Constructing an animat mind using 505 sub-minds from 234 different authors

Ciarán O'Leary* **Mark Humphrys**** **Ray Walshe****

*School of Computing
Dublin Institute of Technology
Kevin St.
Dublin 8
Ireland
www.comp.dit.ie/coleary

**School of Computing
Dublin City University
Glasnevin
Dublin 9
Ireland
computing.dcu.ie/~humphrys
computing.dcu.ie/~ray

Abstract

The World-Wide-Mind (WWM) is a scheme for putting minds (by which we mean the software to drive a real or virtual agent) online for remote re-use by third parties. Although the animat community favours the extension of existing agent minds in an ongoing process, there is currently no easy way to re-use minds built by other authors. If re-use became the norm, it is suggested that larger, more diverse and more complex minds could be built.

This paper describes an ongoing project that uses WWM technology to develop a novel set of minds that incorporate a range of algorithms and techniques. We demonstrate that it is possible to evolve superior minds by the artificial selection and combination of existing online minds.

The project that is described here involved 234 authors who developed 505 different minds for the virtual animal in a re-implementation of Tyrrell's Simulated Environment (Tyrrell, 1993). These minds, which were developed by two classes of undergraduate Computer Science students, were subsequently selected based on their performance in the virtual world, and integrated into larger minds. At the time of writing the most successful mind developed used as components the two sub-minds that most successfully satisfied the animal's two main goals. Since a huge number of combinations of minds is possible, it is important that the work is distributed among a community of researchers. The architecture of the World-Wide-Mind makes this possible.

1. Introduction

The World-Wide-Mind (WWM) (Humphrys and O'Leary, 2002) is a scheme for putting animat minds online for remote re-use by third parties. Although the animat community favours the extension of existing agent minds in an ongoing process (Wilson, 1990), there is currently no easy way to re-use minds built by other authors (Guillot and Meyer, 2000). If re-use became the norm, it is suggested that larger, more diverse and more complex minds could be built.

This paper describes an ongoing project that uses WWM technology to develop a novel set of minds that incorporate a range of algorithms and techniques. We demonstrate that it is possible to evolve superior minds by the artificial selection and combination of existing minds written by diverse authors, in a way that would not previously have been possible.

1.1 Project Description

The ongoing project that is described here has so far involved 234 authors who each developed minds (or *action selection mechanisms*) for the virtual animal in a re-implementation of Tyrrell's Simulated Environment (Tyrrell, 1993). Each author (the majority of whom were undergraduate Computer Science students) is permitted to develop any number of minds, which are then made available online, as web services. Since the minds are online, third parties can select a specific mind and run it in the virtual world, located at w2m.comp.dit.ie. This effectively means plugging the mind into the animal in the virtual world, and letting it select the actions at each point in time. The score that is achieved by each mind in each run is recorded in an online scoreboard that can be viewed by all other authors. The scoreboard had two effects (i) it introduces an element of competition between authors that encourages them to write improved versions of their minds, and (ii) it identifies for authors the best minds to incorporate into their own minds as *sub-minds* or modules.

1.2 $Mind_M$ Services

Authors are actively encouraged to "cheat" or use other author's minds as modules in their own minds. This type of

integration is made possible through remote reuse of online components. The manner in which existing minds can be integrated into new minds takes a number of forms. For example an author can select the two minds at the top of the scoreboard and wrap a high level mind around them. The high level mind will make the decision on which sub-mind to call at which point in time, resulting in a division of the problem state-space between the two sub-minds. Obviously this strategy can be employed with any number of sub-minds. Using the terminology introduced in (Humphrys, 2001), this is a $Mind_M$ service.

A number of other strategies involving varying levels of competition between sub-minds are described in (Humphrys and O'Leary, 2002) but have not been tried out as part of this project, for reasons that will be described later.

At the time of writing 505 minds had been written and made available for re-use online. Of these, less than 5% were $Mind_M$ services. In total, the scoreboard contains 4684 runs, meaning that the average number of runs for each mind was 9.28 (however, many minds would have been abandoned by their authors after a very small number of runs). In order to assess each individual mind correctly, a much higher number of runs would be necessary, but very difficult to carry out given the time required (for statistically sound results, Tyrrell (1993) tested each action selection mechanism he implemented 1650 times).

For this reason, we rely on the community of authors, or interested third parties to run the tests. They do so by selecting their mind, or a mind they are interested in and running it in the world. Over time the better minds will rise up the scoreboard at the expense of the poorer minds. These better minds are then more likely to get artificially selected for inclusion in a $Mind_M$ service. In this way, we expect to see the evolution over time of successively improved minds (Darwin, 1859, Ch. 1).

The results outlined here are based on ten $Mind_M$ services developed by the first author. They are accompanied by a description of the strategy used for developing each $Mind_M$ service based on the scores available in the scoreboard. In addition we describe how these $Mind_M$ services, which are also available online, will be subject to the same sort of evolution by artificial selection that the original minds are.

1.3 Roadmap

This paper will proceed as follows. Section 2 provides an overview of the World-Wide-Mind project, including a detailed description of the architecture. Section 3 describes Tyrrell's Simulated Environment (SE), as a re-implementation of this virtual world was used for the research outlined in this paper. Section 4 shows how the implementation of Tyrrell's SE was put online as part of the World-Wide-Mind architecture. This includes a description of the format for the scoreboard that is core to evolving the minds. Section 5 provides a discussion of the ongoing project that started with 234 diverse authors. We explain how their minds were made available online, subsequently run and recorded in the scoreboard. These minds were then selected based on their scores, and integrated into $Mind_M$ services, according to different strategies, as described in section 6. Section 7 provides the results for these $Mind_M$ services. Section 8 provides a discussion of these results. Section 9 describes further work, and section 10 will place our work in context by showing its relationship to other modular forms of artificial minds and agent architectures.

2 World-Wide-Mind

The World-Wide-Mind (WWM) is a scheme for facilitating the development of large minds through publishing virtual worlds and sub-minds online as web services. A web service is effectively a programming language object whose methods are invoked over HTTP, the language of the web rather than being invoked in a local environment. The WWM specification (Humphrys, 2001a) defines three types of entity, a *world service*, a *mind service* and a *client*. Messages sent between entities are composed in the XML-like protocol of the WWM named SOML, the Society of Mind Markup Language (this is an update of the protocol referred to as AIML in (Humphrys and O'Leary, 2002b)).

2.1 World service

A world service is a virtual world (real worlds involving embodied agents are possible, though none are implemented as yet) that supports a set of SOML messages for retrieving the *state* of the world, and for instructing the agent in the world to execute a given *action*. For most problems, the state refers to the current perceptions of the agent in the world, although this could obviously be extended to include any type of information about the world. The action will be some value or symbol that can be interpreted by the agent in the world.

A world service defines its own format for state and action, so that when another entity queries it for its state, the requesting entity must parse the state according to the rules defined for that particular service. Similarly the world service will define its own format for its actions. This removes the requirement for the type of symbolic languages used for agent communication in multi-agent systems and on the Semantic Web, as discussed in (O'Leary and Humphrys, 2002).

2.2 Mind service

A mind service generally represents an action taking mechanism developed for a given type of world service. The mind service will support the necessary SOML messages to receive the state from the world service, parse it according to the format defined for the world service, and then suggest one of the actions possible for that particular world.

Once a mind is created there are three options available to the mind author. Firstly they can put the mind online with no additional information describing the algorithm used by

the mind. This effectively means that the mind will be used as a black box and it can only be selected for reuse in a $Mind_M$ service based on its performance.

Secondly, the author of a mind service could publish details on its implementation on an accompanying web site. $Mind_M$ authors who wished to reuse this mind as a component could then design their $Mind_M$ to take advantage of the published attributes of the mind.

Thirdly, minds could propose an action accompanied by some additional information to help in the higher level action selection. Such minds (referred to as $Mind_I$ services in (Humphrys and O'Leary, 2002)) are aware that they may not be the only mind involved in selecting actions for a particular body, and would need to be designed appropriately. For example, Humphrys (1997) describes how sub-minds that use reinforcement learning can calculate values that represent how much they want to be obeyed in particular states.

In the experiment carried out here, minds are not accompanied by web sites describing their algorithm, nor do they propose weights with their chosen actions, and as a result must be reused as black boxes. Future research will address methods for proposing weights in societies of sub-minds that implement heterogeneous algorithms.

2.3 Client

The client is the entity that is responsible for *plugging* a mind into an agent in a world. It does this by selecting a world service at a given URL and selecting a mind service at another URL. It then informs both the world service and the mind service that it intends to start a *run* involving them. Both services will typically create new instances of the algorithm that they represent and return a unique ID for this instance to the client. The world service algorithm implements the problem (e.g. Tyrrell's SE) and the mind service algorithm implements a solution (one author's mind).

The client will use this ID in all communication with both services. Both IDs along with both URLs uniquely identify the subsequent interaction between the entities, which is termed a *run*. A run consists of constant iterations of the following steps:

(i) Client requests the world service to return its state.
(ii) Client takes the world state and sends it to mind service, which then returns the action that it suggests for the given state.
(iii) Client takes the action selected by the mind and sends it to the world service, instructing it to execute this action in the world.

This process continues until the world service sends a message to the client to say that the run has ended. The client will then send a message to the mind to tell it that it is no longer needed for this run.

During a run, the world service does not need to know which mind service is being used to select its actions (although in reality as we'll see, this information is sent so that it can be recorded on the scoreboard). The mind service will normally not need to be sent any information regarding the world service, since it will have been created specifically for that particular problem. The client will only need to know the URL of both services, but will not need to understand anything about the representation of the state or action for the world, since its only function is to pass this data between the services.

2.4 SOML

The Society of Mind Markup Language (SOML) is the XML-like language that defines the general format for the messages that are sent between entities on the WWM. The protocol is defined in full in (O'Leary, 2003). In brief, the following are the core messages:

(i) `newrun`: Can be sent to any type of service to request that a new instance be created.
(ii) `endrun`: Can be sent to any type of service to inform it that a given run has ended. This type of message can also be sent by a world service to a client to inform it that the run has ended in the world.
(iii) `getstate`: Can be sent to a world service to request that its state be returned.
(iv) `getaction`: Can be sent to a mind service to request that an action be suggested for the state that is contained in the request.
(v) `takeaction`: Can be sent to a world service to instruct it to take the action contained in the request.
(vi) `getdescription`: Can be sent to any type of service. The response will typically contain information such as the author of the service, the date it was last modified and so on.
(vii) `getscoreboard`: This request can contain a URL of a given mind and a number (x). When received by a world service, the top x scores achieved by the mind identified by the URL should be returned. If the request does not contain any parameters, all scores should be returned. The format of the score is also defined by the service, as described in the section below.

2.5 Scoreboard

As described above, world services should support a request named `getscoreboard`. This should return the scores achieved by a given mind, or indeed all scores achieved in the world. The world service can define its own format for its score, the only requirement being that the scoreboard should be ordered in some way that places better performing

minds closer to the top of the scoreboard than those minds that they outperform.

One possible way of constructing a scoreboard is to record every run in the world, with the best run at the top. Using this approach the size of the scoreboard grows very large very quickly (this is the approach we used, and the scoreboard is already over 2MB in size), and the ordering on the scoreboard may appear misleading. For example, the mind that scored the best run may have been very fortunate in the world that it was given for that particular run, and may perform badly in most other instances of the world.

Another possibility would be to record statistical information on the performance of each mind, so that instead of recording every score, the average and standard deviation of its scores are recorded, and the scores are ordered appropriately, more accurately reflecting the relative performances of the minds.

We used the first format for this experiment. Since all information is recorded in the scoreboard, and is available to others using the `getscoreboard` request, anyone who is interested can perform their own statistical analysis of the scores and select the best mind based on this.

The information that is recorded for each score includes the URL of the mind service, the date and time of the run, the name and contact details (e-mail/URL) of the author of the mind, the name and contact details of the user, or person who started the run and most importantly the score achieved, in a format decided upon by the author of the world service.

3 Problem Domain

At present there are two world services online that conform to the latest version of SOML. The first of these is a simple blocks world implementation, for which twelve mind services have been developed, including two $Mind_M$ services. A description of the problem and the design of the minds is discussed in earlier work (O'Leary and Humphrys, 2003).

A second available world service, which is the problem world at the centre of the work described here, is a re-implementation in Java of Tyrrell's Simulated Environment (SE) (Tyrrell, 1993). The SE will be described in this section. Its implementation as a world service will be dealt with in the next section.

3.1 Tyrrell's Simulated Environment

Tyrrell's PhD thesis examined a number of different action selection mechanisms when applied to a complex, multi-goal problem. In order to do this he wrote a virtual environment that modelled an animal in a heavily populated, dangerous environment. The primary goal of the animal was to mate, as this would ultimately determine how likely it was to pass on its genes to subsequent generations. However, in order to mate the animal needed to survive in the world long enough to be presented with sufficient opportunities to court and mate with other members of its own species.

The world which the animal occupied had fifteen different features, including food and water (both of which could be toxic), a den, cover and shade. It also featured several different types of animal; predators and non-predatory animals that needed to be avoided, prey that could be consumed and animals of the same species that could be mated with. The world experienced changes in weather and also cycled through different times of day.

The animal had its own models of perception and navigation that provided it with information about its surrounding environment and memories of places that it had previously visited. It was also provided with motor control, or the ability to convert a chosen action into the necessary movements of the body. Each of these three systems were error-prone, in that the animal could misperceive features in its immediate environment or *forget* about a feature it had encountered. Also, any action that it chose was subject to a low probability of being incorrectly executed.

3.2 Action Selection

At each time-step the animal was required to choose one of 35 actions to execute. These included 16 different moving actions for travelling at different speeds in different directions, 9 different looking actions that could improve its perceptions for a given direction, or allow it to quickly scan its environment for predators, and various actions for eating, sleeping, resting, cleaning and mating.

The animal should choose the action that is most likely to increase its expected future genetic fitness i.e. the number of times it mates in this case. In order to do this the animal should maintain good health by eating and cleaning. It should also avoid an early death by hiding from predators and looking out for other dangerous places such as cliffs and marshes. In addition it should try to make its way home at nighttime to sleep.

Tyrrell implemented six different algorithms for the animal's action selection in the environment. His own algorithm which he named the *Extended Rosenblatt and Payton* algorithm performed best in the tests he conducted. His results were later improved upon by Bryson with her Edmund algorithm (Bryson, 2000).

3.3 Discussion

It is obvious that the action selection problem for Tyrrell's SE is difficult. All approaches to solving it so far have focussed on building a single algorithm that can account for all the goals that must be satisfied by the animal. What we propose is that rather than designing the best solution from the beginning, we start with a solution and then add to that in subsequent implementations. This would be a difficult task if every author were required to understand every implementation that came before him. We intend to examine if it is possible to integrate solutions that we do not

understand completely or at all, that are judged simply on their performance. This may become easier if the algorithms are available for remote re-use, online.

4 Tyrrell's SE online

An implementation of Tyrrell's SE is available online as a WWM world service at w2m.comp.dit.ie. When queried for its state it will return a vector on numbers representing the animal's perceptions and memories and well as some additional information about the environment (it is given all the information that was available to the action selection mechanisms developed by Tyrrell).

It will accept as an action any number from 0 to 34, each representing one of the 35 actions that can be selected by the action selection mechanism. For a detailed explanation of state and action see w2m.comp.dit.ie.

Any mind service that will be created for this implementation of the SE should accept the state, process it and return an action i.e. it will serve as an action selection mechanism, except it will be located somewhere else on the Internet.

4.1 Scoring in the SE

Every run in the world service is recorded on the scoreboard, including the URL of the mind, author and user details as explained. The scores are ranked according to two criteria, firstly the number of times that the animal mated in the environment during that particular run, and secondly according to how long the animal survived in the environment, in time steps.

In addition, various other items are recorded for each run. This information is useful in developing a profile for the mind that was involved in the run, which will be used when we attempt to integrate individual minds into $Mind_M$ services.

This additional information includes (i) the number of times that the animal successfully ate food or drank water, (ii) the number of times that the animal ate toxic food or drank toxic water, (iii) the number of times the animal caught and consumed prey, (iv) the number of times the animal was injured by predators, other animals or by visiting dangerous places and (v) the amount of time that the animal spent in the environment without knowing how to return to its den.

In addition, for each run in the world, the number of times that each of the 35 actions was chosen is recorded. Using all of this information it is possible to infer to some degree the type of strategy that was used by the mind. The more information that is recorded, the easier it is to build up this type of profile, but there are real limits that must be observed. As mentioned earlier, the size of the scoreboard is already over 2MB in size. The more information that is recorded the larger this will grow.

A decision was made to record the data for each run rather than simply maintain statistical information for each mind. This meant that anyone interested in using a particular mind could perform their own analysis of the runs conducted using the mind, and develop their own profile for it. This also meant that the scoreboard was ordered according to each run, rather than the performance of each mind in all its runs collectively.

A HTML version of the scoreboard is available at w2m.comp.dit.ie. The SOML for the scoreboard can be retrieved using the `getscoreboard` request. This will contain all the additional information recorded for each run.

5 Format of experiment

The world service is publicly available for anyone to use (at w2m.comp.dit.ie), and anyone is free to develop a mind service to work with it, once they observe the correct formats for state and action. We now explain how the first set of minds were built.

5.1 Community of Students

In order to get a large number of minds online for the purposes of this research, an assignment was given to two classes of undergraduates studying for an honours degree in Computer Applications at Dublin City University. Altogether, over two hundred students developed mind services for the version of Tyrrell's SE that is online.

In order to focus on the action selection aspect of the assignment rather than the networking (relatively trivial though it is) the students were given a tool kit that they could use to develop the mind and put it online. The majority of the students developed their software in this way, although a minority did implement minds using web technologies such as PHP and Java Servlets.

Students were permitted to create as many different mind services as they wanted over a period of six weeks. At the time of writing (two weeks after the deadline for the assignment) 505 had been put online. Once a mind was put online, it was automatically run in the world to record some account of its performance. Many minds were then run again several times by their authors or others who wanted to observe the performance of the mind, with a view to integrating the mind in their own $Mind_M$ services. Some students developed such minds, although this can only be inferred from the response times of the mind service (it is not unusual for a $Mind_M$ service to take over an hour and a half to perform a run in the world, in particular when it is calling on more than one sub-mind).

5.2 Artificial Selection of Minds

As the experiment continued, the better minds rose to the top of the scoreboard. Over a prolonged period of time, the artificial selections made by authors and users will result in a larger number of runs in the better minds, which will in turn result in their being selected more often for inclusion in

$Mind_M$ services, and consequently the creation of better minds.

In addition, since there is useful information apart from scores recorded on the scoreboard, authors can observe the strategies that are employed by the better minds (in one case, it would appear the *sleeping* was preferred to *eating* in order to live longer) and try to employ these strategies in the minds they are authoring without remotely re-using the original mind.

6 Integration strategies

Using the large set of mind services that had been put online, we hand designed a further ten $Mind_M$ services. The first problem that we are faced with in developing a $Mind_M$ service is in identifying at least two existing minds that could work together. A second problem is in identifying when to select the action proposed by either mind. One simple approach would be to start a run in one mind and let it run for a finite number of steps, then stop it and start another mind which will then run and so on. A problem with this is that everything that the mind may have learned up to that point from its experience of the environment is lost. A second approach might be to start both minds and present both of them with the state of the world at every time-step, letting them all suggest an action, but only select the action proposed by one of them, possibly based on a voting strategy. At least with this strategy all sub-minds get to experience the environment for the entire duration of the run, but now all the minds think that they are being listened to at every point in time, which is clearly not the case.

We describe here a number of different approaches we took to selecting and integrating two minds.

6.1 $Mind_M$ I

The first strategy used for building a $Mind_M$ service was to select the mind that had achieved the best run in the world, i.e. the one at the top of the scoreboard, and examine its entries in the scoreboard. From this we were able to develop a profile of the mind, based on the actions that it had chosen in its runs as well as the additional pieces of information described above.

The profile that was developed for the mind was based on its average performance across all its runs, as well as analysis of individual scores. A summary of its profile is given here

1. Lives for an average of seven days in the world (a day is 500 time steps).
2. Often employs sleeping as a method for living long, and consequently misses out on some mating opportunities.
3. Favours drinking over eating. Does not eat enough.
4. Efficient cleaner.
5. Has not been injured by other animals or dangerous places, possibly because they have not been encountered yet. Has not used its *move fast* actions at all, which implies that it has not had to escape from predators.

Based on this we selected a mind that could compensate for the deficiencies highlighted in the above profile. We identified the failure of the animal to eat enough to be its primary fault, so we analysed the scoreboard to find a mind that regularly selects eating actions, and still occupies a high position on the scoreboard. The mind selected was the one that ate the most on average in a run.

To integrate the two minds, we employed a simple strategy. We started a run in both minds at the beginning of our run. We selected the action of our main mind the majority of the time. We selected the action of our second mind whenever it suggested an eating action.

There are obviously many shortcomings in this approach. If the first mind is pursuing some goal such as *avoid predator*, and the second mind suggests an eating action then the animal will stop to eat, and probably end up eaten himself. However, the second mind we selected is relatively high up on the scoreboard, so it would be fair to assume that it is aware of the presence of the predator as well, and would not select an eating action while a predator was close.

6.2 $Mind_M$ II

A second approach we took to developing a $Mind_M$ service was to select, once again, the mind at the top of the scoreboard, but this time make a different judgement on how to select an *eating* mind for it. In this case we selected the mind that was the most successful eater i.e. the one that ate non-toxic food every time that it selected an eating action. Other minds had selected eating actions when it was incorrect to do so. We used the same approach to integrating the minds i.e. the eater was only listened to when it suggested an eating action.

6.3 $Mind_M$ III

For this mind, we selected as the primary mind the one which had the best average score, rather than the highest individual score. We created a profile of this mind and saw that it was quite similar to the mind described in section 6.1 above. It is not surprising that two successful minds should have a similar strategy, even if this is not the best strategy. One may itself be a $Mind_M$ service that wraps up the other mind, using the actions that it suggested the majority of the time. Alternatively, the author of one of the minds may have observed that the other mind was achieving good scores on the scoreboard. Then using the information in the scoreboard, he may have developed his own profile of the mind and attempted to mimic that strategy in his own implementation. Another reason may of course be that they both arrived at the same design independently.

We then integrated the mind we had selected at this stage with the *successful eater* we had identified for $Mind_M$ II above, using the strategy described earlier.

6.4 $Mind_M$ IV & V

In this case we picked another successful eater that had a better average overall performance in the world than the eater used above. We integrated it with the two main minds we had been using so far to give $Mind_M$ IV and $Mind_M$ V.

6.5 $Mind_M$ VI

In order to score well in the world the animal must survive for a long time and mate as frequently as possible. With this in mind we selected two minds which we felt would satisfy these two goals best. The first mind is the one that survived longest in the world on average, and the second one also lived quite long (more than two days) but mated more frequently. We integrated them in a manner similar to the minds built earlier, obeying the actions of the first mind in all cases except where the second mind proposed a mating related action (*courting* or *mating*).

6.6 $Mind_M$ VII & VIII

Some mind implementations focussed entirely on mating to the cost of all other goals. These minds did not survive long in the world, but mated more frequently in the short period of time that they did survive. This suggested that they were better at identifying good mating opportunities since they were not concerned with other goals. We picked one such mind and integrated it with the main mind used in section 6.5. This was $Mind_M$ VII.

We then selected another frequently mating mind that appeared to have an interest in a limited number of other goals such as eating and cleaning. We integrated this with our long living mind, taking its suggestion whenever it proposed a mating related action, and named it $Mind_M$ VIII.

6.7 $Mind_M$ IX

A $Mind_M$ service can call on, or wrap up, any number of sub-minds that it invokes remotely. The interface to these minds remains the same regardless of how they are implemented, including situations where the sub-minds are themselves $Mind_M$ services. For the construction of $Mind_M$ IX we integrated $Mind_M$ II as described above with a frequent mater. This whole mind then uses three sub-minds to select its actions, with the frequent mater only being listened to when it proposes a mating action.

6.8 $Mind_M$ X

The obvious shortcoming of all the minds described thus far is that a mind that has been identified as being strong in achieving a particular goal is only called upon when it proposes the final consummatory action. This is because it is difficult to identify when a mind is suggesting an appetitive action that will ultimately lead to the achievement of the goal. For example, although we have used sub-minds that we identified as being best for solving the *eating* or *mating* goals, we have not accepted the actions that they proposed when they may have started to solve that particular goal. The reason for this is that we have no way of knowing that they are beginning to solve that goal, since the only information we are provided with from the sub-mind is the action that is proposed.

In an attempt to address this particular issue, we implemented a $Mind_M$ service that performed its arbitration using the *Drives* algorithm that Tyrrell had developed for the Simulated Environment. This algorithm worked by taking all the animal's perceptions and memories and calculating simple *motivational strengths* or *drives* for each of the sub-goals such as finding food, water or mates, or avoiding predators or dangerous places.

For our implementation, we calculated the *drive* for each sub goal and selected a sub-mind based on the winning, or highest drive. Whenever *mating* won the competition we selected a frequently mating mind; if eating won we selected a successful eater, and for all other drives we defaulted to the most successful overall mind.

Clearly, any number of strategies could be used for determining when to switch between sub-minds, as will be discussed in the *Further Work* section below. This $Mind_M$ service was simply a demonstration of one possible strategy.

6.9 Discussion

We have outlined how ten different $Mind_M$ services have been developed from existing components. As can be seen there are two main issues involved in the creation of a $Mind_M$ service using minds in this black box fashion:

(i) *Profiling each mind based on its performance in the world*. The profile should provide sufficient information to be able to identify what goals are handled well by each mind.

(ii) *Deciding when to switch between minds*. In the first nine implementations above, minds were selected and deselected based entirely on an action that could be related to only one goal. When appetitive actions can be associated with more than one goal (e.g. move towards direction of nearest mate) it is difficult to infer that the sub-mind should be given control of the agent.

7 Results

Firstly it should be stated that the implementation of Tyrrell's SE used here is not identical to that used for other published results by Tyrrell (1993) and Bryson (2000). The first author reimplemented the SE using Java, based on the description provided in Tyrrell's thesis and also his freely available C code.

7.1 Tyrrell's Extended Rosenblatt and Payton Algorithm

Tyrrell's algorithm used a hierarchy of nodes that take input from both internal and external stimuli and pass excitation on to nodes at lower levels in the hierarchy. Nodes at the lowest level represented actions, and at every point, the action node with the highest level of activation would be selected for execution. This system allowed all actions to be considered at each timestep, and for separate parts of the creature's mind to contribute excitation to the same action, facilitating the selection of compromise actions, or actions that can contribute to the satisfaction of more than one goal. Using this algorithm, Tyrrell's animal mated an average of **8.09** times in a lifetime.

7.2 Bryson's Edmund Algorithm

Bryson's Edmund algorithm was implemented for several different versions of the SE. This algorithm allowed the creature for focus its attention on the highest priority goal that could be addressed in a particular state. For this reason it was able to dramatically reduce the amount of computation that was required at each timestep, and was also able to improve on the results achieved. Bryson feels that the simplicity of the implementation leads to a higher probability of finding the correct parameters for the system. Using this algorithm, Bryson's animal mated an average of **9.12** times in a lifetime.

7.3 Implementation of Simple Minds

Although the internal design of each of the minds developed for this experiment were not available to the authors of $Mind_M$ services, subsequent assignment submissions by the cohort of students contained descriptions of the algorithms used. Minds tended to be hand designed, priority based algorithms, where the animal checked various goals in order, obeying the first one that could be satisfied. The ordering which was given to the goals, the manner in which an action was selected for a goal, the manner in which it could be determined whether a goal should be addressed in a particular state and the amount of state that was saved between actions determined the relative success of each of the minds. Most minds implemented highly reactive algorithms, although the most successful individual mind maintained its own (more accurate) record of recently visited states, in addition to that provided by the world service.

In total there were 505 minds that were involved in 4684 runs. The minds were scored firstly on the number of times they mated, and secondly on the amount of time-steps they survived in the environment. The overall average number of times that the animal mated across all runs was 1.23 times. The animal lived for an average of 1066 time steps.

The top ten minds mated an average of 3.3 times per run, and lived for an average of 1916 time-steps. For the top one hundred minds these figures were 2.6 and 1686 respectively.

The best performing mind mated an average of 3.65 times and lived for an average of 1378 time-steps. The mind that lived the longest in the environment was able to survive for an average of 4413 time-steps, although its strategy was to find its den and sleep continuously, consequently it never mated. The average lifetime of the longest living mind that was also able to mate was 3348 time-steps, spending 36% of its time asleep.

7.4 $Mind_M$ Results

The results achieved in this implementation of the Simulated Environment do not rival the results published by Tyrrell or Bryson for this particular problem. The scores achieved by the best $Mind_M$ services *did* however beat the scores achieved by all the other minds developed, and did so by selecting and integrating these as sub-minds. This point will be returned to in the *Discussion* section.

The results achieved by these $Mind_M$ services are given in Table 1 below.

	Average Performance			**Best Performance**		
Mind	**Mated**	**Lived**	**Pos.**	**Mated**	**Lived**	**Pos.**
I	2.67	2365	22	8	3196	25
II	3.16	2226	9	12	4768	2
III	2.89	1182	15	10	1936	8
IV	3.67	1276	2	9	1769	12
V	2.2	2421	41	5	2886	116
VI	1.96	2827	61	7	4496	34
VII	3.7	1539	1	20	3916	1
VIII	1.53	226	120	6	248	115
IX	2.24	2776	36	5	4192	127
X	2.46	979	27	7	2013	47

Table 1: The number of mates, time-steps lives and overall positions of each of the ten MindM services.

$Mind_M$ VII performs best on average and also reached the top of the scoreboard with the best individual score.

8 Discussion

From the results that have been shown it is clear that it is possible to create superior minds from existing minds if the correct minds are chosen, and if the integration strategy suits the algorithms that are being executed by each of the minds. In our case, the best performing mind used the mind that had lived the longest in the environment and the most aggressive mater. An aggressive mater will constantly prioritise mating over the pursuit of other goals. The mind that it was integrated with appeared to distribute its attention among multiple goals. When the two worked together they appeared to get the best of both worlds. The mating mind

sought out opportunities to mate whenever possible, whereas the remainder of the time the other part of the mind was focussed on ensuring that the creature lived long enough to be presented with sufficient opportunities to mate.

In many cases minds were put together that did not work well, at least with the type of strategy that was used for integration. For example, $Mind_M$ X should have outperformed all other minds because of the care that was taken in switching between the various sub-minds at the correct time. However, in this situation it would seem that all that happened was that the sub-minds were interrupted while pursuing particular goals that the other minds could not pick up on. Even in this situation, the minds that were selected for the mating and eating goals were often not the ones that actually executed those actions. Better minds, such as our winner $Mind_M$ VII use sub-minds that are very different from each-other. It is easier to divide the state-space between these types of minds, as their respective roles are more clearly defined.

Tyrrell (1993) and Bryson (2000) both built action selection mechanisms that outperformed our collection of minds. However, our sub-minds require much more testing, as does the re-implementation of Tyrrell's SE. Also, the action selection needs to be more sophisticated. Minds need to be designed to co-operate with the types of action selection that is taking place in other minds. For example, minds may need to flag appetitive and consummatory actions, perhaps by returning strengths or W-values (Humphrys, 1997) with each action.

9 Further Work

This work is at a relatively early stage. The assignment that was given to the undergraduate students only ended recently, thus limiting the time available to develop more sophisticated $Mind_M$ services. However, it is clear that if minds are available online as services, they effectively exist forever, and can be used by anyone with access to the World-Wide-Web. This should mean that over time, as more runs get executed and more scores are recorded, it will be possible to get a clearer image of the profile of each mind. With more information about the profiles of each of the minds, it should be possible for interested authors to develop better $Mind_M$ services, and consequently get better ideas for how to write minds for the animal. Evolving a mind in this fashion is different to developing it as part of a single project. Evolution requires time and the cumulative effects of much recombination in order to be successful (Dawkins, 1986). In nature this happens by natural selection. In our situation what is required is the conscious selections of a community of interested researchers.

9.1 Integration

Our integration strategies were relatively straightforward, with the exception of the *drives* based mind, which failed to perform well in the tests conducted. We intend to examine related work in behavior/goal selection and agent architectures which can be tried out with our architecture. Some of the work in this area is discussed in the *Related Work* section below.

Earlier publications discussed the possibility of allowing sub-minds exchange numeric weights that express the degree to which one sub-mind *wants* to run at any particular point in time. Such a mind could use Hierarchical Q-Learning (Lin, 1993) or W-Learning (Humphrys, 1997) as outlined in (Humphrys and O'Leary, 2002). While approaches such as these will be tried for certain problems, they may not be appropriate for the research described here, in their current form. Firstly, both learning schemes require specific types of problems. Tyrrell (1993) outlined why his SE is not suitable for these types of algorithms, given the size of the state-space and the requirement for experience of all different types of states. Secondly, each mind service author would need to understand and correctly implement the algorithm for proposing an action with a given weight, which will be possible in certain domains but was not possible here. It may be possible if proxies (Schmidt *et. al.*, 2000) are wrapped around sub-minds that will then implement the required algorithm.

10 Related Work

Bryson (2001) proposes a design methodology named *Behavior Oriented Design (BOD)* that defines an iterative engineering approach to the development of complex, complete agents. BOD stresses the need for an initial decomposition of the agent requirements into a set of sub-systems and then a continuous revision, weighing the level of granularity of each system against the requirement for arbitration between these. Such a cyclic approach could be applied to our own work in the development of a single agent where the requirements are considered in terms of the available sub-minds.

Other modular forms of agent design, including Brooks (1986) and Humphrys (1997) address issues in behavior decomposition and arbitration, while the whole animat approach to AI (Wilson, 1990) is predicated on the idea of starting with simple whole minds to which new components are added over time. This is the approach we took to the development of $Mind_M$ services.

Work in Internet-agent systems is related to our own in that the various components are distributed throughout the network. While we have no initial requirement for the type of symbolic languages (FIPA, 2002) or complex platforms (Poslad et. al., 2000) that are used in this area, we still face similar technical issues in terms of network latency.

Distributed systems programmers develop components and web services that are automatically located and invoked by clients, primarily in the area of e-Commerce. A great deal of work in this area is now focussed on the development of profiles for each of the services that run over the Web and more recently the Semantic Web (DAML-S Coalition, 2002). Such profiles however are written at

development time by the creators of the service and are not automatically inferred and updated from the performance of the service.

11. Summary

Complex minds are necessarily composed of a multitude of different algorithms and structures (Minsky, 1986). Such minds are difficult to build because the required expertise is rarely available to a single project or lab. The World-Wide-Mind project aims to make it easier to build these types of minds, by proposing the online reuse of existing mind components.

The work described here has shown how a mind can be evolved over a period of time by utilising the independent expertise and artificial selection of a community of authors.

Acknowledgements

The authors wish to thank the many students from CA3 and CAE3 in Dublin City University whose hard work made this research possible. All author's names can be seen on the scoreboard at w2m.comp.dit.ie; those used here, and the $Mind_M$ services they were used in are Peter Kehoe (version 7 of his mind was used in I, II & V, version 10 in VI, VII, VIII & X), John Dundon (I), Colin Sheridan (II & III), Eugene Gibney (III, IV, VI & X), Lousie O'Nolan (IV & V), Mark Ruddy (version 8 used VII, version 9 in VIII), Andrew Pimlott (IX) and Michael Dowling (X).

The authors also wish to thank two anonymous reviewers for their helpful comments.

References

Brooks, Rodney (1986), A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*

Bryson, Joanna (2000), The Study of Sequential and Hierarchical Organisation of Behaviour via Artificial Mechanisms of Action Selection, *MPhil Dissertation: University of Edinburgh*

Bryson, Joanna (2001), Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents, *PhD Thesis, MIT*

DAML Services Coalition (2002), DAML-S: Web Service Description for the Semantic Web, *in proceedings of First International Conference of the Semantic Web (ISWC-02)*

Darwin, Charles (1859), The Origin of Species.

Dawkins, Richard (1986), The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design, *W.W. Norton & Company*

FIPA (2002). FIPA ACL message structure specification. *http://www.fipa.org/specs/fipa00061/SC00061G.html*

Guillot, A. and Meyer, J.-A. (2000), From SAB94 to SAB2000: What's New, Animat?, *Proc. 6th Int. Conf. on Simulation of Adaptive Behavior (SAB-00).*

Humphrys, Mark (1997) Action Selection methods using Reinforcement Learning, *PhD Thesis, University of Cambridge*

Humphrys, Mark (2001), The World-Wide-Mind: Draft Proposal, *Dublin City University, School of Computing, Tech Report CA-0301 computing.dcu.ie/~humphrys/WWM/*

Humphrys, Mark and O'Leary, Ciarán (2002), Constructing complex minds through multiple authors, *Proc. 7th Int. Conf. on Simulation of Adaptive Behavior (SAB-02)*

Lin, L-J (1993), Scaling up Reinforcement Learning for robot control, *10th Int. Conf. on Machine Learning*

Minsky, Marvin (1985), The Society of Mind, *Simon and Schuster*

O'Leary, Ciarán and Humphrys, Mark (2002), Lowering the entry level: Lessons from the Web and the Semantic Web for the World-Wide-Mind, *1st Int. Semantic Web Conf. (ISWC-02)*

O'Leary, Ciarán and Humphrys, Mark (2003), Building a hybrid Society of Mind using components from ten different authors, *in Proceedings of Seventh European Conference on Artificial Life.*

O'Leary, Ciarán (2003), SOML Specification, *http://w2m.comp.dit.ie/services/documentation/*

Poslad, S, Buckle, P and Hadingham, R (2000), The FIPA-OS agent platform: Open Source for Open Standards, i*n 5th Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-2000)*

Schmidt, D, Stal, M, Rohnert, H and Buschmann, F (2000), Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, *Wiley and Sons*

Tyrrell, Toby (1993), Computational Mechanisms for Action Selection, *PhD Thesis, University of Edinburgh, Centre for Cognitive Science*

Wilson, S.W. (1990), The animat path to AI, *Proc. 1st Int. Conf. on Simulation of Adaptive Behavior (SAB-90)*