation within the contract net framework, by means of the strategy of audience restriction. In this strategy, offers of contracts are only broadcast to those agents who have previously responded to such offers: thus the broadcasting agent is beginning to build a representation of the organisation that has been created. This would appear to have much in common with the social processes that produce our commonsense view of an organisation as some fixed structure.

2.5. Limitations of Theories

The majority of the quoted research into organisation theory has been performed by management scientists or economists. This has led to the emphasis of certain features and forms of organisation, which I believe should be made explicit in any discussion of those theories:

- The majority of the work deals with companies, where an overall organisational task is well-defined, and the concepts of performance and efficiency are relatively clear (e.g. making as much money as possible for shareholders as fast as possible). This sidesteps the issue of how an organisational task "emerges" from the goals of its members and the constraints they impose on one another, which is perhaps more obviously significant in other kinds of organisation.4

- The work is aimed at giving practical advice for designers of existing types of organisation, rather than, perhaps, the most general theory achievable. In consequence, there is a tendency to leave unquestioned many "commonsense" assumptions about the nature of organisations and the relationships within them, which I feel ought to be addressed in a more general theory.5 This view is shared by Hewitt et al. in their review of organisation theories [18].

3 These are: Bayesian probabilities; Dempster-Shafer theory; possibilities (fuzzy set theory); and qualitative heuristics.

4 Though Williamson [13] views the "emergence" of a hierarchical organisation as a consequence of market failure, when price is unable to provide an adequate summary of the available information.

5 Lack of clarity over the meaning of an organisational goal is an instance of this effect.
3. The Organisations Test Program: Overview

3.1. Objectives
The aim of the research presented in this dissertation is to gather evidence for or against various aspects of Galbraith's theory. I have chosen to use simulation rather than, for example, the techniques of operational research (optimisation) for the following reasons:

- To date, simulation appears not to have been used significantly in the study of organisations.
- Simulation allows an intimacy of contact with the processes under study which cannot be achieved by other techniques.
- Simulation allows the exploration of problems which are extremely difficult to cast into a mathematical framework.

To this end, I have written a program called the Organisations Test Program (OTP).

Within the timescale of an MSc project there are, of course, severe restrictions on what can be accomplished, and I have limited the scale of my investigations accordingly. The sections below discuss the meaning of various concepts central to Galbraith's theory in the context of the limitations imposed by the OTP. This is followed by a summary of the major limitations and of the parts of Galbraith's theory on which I have concentrated.

3.2. The Organisational Task
Prosser [10] describes in his thesis a distributed AI system (DAS) for scheduling engineering jobs over machines, a task previously performed by humans. Such a task is also described by Van Dyke Parunak [19] in the context of the contract net protocol [20]. The organisations simulated by the OTP aim to perform a similar (but considerably simpler) scheduling task. This has the virtue of picking a task which has been tackled by both humans and machine agents previously: my conclusions about organisational structure can be compared with the structures adopted previously for this task.

3.3. Division of Labour
The task chosen for the OTP organisations is very simple compared with the tasks of the majority of human organisations. Labour is divided between the various agents in two ways:

- According to the machines that they control: different agents can control different numbers and types of machine, on which different types of job can be scheduled. Some machines can execute several different types of job: these are called multi-functional machines.
- According to their position in the organisation: the reporting hierarchy is a fixed part of the definition of the organisation. All agents at the base of the hierarchy control machines and actually perform the insertion of jobs into schedules (these are called tactical agents). All agents above the base of the hierarchy simply delegate scheduling requests to their subordinates (these are called strategic agents), spending time deciding which subordinate to delegate each job to.

3.4. Uncertainty
There are two sources of uncertainty in the definition of the task performed by the OTP:

- Uncertainty in the parameters of the jobs submitted. The agents do not know in advance the rate at which jobs will be submitted; the sizes and deadlines of jobs submitted; or the distribution of the different types of jobs submitted. Were this information available, a plan could be made which ensured the highest possible level of performance.
- Uncertainty in the availability of machines on which to schedule jobs. Communications with machines can break down, in which case it becomes impossible to schedule any more jobs on the machine until they are repaired. If information about machine availability was available to the organisation in advance, it could also be used to make a plan which ensured a higher level of performance.

In later experiments, attempts are made by the simulated organisations to gather some of this information. In particular, information is gathered about the following:

- The rate of arrival of requests to schedule jobs of different types.
- The number of scheduling requests which have been sent to each agent but not yet processed.

These sources of information can be used to reduce uncertainty in the distribution of the different job types submitted, and the availability of machines, respectively.
3.5. Performance

Galbraith's theory [1] describes a relationship between task uncertainty, organisational structure, and performance, without clarifying how performance is to be measured. Fox [3] follows the work of Williamson [13], in suggesting that performance should be measured in terms of the costs and benefits accruing from the execution of primitive transactions by members of the organisation. Transactions are the primitive actions into which the organisational task has been analysed in order to divide them amongst the available workforce, together with those additional primitive actions which are made necessary to coordinate the organisation.

In the context of the OTP, each transaction is represented by a method execution. Thus, in theory, we should assign a cost and a benefit to each method execution (or possibly to each execution path through a method), count the number of executions of each path during an experiment, and then calculate the overall benefit to the organisation. Performance is here equated with achieving the maximum possible benefit.

This approach does, of course, have the same problems as any "price" models of cost and benefit: it is unclear exactly what the costs and benefits of any given transaction should be. In absolute terms, this is irrelevant, but in order for the measure of performance to be credible, the relative costs and benefits of transactions must be meaningful.

I have adopted a model of performance where the only benefit to the organisation results from the reporting to the exception handler (part of the environment) of a successfully scheduled job. It can reasonably be argued that the reporting of a scheduling failure might accrue some benefit to the organisation (as at least it might enable the organisation to compensate for its inability to schedule a particular job, whereas if a failure to schedule is not reported, no compensation can take place). However, it is not clear what weighting should be given to this benefit in relation to the benefit of a successfully scheduled job. I have assumed it is small enough to be insignificant.

This leaves the costs of the various methods to be decided. One possibility is that the cost of a method should be proportional to the amount of time spent executing the method. This also applies to idle time: the cost of supporting an idle agent for any given period is the same as the cost of supporting an active agent for that time. This has the following consequences:

- Any comparison between experiments on the same organisation over identical time intervals can be made independently of the costs incurred in the experiment.
- This point also applies to comparisons between two organisations over the same interval provided that the organisations have the same number of agents.
- There is a need for all other comparisons to decide on the relative importance of the benefits compared with the costs.

This final point is of extreme significance, as by suitable manipulation of the weightings assigned to costs and benefits, we can make virtually any organisation appear to have the "best" performance.

For this reason, I have decided to simplify my results collection by ignoring costs altogether, which is equivalent to giving them a weighting of zero relative to the benefits. This can be likened to the situation which exists in a voluntary organisation, such as a charity or political party, where the members give their time freely.

However, this does not mean that the results are completely irrelevant to other forms of organisation, as the addition of costs will not affect the qualitative behaviour of different organisations, only the quantitative summary of their behaviour. Thus the types of changes in behaviour as uncertainty is increased will be of significance, even though my experiments will be able to say nothing about the actual values of uncertainty at which one organisation becomes more effective than another.6

3.6. Limitations of the OTP

There are a number of limitations in the design of the OTP, some of which influence the parts of Galbraith's theory that can be explored:

- The OTP reifies7 the communications paths between members of an organisation. They are rigidly defined at the start of an experiment, and cannot be altered. Furthermore, the relations between agents are forced to be hierarchical.
- Likewise, the primitive transactions into which the organisational task is divided are reified. There is no ability for agents to negotiate the definitions of either the primitive transactions or of the organisational task itself. Thus the OTP is unable

---

6 Clearly, this is the case even without consideration of the effects of costs. As there is no absolute measure of uncertainty, I am already using an arbitrary scale for measuring results.

7 Reification is a term commonly used by sociologists to describe a type of conceptualisation where processes are regarded as if they were objects. Thus the process of interaction between the members of an organisation is reified to a structure, which is conceived of as fixed, and we experience as difficult to change. It is argued that only by deconstructing the reified objects into their originating processes can we ever hope to gain real understanding of the social world.
to explore the recursive nature of division of labour.

- Agents' knowledge is not explicitly represented, and the model of inference (decision making) is naive (all decisions are hard-coded as rules). It is difficult to use OTP to relate Galbraith's view of uncertainty to the uncertainty of agents' knowledge.

- The organisational task is very simple: there is less scope for exploration of aspects of Galbraith's theory such as creation of a line-staff organisation, or creation of self contained tasks.

In practice, time limitations have further restricted my experimentation to the following organisational features:

- Span of managerial control
- Height of organisation
- Decision making algorithm
- Delegation of decision making
- Creation of lateral communications paths
4. The Organisations Test Program: High Level Design

4.1. Introduction

The following sections give an overview of the design of the OTP; the details are presented in Appendix A. The OTP is an object-oriented program [9, 14], based upon the existing NOOP package. An agent is modelled as an object whose available skills are its methods. This follows the lead of DAI testbeds such as MACE [11]; it allows for the flexible creation of different numbers and types of agents, for the relatively easy packaging and transfer of skills between agents, and for the re-use of code defining existing skills in order to build more complex skill packages for new agents. A method is an indivisible unit of work, and as such can be regarded as a primitive transaction [13, 3] for the purpose of performance measurement.

In the remainder of this section, I first discuss the nature of some modifications to the NOOP package which I found necessary. I then discuss the types of agent available, and sketch the passage of a scheduling request through the program. I present the detail of three different delegation algorithms used, discuss the use of lateral communications between agents, and comment on the use of different priorities for message sending. Finally, I discuss the modularisation and testing of the program.

4.2. Modifications to NOOP

4.2.1. Inadequacies of Existing NOOP Implementation

As provided, the NOOP environment did not allow adequate control of how execution time was allocated to objects. The reason for this was the technique used for method invocation: when a message was passed to an object, it resulted in an immediate direct procedure call to the method. Whereas this is the simplest technique for implementing message passing between objects, it suffers from several problems:

- The run-time call stack can grow indefinitely. This occurs when methods send messages which invoke methods which send messages, recursively.
- It is impossible to separate the order in which methods execute from the order in which they are requested by the application to execute. Thus, due to application design, one object may get three of its methods run in a row, say, before any other agent gets a look in. This is not a problem for all object-oriented systems, but it makes it impossible to establish any notion of simulated time, as method executions are not allocated fairly between agents.
- As it is impossible to set up a scale of simulated time, it is impossible to vary the relative times taken to execute different methods, except by changing the code.

4.2.2. New NOOP Implementation

All the above problems can be alleviated by changing the technique used for method invocation, which has been done in my new implementation.

A message queue is maintained for each object. When a message is sent to an object, all that happens at that moment is that the message is added to the object’s message queue.

A message scheduler is responsible for deciding what methods will run on what objects at what time. In order to provide a framework for simulating the passing of time, each object in turn is given an opportunity to execute one method. The method to be executed is that corresponding to the message at the front of the message queue for that object: the message is removed from the queue immediately prior to execution.

Simulated time can now be measured in scheduler cycles, where a scheduler cycle is the time taken to offer every object a chance to execute one method.

This basic framework leaves room for three small enhancements, which I have found useful during my experimentation:

- The ability to send a high priority message: this will be placed at the front of the receiver’s message queue, and will therefore be executed in the next scheduler cycle.\[10\]

---

8 The NOOP object-oriented Prolog programming environment was used by kind permission of Professor Nigel Gilbert, Department of Sociology, University of Surrey, Guildford.
9 The discussion in this section presupposes some familiarity with the concepts of object-oriented programming as presented in Broch [14] or Meyer [9].
10 Many more possibilities for experimentation are opened up by assigning different priorities to different messages: more will be said about this issue later.
• The ability for an object to spread repeated executions of a method over several machine cycles without interruption by other agents’ messages. This is used where an object may have to repeat the same activity several times in a row with slightly different parameters, and it is felt that one cycle of time should be allocated to each repeat.

• The ability to alter the simulated time taken by a message execution, without changing the code: this is implemented by associating a configured execution time with each method. The message scheduler counts down the cycles until the right number has passed, then executes the method in the last cycle of the count.

4.3. Types of Agent

Four kinds of agent have been used in the experiments presented:

• An environment agent, responsible for introducing new job scheduling requests into the system, and for generating some kinds of environmental uncertainty.

• A strategic agent, responsible for accepting scheduling requests and delegating them to a suitable inferior (either strategic or tactical) agent. This agent has a role similar to that of a manager in a human organisation.

• A tactical agent, responsible for accepting job scheduling requests and for assigning each job to a specific machine under its control and an execution time. This agent has a role similar to that of a "worker" in a human organisation.

• An exception handling agent, responsible for recording all reports of successes and failures made by the organisation as a whole (and thus for monitoring performance). This agent’s task has been kept distinct from that of the environment agent in order to prevent interference between the two tasks: notionally they both correspond to an interface with the users (customers, etc.) of the organisation.

4.4. Operation of the OTP

The contents of this section are summarised in Figure 1.11 The top strategic agent receives job scheduling requests from the environment agent. Several requests may be submitted in one scheduler cycle; such a group will be described below as a batch of requests.

A job is represented within the OTP as a Prolog term (see [12]) with four parameters:

• A unique job identifier.
• A job type, which controls which machines can execute it.
• An anticipated execution time.
• A deadline, by which time its execution must be complete.

Variability is introduced by the environment agent into the parameters of the submitted jobs. This is one mechanism used to generate task uncertainty for the organisation.

Essentially, scheduling requests are always delegated by strategic agents according to the job type (which indicates the type of machine it can be run on), and using one of three different algorithms for deciding which of the tactical agents supporting that machine type to delegate to. The OTP supports a multifunctional machine type, on which any type of job can run.

Where a tactical agent supports more than one machine, a round-robin scheme is used to decide which machine to attempt to schedule the job on.

The environment agent is also capable of directing random machine failures, to provide another source of uncertainty. The rate at which these failures (and recoveries) take place is determined by parameters supplied to the environment agent.

4.5. Strategic Agent Delegation Algorithms

I have defined three different algorithms which a strategic agent may use to decide which subordinate to delegate a scheduling request to. They are referred to as the Naive, Improved, and Adaptive13 algorithms in the rest of the text.

---

11 In later experiments, tactical agents communicate directly via lateral links: this feature is not included in the diagram in Figure 1.

13 Another obvious choice of algorithm is always delegating a scheduling request to the least busy subordinate. This shares with the adaptive algorithm the property of being responsive to the actual jobs submitted, but no additional message passing between agents is required. For this reason, in combination with the time constraints of an MSc project, I decided against implementing it.
Figure 1.

Overview of the Organisations Test Program

**Environment Agent**
Submits batches of jobs for scheduling to the organisation. Introduces uncertainty into some of the job parameters.

**Exception Handling Agent**
Receives reports of all successes and failures at scheduling jobs. Maintains count of successes and failures.

**Top Strategic Agent**
Head of the organisation. Delegates all scheduling requests to its subordinates. Decides on most appropriate subordinate for each job.

**Strategic Agent**
Middle manager. Delegates all scheduling requests to its subordinates. Decides on most appropriate subordinate for each job.

**Tactical Agent**
Worker. Controls machines. Maintains a schedule for each machine under its control. Reports successes and failures to its boss.

**Key**
- Path of successful scheduling request
- Report of scheduling failure by tactical agent to its immediate manager; manager response to alternative tactical agent
- Report of scheduling failure by strategic agent to its immediate manager; manager response to alternative subordinate strategic agent
- Report of complete scheduling failure to outside world
4.5.1. Naive Algorithm

For the naive algorithm, the strategic agent has a list for each possible job type of all the subordinates which can handle that job type. When a new scheduling request is received, it is always passed to the first subordinate in the list for that job type. If this subordinate fails to schedule the job, then it is passed to the second subordinate on the list, and so on, until all subordinates in the list have been tried.

The algorithm is called naive because it takes no account of the loading of the individual subordinates. Invariably, the first subordinate on each list becomes overloaded, but the last one may well never succeed in scheduling a job, as it has to wait so long for anything to be passed to it.

4.5.2. Improved Algorithm

The improved algorithm goes some way towards balancing the load between all the subordinates for each job type. It does this by rotating among all subordinates on the list which will be the first agent to receive a scheduling request for a job of the type in question. Thus, if subordinates 1, 2, and 3 can all handle job type 1, then the first job of this type to be received will be passed to subordinate 1, the second to subordinate 2, and the third to subordinate 3. The fourth will start by being passed to subordinate 1 again.

If the first choice of subordinate fails to schedule a job, the algorithm proceeds exactly as for the naive algorithm, except that it rotates past the end of the list to the beginning again, and stops when all subordinates in the list have been tried, not when the end of the list is reached. Retries have no effect on the rotation of duty between subordinates for which will be first to receive a job of a particular type.

4.5.3. Adaptive Algorithm

The adaptive algorithm works like the improved algorithm, except that the strategic agent is allowed to specialise or generalise the functions of its subordinates, depending on the frequencies with which jobs of the different types arrive.

A frequency check cycle is defined to be the time taken for the strategic agent to receive a number of scheduling requests equal to the number of different job types which it can handle. During each frequency check cycle, the strategic agent counts how often scheduling requests of each type are received. If two requests of the same type are received in one cycle, then the agent looks for a subordinate to generalise, so as to increase capacity for that job type. If no requests at all are received for a particular type within a complete frequency check cycle, then the agent looks for a subordinate to specialise, by removing its ability to handle jobs of that type.

The strategic agent maintains a list of job types currently enabled and those which could be enabled, but are not so currently, for each subordinate. Specialisation implies disabling a job type for a machine; generalisation is enabling a currently disabled job type. Obviously if all the subordinates have all machine types enabled, then no more generalisation can take place, and similarly for specialisation if all subordinates have only one job type enabled. When a subordinate is located to specialise or generalise, it is sent a message requesting the required change. A reply to the message is expected before the strategic agent adapts the job types it sends to the subordinate: however, the methods are written so the reply is sent at high priority, in order to make the response as fast as possible.

4.6. Lateral Communications Between Agents

In experiments 8 and 9, tactical agents are allowed to communicate directly with one another, rather than only via their superior strategic agent. This section explains the circumstances under which lateral communications occur, and the information that is communicated.

The intention is that the lateral links will be used to reduce the volume of communications between the strategic agent and its subordinates. Thus, when a subordinate is unable to schedule a job that it has been delegated, instead of passing a failure message back to its boss, it offers the job directly to another tactical agent for scheduling. Currently, it offers the job to the agent which it "believes" to have the smallest number of jobs waiting to be processed, i.e. the least loaded agent.

The agent which is offered the job may accept or refuse it, depending on its actual loading. If its loading is less than the original agent's estimate of the average loading of all the tactical agents, then it accepts the job; otherwise it rejects it.

A rejected job is passed back to the sender, with an update of the loading of the agent which rejected it. The sender must now attempt to find another agent to which to transfer the job.

\[^{14}\text{A subordinate cannot be "specialised" so that it handles no job types at all.}\]
An accepted job is queued just like a scheduling request from the strategic agent would be, and handled identically. There is no guarantee that because an agent has accepted a job it will be successful in its attempt to schedule it.

It is interesting to note that the offering of a job to another agent depends on a "belief" about loading, not accurate knowledge of the loading of other agents. This belief may often be incorrect, although it will provide a reasonable approximation. This situation is analogous to the situation in human organisations, where individuals will not necessarily have a particularly good idea of the workload of their colleagues.

4.7. The Assignment of Priorities to Messages

The ability to assign a priority to a message is one not considered by Galbraith, but which is provided by the OTP. During my experimentation, I made use of this feature with caution, but found that in some cases it was necessary to use it in order to produce meaningful results from experiments.

There were two places where high priority messages were sent (all the others were low priority):

- Where a tactical agent offers a job to another tactical agent via a lateral link, the offer is high priority. Any rejection of the offer is also given high priority. In my judgement, the mechanism could never generate performance improvements if requests to take over scheduling responsibility were queued up. The argument for using priority messages is that in a human organisation, any discussion over such a transfer of work based on individual workloads would be relatively short in duration, and would probably occur informally, alongside any actual scheduling occurring at that time.

- Where a tactical agent accepts an order from its boss to specialise or generalise the job types it will handle. The argument for using a high priority reply is similar to that used for lateral connections.

4.8. Modularisation and Testing

The OTP is divided into five modules as follows:

- Entry point (and exit handler)
- Initialisation (reading configuration file)
- Message Scheduler (and NOOP)
- Methods
- Statistics handling

The modular structure has been chosen to minimise the number of interfaces between modules, to facilitate some data hiding, and to ease the process of testing. In all cases, much of the testing was performed with the aid of the Prolog debugger; this enables the performance of structural testing, where the values of variables and the flow of control can be checked against expectations.

In the following section, I discuss the functional level tests applied to each module in turn: these tests ignore the internal operations of routines and modules, concentrating instead on the final output. In general a bottom-up testing strategy was adopted, and the order of the descriptions follows this procedure.

4.8.1. Message Scheduler

Some extremely simple methods were written, which, when invoked, would write out their names and the name of the object which was running them. These methods were then registered with NOOP for five different objects. The method-invoking messages for these methods were then sent using the new message-sending interface. The message scheduler was then invoked, and the output from the methods compared with that expected (both for order and content). This test was repeated for high priority messages.

A pair of methods were then written so that one sent a message to another object with the name of the sender as data, and the other displayed the name received as data at its invocation. These methods were assigned to a pair of NOOP objects, and a message sent to the first. The message scheduler was started, and the output observed.

Similar tests were performed to check that high priority messages really received priority over low priority ones, and to test the ability of an object to send a continuation message for a method to itself without interruption.

4.8.2. Statistics Handling

This module consists of two parts: a set of routines for amending the values of statistics stored by individual agents, and routines for writing statistics from an agent to the statistics file.
As each of the individual routines in this module was quite simple, testing was also relatively easy. An object was created, using NOOP, with a number of named slots. Initial values were assigned to the slots using NOOP, then the routines to write statistics to the file were used to write the values of these slots. The statistics modification routines were then used to alter the values in the slots, and the values were again written to the file. This was repeated several times, and the values in the file were compared with the expected values.

4.8.3. Initialisation
This module consists of routines to read the configuration file and routines to create the required objects. The creation of objects was tested first, by calling the routines directly, and examining the NOOP database immediately afterwards. The reading of the configuration file was then tested by submitting configuration files with a wide variety of parameters in each allowed clause type, and examining the NOOP database immediately afterwards.

4.8.4. Entry Point and Methods
These could only be tested in conjunction with each other and the rest of the software. The working of the entry point code was verified simply by the fact that the other components were correctly invoked. The testing of the methods could only be done by comparing their behaviour, step by step, with the detailed design (see appendix A). This always required use of the Prolog debugger.
5. Experiments and Results

5.1. Experimental Strategy

My experimental strategy was as follows:

- Measure and explain the performance characteristics of some very simple organisations under increasing load, with no uncertainty present.
- Introduce variability into some parameters, thus generating uncertainty, and explain the changes in performance, relating them to Galbraith's theory.
- Repeat the same experiments on several different organisations with the same division of labour.
- Repeat the same experiments for different sources of load and uncertainty.
- Explore the effects of changes in the algorithm for particular subtasks (transactions).
- Follow Galbraith's suggestions for organizing males, by creating different organisational structures. Repeat the experiments on these new organisations and see if their performance under uncertainty improves as expected.

5.1.1. Sources of Load

I will consider two separate kinds of load on the organisation:

- Anticipated job execution time as a proportion of time until job deadline
- Rate at which jobs are submitted for scheduling

In both cases loading takes place by increasing the rate at which messages are sent to agents within the organisation: in the first case by increasing the number of scheduling failures and retries; in the second by increasing the rate at which scheduling requests are sent to the top level strategic agent. An agent can only process one message per scheduler cycle: therefore if messages arrive faster than this, the agent's message queue increases in length, and the agent is said to be overloaded.\(^\text{15}\)

5.1.2. Sources of Uncertainty

Four sources of uncertainty are used in these experiments:

- Each job is assigned an anticipated job execution time falling between a minimum and maximum supplied to the environment agent as parameters. The probability distribution of values between the limits is rectangular.
- Every time a batch of jobs has been submitted, the time delay until the next batch is decided. The value chosen for the time delay will lie between a minimum and maximum supplied to the environment agent as parameters. The probability distribution of values between the limits is once again rectangular.
- Each job is also assigned a job type, which controls which machine it can be executed on. The job type is one of a range of types supplied to the environment agent as parameters: the number of types supplied can be varied by the experimenter. The probability distribution of selected job types is rectangular.
- A percentage probability of machines breaking down or recovering within any given OTP scheduler cycle. These two percentage probabilities are supplied as parameters to the environment agent.

Continuous uncertainty by the agents about the values of these parameters follows from their variability. This follows from Galbraith's view of uncertainty by considering the following. The task of the organisation is to schedule (and report scheduled) as many of the submitted jobs as possible within a fixed time period. Let us consider, as an example, the state of an agent's knowledge about the relative numbers of different job types among the jobs submitted.

Initially, the agent knows nothing about the relative numbers of different job types. As jobs are submitted to the organisation, it can acquire this information within a short time period, provided the numbers remain constant. It may be able to use this information to better control which parts of the organisation are used to schedule which job types. The information will have been gathered once, and there will be no remaining uncertainty due to this factor.

However, if the relative numbers of the different job types are variable, it can never reduce its uncertainty to zero in this way. Even if it continually gathered information about the job types submitted, it could never optimise organisational performance once...

\(^{15}\) It would have been quite possible to limit message queues to finite size, as well, with additional messages being "lost". This would correspond to constraints on space required to execute the algorithm, whereas I have only modelled constraints on time. However, I felt that one form of cognitive limitation was sufficient to achieve the correct qualitative behaviour.
and for all.

5.1.3. Algorithm Changes

The algorithms used are the naive, improved, and adaptive algorithms, described above.

Experiments 1, 2, and 3 use the naive algorithm; experiments 4 and 5 the improved algorithm; experiment 6 both the adaptive and the improved algorithms; experiment 7 the adaptive algorithm; and experiments 8 and 9 the improved algorithm combined with lateral links between tactical agents.

5.2. Experimental Method

The OTP reads definitions of agents and the object classes upon which they are based from a configuration file. This enables easy specification of different organisations, and different parameters (for example, those controlling variability in the properties of jobs submitted for scheduling). A numbered configuration file was set up for each organisation and parameter setting used for the experimentation.

I chose to run all experiments for 500 scheduler cycles; this seemed to be long enough to eliminate any undesirable effects caused by the initial boundary conditions, but short enough in terms of real execution time to be practicable.

During its run, the OTP writes a large number of statistics to disk every 5 cycles: the statistics file is numbered following the configuration file with which it is associated. The statistics record many facets of the operation of the organisation, for example, the numbers of messages received, sent, and currently being processed by each agent; the number of jobs successfully scheduled and the number of scheduling failures to date by each agent. All the graphs presented below rely on the number of jobs reported to the exception handling agent as successfully scheduled: this is also written to the statistics file every 5 cycles.

After running the OTP, the statistics files generated were analysed. Where patterns were detected relating performance to some other variable, the additional statistics provided much of the evidence required to produce an explanation for the pattern observed.

Where the configuration involved variability in one or more parameters, more than one run was performed on that configuration file. Care was taken to preserve the statistics files for all of the OTP runs on these configurations. Depending on the magnitude of the variability, I have performed up to ten runs of a single configuration file, calculating the mean and standard deviation for the performance figures.

5.3. Experiments and Results

5.3.1. Experiment 1: Effect of Varying Anticipated Job Execution Time

Figure 2 demonstrates the main features of the variation in performance of a simple organisation as anticipated job execution time is varied. I have chosen the simplest possible organisation that is recognisably a hierarchy for this experiment, so that other results can be compared with it quite easily. Performance tails off quite steeply as anticipated job execution time increases: initially, the rate of decrease in performance itself decreases, but as the job length approaches the maximum time allowed for the job (the deadline), the rate of decrease becomes larger once again.

The explanation for these variations in the rate of decrease in performance lies in the time taken to pass a job on to the second machine for an attempt at scheduling (if the attempt on the first machine fails). Initially, with job lengths small, most jobs can be scheduled on the first machine. As job length increases, some of the jobs are passed on to the second machine and successfully scheduled there (which explains the flattening of the curve in its middle portion). However, as job length approaches the deadline it becomes impossible first to schedule any job rejected by the first machine on the second (the rejection process took too long); and finally (at the deadline) to schedule any job at all. The steepening of the curve again is the result of inability to schedule rejected jobs on the second machine.

The introduction of variability in anticipated job execution time (and its consequent uncertainty) has two consequences (see figure 3):

- As uncertainty increases, the performance of the organisation increases. This can be explained by supposing that the increased performance due to jobs shorter than the average anticipated job execution time outweighs the decrease in performance due to jobs longer than the average. This ties in with the early decrease in steepness shown in Figure 2, but an alternative explanation is required closer to the deadline. The effect here can be explained by the fact that the degradation of performance due to jobs of any length greater than the deadline will be identical, whereas the improvements in performance associated with jobs less than the deadline in length will not.
Figure 2.

Key to Organisation Diagram:
- Strategic (Management) Agent
- Tactical (Worker) Agent
- Machine Type 1

Jobs Scheduled and Reported in 500 cycles vs. Anticipated Job Execution Time
(Job Deadline fixed at 11 cycles after submission)

Increase in Performance due to Variability of plus/minus 3 in Anticipated Job Execution Time