Motivations for MADbot: a *M*otivated *A*nd Goal *D*irected Ro*bot*

Alexandra Coddington
Department of Computer and Information Sciences
University of Strathclyde
Alex.Coddington@cis.strath.ac.uk

Abstract

This paper presents the MADbot architecture which enables a motivated autonomous agent to generate goals in response to changes in its underlying drives or *motivations*, while it is both planning and executing. We use a Mars rover domain to illustrate the workings of this architecture and describe two ways of modelling motivations and goal generation: the first decouples the model of motivations and goal generation from the workings of an AI planner; the second models motivations and goal generation implicitly in the planner's domain model. We discuss the advantages and disadvantages of each approach and offer ideas for further work.

Introduction

AI planning is traditionally pursued under the assumption that all goals are known in advance and externally imposed, and that the planning process is finished once a plan has been generated in which all goals are achieved. AI planners are typically not directly connected to executives—an external user supplies problems to be solved, where a problem consists of a model of the agent's current state, the goals to be achieved, and a representation of the activities which the agent is capable of carrying out. Once a plan has been generated, the execution of the plan is then no longer the concern of the planner. There are exceptions to this scenario, for example, work by Knight et al. (Knight *et al.* 2001) and Chien et al. (Chien *et al.* 2001) has relaxed these assumptions by introducing the idea of continuous planning in the Casper system, an architecture for an autonomous planning and control system intended for application in space missions involving onboard autonomy. Such work emphasises the need, in an autonomous system, to integrate the tasks of planning, execution and goal management.

Although autonomous systems are generally created to carry out tasks on behalf of their users, any extended period of autonomy will require the system to react to its environment and to be capable of generating its own goals. In contrast to traditional AI planners, real world autonomous systems must be capable of directing their own behaviour which means they must be capable of generating new goals whilst a plan is in the process of being executed. This requires a model of continuous goal generation as well as the interleaving of planning and execution in order to achieve those goals. The aim of the work described in this paper is to investigate the role that an agent's context may play when it generates goals and plans, where context is defined as the agent's model of its current as well as predicted future states of the environment (this includes models of both its internal and external physical or virtual state) and its desires and preferences which we aim to partly represent by modelling the agent's *motivations*.

In this paper we consider the purpose and mechanism of motivations and describe MADbot, a planning and execution architecture for a motivated, autonomous planning agent. We investigate two models of goal generation: the first model involves goals being generated explicitly in response to changes to the agent's motivations where such goals are then provided to a planner; the second model involves those goals being implicitly generated by the planner.

Motivations

Motivations have been defined by Kunda (Kunda 1990) as "any wish, desire, or preference that concerns the outcome of a given reasoning task". Balkenius (Balkenius 1993) states that all thinking is biased by emotive values and that motivation precedes cognition because cognition depends on the motivational system whereas the motivational system can operate without cognition. The role of the motivational system is to direct an animal towards one of its different engagements, where an "engagement" is defined as a particular activity. Many researchers in the fields of psychology, philosophy and AI have been interested in the way motivations affect the reasoning process (Balkenius 1993; **?**; Ferber 1999) and have attempted to classify and develop taxonomies and hierarchies of motivations (e.g. Maslow's hierarchy of *needs* (Maslow 1971), Ferber's taxonomy (Ferber 1999)). In (Coddington & Luck 2004), motivations were modelled as a set of tuples: *motivation == (name, value, importance)* where *name* is a unique identifier associated with the motivation, *value* is a measure of the current value (or strength) associated with that motivation (which we refer to as *motivational value*), and *importance* is a measure of how important that motivation is to the agent. This representation of motivations is also used in the work described here. For example, an agent might have the motivation *conserve-energy* whose *value* will increase once the agent has recharged its batteries, and then gradually decrease as the agent performs activities which consume battery power.

Once the value of *conserve-energy* falls below a threshold, a goal to recharge will be generated. The *importance* associated with the *conserve-energy* motivation will vary depending upon the agent's current situation and is introduced in order to resolve conflicts that may arise between motivations. So if an agent requires its batteries to be recharged but is in the presence of an interesting object, it will be less important to satisfy the desire to take an image of the object, than it will be to recharge its batteries. It is assumed that different agents will have different motivations depending upon their design (physical or virtual) and purpose, and that motivations may change in response to changes in the agent's environment where such changes may be brought about directly by the agent itself (by executing actions in the environment), as a consequence of physical processes occurring within the environment, or by the activities of other agents. In addition, different agents are capable of carrying out different activities within their environments and such activities may affect the *value* associated with each motivation.

In the work outlined in this paper we view AI planning as a key cognitive component of an autonomous agent. That is, we represent an agent's cognitive abilities by enabling it to invoke an AI planner, and we model the way that motivations may influence the planning process through the insertion of newly generated goals. Whereas previous work by (Coddington & Luck 2004) examined how modelling the motivations or drive of an agent may influence the agent's choice of action (i.e. the agent will favour plans containing actions that will reduce that drive), the work described in this paper aims to examine various other issues raised which were not implemented, in particular the development of a model of goal generation and arbitration.

The MADbot Architecture

The MADbot system (*M*otivated *A*nd Goal *D*irected R*obot*) (Coddington *et al.* 2005) shown in figure 1 is a continuous, motivated, autonomous planning and execution system which is capable of generating its own goals in response to changes in its low-level drives and impulses (motivations). The system was developed in collaboration with Derek Long and Jonathan Gough (University of Strathclyde) and Ivan Serina (University of Brescia).

The system contains several motivations, each of which monitors one or more associated state variables which model the agent's internal state. As the agent executes its plan, the value of each state variable will change which in turn will trigger changes to the values of the motivations. When certain constraints are met, changes to the value of a motivation will cause one or more goals to be generated. The *Motivation* component of the architecture is responsible for updating the value of each motivation in response to changes to the agent's state variables, and for generating new goals whenever the values associated with the motivations satisfy certain constraints. Goals may also be passed to the system by an external source. Once generated, goals are passed via the *Controller* to the *Goal Arbitration* component. The intention of the *Goal Arbitration* component is that it will choose which of the newly generated goals will be achieved

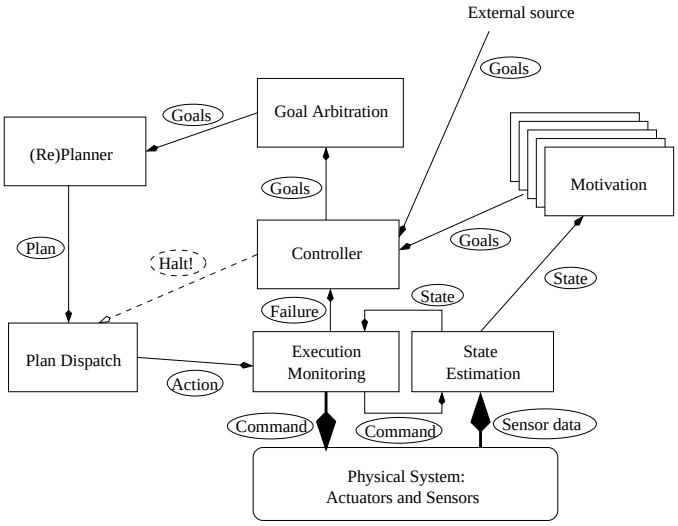

Figure 1: The architecture of MADbot.

by the *Planner* as it may not be possible to achieve all of the goals due to insufficient time and/or resources—this is an area for further investigation. The goals selected by the *Goal Arbitration* component are then passed to the *Planner* component, the planner ADAPT_LPG (Gerevini & Serina 2000), which adapts the system's current plan to incorporate the new goals. Each action of the resulting plan is sent via the *Plan Dispatch* and *Execution Monitoring* components to be executed by the agent. The *Execution Monitoring* component monitors execution and if execution fails, a message is sent notifying the *Controller* which may invoke further replanning. The *State Estimation* component uses the information acquired by the *Execution Monitoring* component to estimate the current state of the system which involves monitoring the values of the agent's internal state variables such as the robot's location and internal battery charge. Such changes to the state variables will in turn be used by the *Motivation* component to update the values of each of the motivations and generate further goals.

The granularity of operations is an important parameter in the design of the behaviour of each element of the system. It is essential that the system should not thrash, continually replanning in response to the smallest changes that occur in its environment but, equally, it is necessary to ensure that motivations are capable of reacting to changes in the (perceived) state in order to generate goals in a timely and appropriate way. Chien et al (Chien *et al.* 2001) also emphasise this need for the responsiveness of the system at different levels of granularity which partly motivates their introduction of a continuous planning system with a shifting plan horizon.

The Mars rover domain

To further illustrate the MADbot architecture, in figure 2 we introduce a simple Mars rover domain which was inspired by the recent Mars Exploratory Rover missions and consists of a rover, or *madbot*, initially located at *waypoint 1* (we have controlled an Amigobot as well as a Pioneer), a rectan-

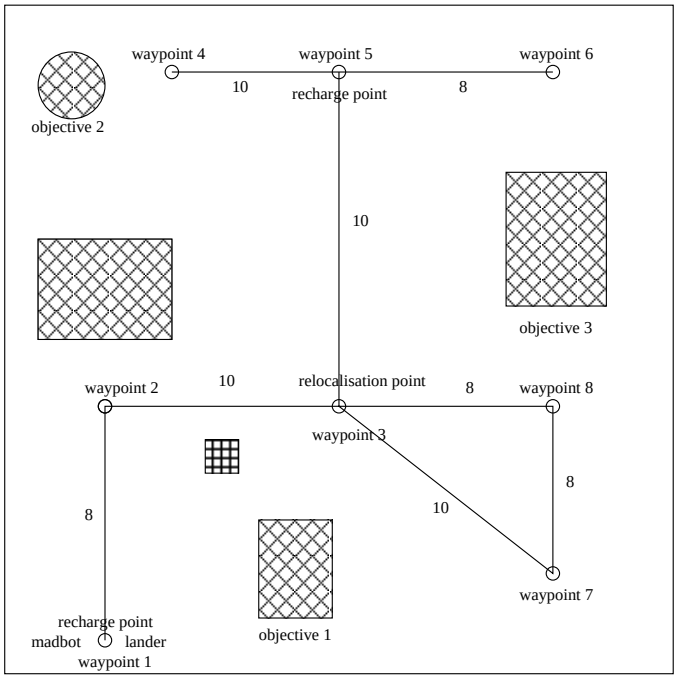

Figure 2: The Mars rover domain.

gular area containing a number of obstacles, a *lander*, two recharge points at *waypoint 1* and *waypoint 5* respectively, and a relocalisation point at *waypoint 3* at which the rover can relocalise. The idea is that the rover is able to explore this environment by following a series of connected waypoints. The domain map may contain a few known locations that may be of interest to an external source such as a science team, and so the rover's initial goal (externally posed by the science team) may be to obtain images of these locations and communicate those images to the *lander*. The rover has an onboard camera that it can use to obtain images of any objects or locations that have been identified as interesting such as *objective 1*. These images are stored onboard the rover in a data storage device, and, as the storage device becomes full, may be periodically sent to the *lander* located at *waypoint 1*. *Objective 1* is visible from *waypoint 1*, which means it is possible to obtain an image of *objective 1* if the *madbot* is at *waypoint 1*. Similarly *objective 2* is visible from *waypoint 4* and *objective 3* is visible from *waypoint 8*. The *lander* is visible from *waypoint 2* which means the *madbot* may go to *waypoint 2* in order to transmit data to the *lander*. *Objective 2* is a calibration target which means the *madbot* must go to *waypoint 4* in order to recalibrate which it must do in order to take images of objectives. The numbers between each pair of waypoints represent both the number of units of battery charge consumed by the rover when navigating between those waypoints as well as the distance between the two waypoints. The rover initially has 100 units of battery, and 30 units of free buffer space in which to store image data. In the following sections we describe first the motivations that have been identified for this domain and then describe the actions that the rover can perform.

The Mars Rover Motivations

The following motivations have been identified for the Mars rover domain and are described together with their associated state variables as well as the goals they will trigger.

- *Conserve-energy*. This motivation monitors the state variable which records the level of battery charge. The motivational value is directly related to the level of charge so when the battery charge falls below a certain threshold, a goal to recharge is generated.
- *Acquire-data*. This motivation causes the rover to periodically acquire new data by obtaining panoramic images while it is exploring its environment. The idea is that a science team may analyse these images and, as a consequence, may identify and direct the rover to visit new sites of interest. This motivation monitors a state variable which records the amount of time which has passed since the rover last acquired new data. As time passes without acquiring new data, the motivational value increases and whenever it exceeds a threshold, a goal to acquire a new panoramic image is generated.
- *Communicate-image-data*. This causes the rover to communicate or transmit the image data stored in its data storage device to a lander, thereby enabling it to free up storage space. This motivation monitors a state variable which records the amount of free space available in the data storage device, so whenever the amount of free storage space available falls below some threshold, a goal to communicate the data with the lander is generated.
- *Relocalise*. The further the rover travels, the less accurate its sense of its current position becomes. This motivation monitors a state variable which records the distance the rover has travelled since it last relocalised. When the motivational value exceeds a threshold, a goal to relocalise is generated which ideally involves the rover rotating slowly and taking careful measurements of distances to nearby obstacles in order to allow it to retriangulate its position. However, for demonstration purposes we make the rover go to a particular location and rotate slowly.
- *Safety*. This motivation monitors the rover's execution status so whenever the rover fails to successfully execute some action, a goal to reach some safe state will be generated. It may be that the safe state simply involves the rover staying at the location it has reached and resetting or reinitialising certain variables (e.g. turn power off, or communicating that it is not available for carrying further tasks). The rover's location should be reported to the planner.
- *Replan*. The planner we are using is stochastic and anytime, so it can generate better quality plans if it is given longer to consider the problem. During periods when the rover is committed to executing an action which takes a long time to execute and the current plan contains many actions, the motivational value increases in order to make the system use the time it has before the current action completes to find a better continuation plan that achieves the current goals. This motivation is unusual in that it

does not generate new goals, but instead triggers an internal activity.

The Mars Rover Actions

The rover is able to execute the following actions which affect the motivations in various ways. These actions are modelled using PDDL2.1 as durative actions. The last four actions are only included in the plan in order to achieve the goals associated with the motivations identified in the previous section.

- *Navigate*. The rover is able to navigate from one waypoint to another provided that the rover is at the initial waypoint, the initial waypoint is "visible" to the rover (this can map directly to the rover's sonar sensor information), and the initial waypoint is connected to the final waypoint (or the rover is able to "traverse" the terrain between the initial and final waypoints). The rover will be at the final waypoint once the action has been executed. Executing the action will have the following effects: the action will consume battery charge and so undermine the motivation *conserve-energy*; the time that has passed since the rover last took a panoramic image will increase and so undermine the motivation *acquire-data*; the distance travelled by the rover will increase and so undermine the motivation *relocalise*.
- *Calibrate*. This allows the rover to calibrate its camera prior to taking an image. In order to execute this action, the rover must be located at a waypoint from which a calibration target (e.g. *objective 2*) is visible (e.g. *waypoint 4*). The action undermines the motivations *conserve-energy* and *acquire-data* as it consumes battery charge and takes time to execute.
- *Take-image*. This allows the rover to obtain an image of a target object. The rover must have a camera which has been calibrated and must be located at a position from which the target object is visible. This action undermines the motivations *communicate-data* and *acquire-data* as obtaining an image causes the data store device to fill up and takes time to execute.
- *Take-panoramic-image*. This action supports the motivation *acquire-data* by enabling the rover to achieve the motivation's associated goal of obtaining a panoramic image. However, because this action requires data storage space, it undermines the motivation *communicate-data*. The state variable associated with *acquire-data* is reset once this action has been executed.
- *Recharge* The rover must be at a recharge point in order to recharge its battery. This action supports the motivation *conserve-energy* and is only included in plans in order to achieve the motivation's associated goal. A consequence of executing this action is that the value of the battery charge state variable is reset. Because this action takes time to execute, it undermines the motivation *acquire-data*.
- *Communicate-data*. The rover must be at a location from which the lander is visible and sends the image data it has acquired to the lander thereby creating free data storage space. This action supports the motivation *communicate-data* and is included in plans in order to achieve this motivation's associated goal (i.e. this action causes the amount of free storage space to be reset). This action takes time to execute and so undermines the motivation *acquire-data*.
- *Relocalise*. The rover must be at a relocalisation point in order to relocalise allowing it to to recalibrate its current position. This action supports the motivation *relocalise* by achieving the motivation's associated goal (i.e. this action resets the motivation's associated state variable). This action undermines the motivation *acquire-data* as it takes time to execute.

```
(:durative-action navigate
 :parameters (?x - rover ?y - waypoint ?z - waypoint)
 :duration  (= ?duration (/ (distance ?y ?z) (speed ?x)))
 :condition (and (at start (can_traverse ?x ?y ?z))
                 (at start (visible ?y ?z))
                 (at start (available ?x))
                 (at start (at ?x ?y)))
 :effect    (and (at start (not (available ?x)))
                 (at start (not (at ?x ?y)))
                 (at end (available ?x))
                 (at end (at ?x ?z)))
```

Figure 3: The PDDL2.1 model of *navigate*.

In the following sections we describe two methods of goal generation. The first method decouples the motivation and goal generation machinery from the planning process, and models goal generation explicitly in response to changes in the agent's motivations. The second approach models the motivations and their associated goals implicitly as part of the planner's domain model. To illustrate in detail these methods of goal generation as well as the the working of the MADbot architecture, we use a simple problem based on the Mars rover domain of figure 2 which shows the initial configuration of the environment.

Modelling Motivated Goal Generation

In this model of goal generation, goals are generated explicitly in response to changes that occur to the agent's motivations. Initially, two goals are presented to the system by an external source, which involve obtaining images of objective 2 and objective 3, e.g. *(have-image madbot o2)* and *(have-image madbot o3)*. The *Controller* sends these goals via the *Goal Arbitration* component to the *Planner*, ADAPT_LPG, which generates a plan. It should be noted that the domain model used by ADAPT_LPG does not model any resources that are consumed or produced by each action (an example of the domain encoding for the action *navigate* can be seen in figure 3). The initial plan output by ADAPT_LPG is shown in figure 4 where each line in the figure shows the time at which execution should commence as well as the name and parameters of the action. In this figure, the parameters *w1*..*w8* are used to represent *waypoint1*..*waypoint8* respectively, *o2* and *o3* represent *objective 2* and *objective 3*, while *c1* represents the camera available on board the rover. The numbers in square brackets in each line indicate the degree to which the action will affect the underlying state variables associated with the motivations: the first number indi-

```
ACT:   0.1:  (navigate madbot w1 w2)      [ 8] [16] [0] [ 8]
ACT:  16.1:  (navigate madbot w2 w3)      [10] [20] [0] [10]
ACT:  36.2:  (navigate madbot w3 w5)      [10] [20] [0] [10]
ACT:  56.3:  (navigate madbot w5 w4)      [10] [20] [0] [10]
ACT:  76.3:  (calibrate madbot c1 o2 w4)  [ 1] [20] [0] [ 0]
ACT:  96.4:  (take-image madbot w4 o2 c1) [ 5] [20] [5] [ 0]
ACT: 116.5:  (calibrate madbot c1 o2 w4)  [ 1] [20] [0] [ 0]
ACT: 136.6:  (navigate madbot w4 w5)      [10] [20] [0] [10]
ACT: 156.7:  (navigate madbot w5 w3)      [10] [20] [0] [10]
ACT: 176.8:  (navigate madbot w3 w8)      [10] [20] [0] [10]
ACT: 196.9:  (take-image madbot w8 o3 c1) [ 5] [20] [5] [ 0]
```

Figure 4: The initial plan output by ADAPT_LPG.

```
ACT:   0.1:  (take-panoramic-image madbot) [ 5] [30] [5] [ 0]
ACT:  30.2:  (calibrate madbot c1 o2 w4)   [ 1] [20] [0] [ 0]
ACT:  50.3:  (navigate madbot w4 w5)       [10] [20] [0] [10]
ACT:  70.4:  (navigate madbot w5 w3)       [10] [20] [0] [10]
ACT:  90.5:  (navigate madbot w3 w8)       [10] [20] [0] [10]
ACT: 110.6:  (take-image madbot w8 o3 c1)  [ 5] [20] [5] [ 0]
```

Figure 5: The second plan generated by ADAPT_LPG.

cates the amount of battery charge consumed by the action (this affects the motivation *conserve-energy*), e.g. the action *(navigate madbot w1 w2)* consumes 8 units of charge; the second number indicates the duration associated with the action (this affects the motivation *acquire-data*), e.g. the action *(navigate madbot w1 w2)* takes 16 units of time to execute; the third indicates the amount of storage space required (this affects the motivation *communicate-data*), e.g. the action *(navigate madbot w1 w2)* does not require any storage space; the fourth indicates the distance travelled as a consequence of executing the action (this affects the motivation *relocalise*). The *Plan Dispatch* component sends each action of this plan in turn to the *Execution Monitoring* component in order that they can be executed by the rover. As each action is executed the state variables representing the rover will change, which in turn will cause changes to the motivations and when certain constraints are satisfied such changes to the motivations will trigger the generation of goals. In our example, the first goal to be generated involves obtaining a panoramic image, *(have-panoramic-image madbot)*. As stated earlier, the motivational value of *acquire-data* is determined by value of the state variable which records the amount of time which has passed since the rover last obtained a panoramic image. This value is initially set to 0, while the motivation *acquire-data* has a threshold value of 100. This means that after the rover has finished executing the action *(take-image madbot w4 o2 c1)* of the plan in figure 4, the motivational value will exceed the threshold (the motivational value is directly related to the value 0+16+20+20+20+20+20 = 116) causing a goal to obtain a panoramic image to be generated.

At this point, the newly generated goal *(have-panoramic-image madbot)* will be sent by the *Controller* along with the current state (the rover is now at *waypoint 4*) to the *Planner* which will adapt the remainder of the rover's current plan in order to achieve the new goal. This results in the new plan shown in figure 5. Again, each action of this new plan is sent in turn to the *Execution Monitor* so that it can be executed by the rover. Once the rover has finished executing the first action of the new plan, *(take-panoramic-image madbot)*, the goal triggered by the motivation *acquire-data* is satisfied. As a consequence, the value of the state variable which records the amount of time that has passed since the rover last acquired new data is reset to the value 0. The rover continues executing the remaining actions of figure 5, and after it has finished executing the action *(navigate madbot w5 w3)*, the motivational value of *conserve-energy* (which is directly related to the state variable measuring the level of battery charge) will have fallen below its associated threshold which the value 40 (the battery charge level has an initial level of 100, so the current charge will be 100-(8+10+10+10+1+5+5+1+10+10)=30). This will cause a goal to recharge to be triggered, *(recharged madbot)*, which is presented to the *Planner* via the *Controller* and *Goal Arbitration* components. The planner will generate a new plan to achieve this goal, which will be passed to the rover to execute. This process continues—once the rover has travelled a distance which exceeds the threshold value associated with the motivation *relocalise*, (the value 60), a goal to relocalise, *(relocalised madbot)*, will be generated, and once the amount of free data storage space falls below the threshold associated with *communicate-data* (the value 10), a goal to communicate image data to the lander, *(communicated-image-data madbot lander)*, will be generated.

This process will continue indefinitely because even when both high-level goals have been achieved, i.e. *(have-image madbot o2)* and *(have-image madbot o3)*, the MADbot architecture reverts to motivated goal generation, so periodically, whenever their associated thresholds are exceeded or fallen below, the motivations will generate goals to obtain panoramic images *(have-panoramic-image madbot)*, to recharge *(recharged madbot)*, to relocalise *(relocalised madbot)* or to communicate image data to the lander *(communicated-image-data madbot lander)*. It should be noted that in deciding which goals to address, and how to incorporate these into the plan, the top level goals of the original plan under execution remain paramount and that the pressing need to respond to a motivated goal will temporarily suppress the plan and allow an originally unplanned for task to take priority.

Modelling Motivations as Resources

In our second model of goal generation, we still use the MADbot framework, but we model the motivation machinery internally within the domain description used by the planner. This means that it is the planner's responsibility to ensure the motivational values remain within certain bounds.

As stated previously, the four motivations *conserve-energy*, *acquire-data*, *communicate-data* and *relocalise* are directly related to the value of their associated state variables (the level of battery charge, the amount of time which has passed since the rover last obtained a panoramic image, the amount of free data storage, and the distance travelled since the rover last recalibrated), which change as the rover acts within its environment. Because it is possible to predict the effect each action will have on the values of these state variables, it is possible to model these motivations implicitly by modelling their associated state variables and their asso-

ciated thresholds in the domain model used by the planner.

This means that when the planner generates a plan it is the planner's responsibility to ensure the motivations remain within their bounds. Figure 6 shows how the *navigate* action is modelled. We can see in this figure that executing the *navigate* action will cause the battery charge to decrease, the amount of time which has passed since the rover last obtained a panoramic image *(time_passed ?x)* to increase, and the amount of distance the rover has travelled since it last relocalised *(distance_travelled ?x)* to increase. The preconditions of the action ensure that the motivational values do not exceed or fall below their associated thresholds. For example, in order to execute this action, the level of battery charge after the action has been executed (we predict this level by subtracting the amount of charge that will be consumed when executing this action from the amount of charge prior to execution), must be greater than the threshold associated with the motivation *conserve-energy*, i.e. the value *(bc_threshold ?x)*. In addition, the amount of time that will have passed (after executing this action) since the rover last obtained a panoramic image, must be less than the threshold associated with the motivation *acquire-data*, i.e. the value *(panorama_threshold ?x)*. Finally, the amount of distance travelled (after executing this action) since the rover last relocalised, must be less than the threshold associated with the motivation *relocalise*, i.e. the value *(reloc_threshold ?x)*. The actions *recharge*, *take-panoramic-image*, *relocalise* and *communicate-image-data* are added by the planner to the plan to ensure that the motivational values stay within certain bounds. In figure 6 we show the action *recharge* which resets the level of battery charge to the value 100. This action also affects the motivation *acquire-data* as it takes time to execute and so increases the amount of time that has passed since the rover last obtained a panoramic image.

```
(:durative-action navigate
 :parameters (?x - rover ?y - waypoint ?z - waypoint)
 :duration (= ?duration (/ (distance ?y ?z) (speed ?x)))
 :condition (and (at start (can_traverse ?x ?y ?z))
                 (at start (visible ?y ?z))
                 (at start (available ?x))
                 (at start (at ?x ?y))
                 (at start (<= (bc_threshold ?x)
                               (- (battery_charge ?x)
                                  (power_consumed ?y ?z))))
                 (at start (>= (panorama_threshold ?x)
                               (+ (time_passed ?x)
                                  (/ (distance ?y ?z)
                                     (speed ?x)))))
                 (at start (>= (reloc_threshold ?x)
                               (+ (distance_travelled ?x)
                                  (distance ?y ?z)))))
 :effect (and (at start (not (available ?x)))
              (at start (not (at ?x ?y)))
              (at end (available ?x))
              (at end (at ?x ?z))
              (at end (decrease (battery_charge ?x)
                                (power_consumed ?y ?z)))
              (at end (increase (time_passed ?x)
                                (/ (distance ?y ?z) (speed ?x))))
              (at end (increase (distance_travelled ?x)
                                (distance ?y ?z)))
         ))

(:durative-action recharge
 :parameters (?r - rover ?x - waypoint)
 :duration (= ?duration (recharge_time))
 :condition (and (at start (recharge_point ?x))
                 (over all (at ?r ?x))
                 (at start (available ?r))
                 (at start (>= (panorama_threshold ?r)
                               (+ (time_passed ?r)
                                  (recharge_time)))))
 :effect (and (at start (not (available ?r)))
              (at end (available ?r))
              (at end (assign (battery_charge ?r) 100))
              (at end (increase (recharge_count) 1))
              (at end (increase (time_passed ?r) (recharge_time)))
         ))
```

Figure 6: The *navigate* and *recharge* actions which modelling the motivations as resources.

In this model of motivated goal generation, the motivations and goals that they generate are implicit, such goals are generated internally by the planner as precondition goals to be satisfied as part of the planning process. This means that we do not require explicit models of the motivations *conserve-energy*, *acquire-data*, *communicate-data* and *relocalise*, although we can still have explicit models of the motivations *safety*, and *replan*. The motivations *safety* and *replan* are slightly different in that their associated state variables have values which cannot be predicted as part of the planning process as they depend upon the outcome of execution and the length of the current plan respectively. In this model goals may still be imposed to the system by an external source, although the external source is the only means by which goals are inserted to the system. This means that in the absence of externally imposed goals the system has no means for operating autonomously.

Results and Discussion

In the previous sections we described two ways of modelling motivations and the way they influence goal generation, both of which has various advantages and disadvantages. The first approach has the advantage of being able to direct its own behaviour by explicitly generating goals in response to changes to its motivations in the absence of externally imposed high-level goals. This means that for an autonomous system such as a Mars rover system which only has limited opportunity to communicate with ground control, the rover will be able to carry on performing useful tasks during periods when it is impossible for ground control to send up further instructions.

However, this approach has a few problems which need to be overcome. One of the main problems of this approach is that by decoupling resource consumption from planning, it is possible for the system to generate plans which violate resource constraints. For example, it is possible for the rover to run out of battery charge, as the *Planner* has no model of battery charge consumption. The idea of motivated goal generation is that as soon as a motivational value exceeds its threshold a goal will be generated, the planner will generate a plan to achieve that goal and then execute that plan. This causes another problem to arise—an indeterminate time period occurs between generating a goal and achieving that goal and during this period the relevant resource might run out. For example when the level of battery charge causes the motivation *conserve-energy* to exceed its threshold and generate a goal to recharge, a gap occurs before this goal is achieved during which the rover may run out of charge. This problem is exacerbated by the Planner ADAPT_LPG which often generates a new plan in which the

newly generated goal is only achieved at the end of that plan, for example, instead of inserting the action *(take-panoramic-image madbot)* at the beginning of the plan as shown in figure 5, ADAPT_LPGP often chooses to achieve the goal at the end of the plan. This problem is further exacerbated by ADAPT_LPG which is an anytime planner and often generates inefficient plans containing cycles, e.g. *(navigate madbot w3 w5), (navigate madbot w5 w3), (navigate madbot w3 w8)*. The two solutions to these problems which require further investigation involve: experimenting with the threshold values assigned to each motivation—if we make these values very conservative we may be able to reduce or eliminate resource violations; ensuring that the planner achieves newly generated motivated goals as soon as possible rather than allowing it to achieve them at the end of the plan.

The second approach in which motivations are modelled implicitly in the planning domain model overcomes the problem with resource violation as resource consumption is modelled explicitly. We ran an encoding of the Mars rover domain on three planners, Metric-FF[1], LPG-td[2] and SGPlan[3], in which the state variables associated with the motivations *conserve-energy*, *acquire-data*, *communicate-data* and *relocalise* were modelled as resources in each planner's domain model along with their associated thresholds. Because Metric-FF is unable to model durative actions we ran it using an equivalent domain model containing instantaneous actions. Of the three planners however, Metric-FF and LPG-td were unable to find plans for our problem (Metric-FF simply reported that no plan could be found while LPG-td timed out after 30 minutes), whereas SGPlan produced a non-optimal plan containing redundant actions. However, the resulting plan produced by SGPlan ensured that no resource violations occurred. Further investigation revealed that Metric-FF and LPG-td were able to quickly produce plans in which only one or two of the motivations were modelled in the planner's domain model as resources (e.g. a domain model in which the motivation *conserve-energy* is modelled in terms of its associated resource and threshold while the remaining motivations were modelled explicitly).

An important lesson we learned when modelling motivations as resources in the planner's domain model was that the planner's ability to successfully generate a plan depends upon the relationship between the layout of the environment, in particular the number and location of resource-replenishment opportunities, and either the level to which the resource is replenished, or the threshold levels assigned to each motivation. For example, in figure 2 we have two recharge points located at *waypoint 1* and *waypoint 5*. If we remove the recharge point at *waypoint 5* without altering the amount by which battery charge is replenished (recharging a battery currently assigns 100 units of charge to the battery), SGPlan is unable to successfully find a plan to solve the problem. This means if we remove the recharge point at *waypoint 5* we must increase the amount by which battery charge is replenished. Likewise, for the motivation *relocalise*, the number and location of relocalisation points, together with the relocalisation threshold value *(reloc_threshold ?x)* will determine whether the planner can successfully find a plan, so fewer relocalisation points requires a higher relocalisation threshold.

When we closely examine the nature of the motivations identified previously we observe that some motivations are more resource-critical than others. For example, *conserve-energy* is the most resource-critical in that if the battery charge runs out, the rover is stuck and unable to recover. The motivation *relocalise* is slightly less resource-critical, although if the rover does not periodically relocalise, it will lose its sense of where it is and so be unable to act effectively. Even less resource-critical is the motivation *communicate-data* because if the rover does not communicate the image data it has collected, thereby freeing up data storage, it may simply over-write the image data it has already collected. Although interesting data might be lost, this will not affect the rover's ability to perform effectively. Finally, the motivation *acquire-data* is the least resource-critical as it depends solely on the amount of time that has passed since a panoramic image was last obtained. Waiting slightly longer to obtain a panoramic image will not affect the rover's ability to act effectively.

With this in mind, another solution is to combine the two approaches described earlier. In this hybrid approach, those motivations which are resource-critical (e.g. *conserve-energy* and *relocalise* are modelled implicitly as resources together with appropriate threshold values so that the planner can ensure that the relevant resource constraints are not violated, while those motivations that are not resource-critical (e.g. *acquire-data* and *communicate-data*) are modelled explicitly as motivations which monitor the value of their associated state variables, and which generate goals whenever their motivational values satisfy the appropriate constraints. In this approach, the system is still able to operate under motivated control, even in the absence of externally imposed goals.

Further Work

The work described in this paper is ongoing—the MADbot framework provides a platform for a number of areas for future work which we describe in this section.

Currently, goals are generated whenever the value of their associated motivation(s) falls below or exceeds some fixed threshold value. One area for further investigation involves examining how the threshold value for each motivation can be dynamically adjusted by taking into account changes to the context. For example, as the "cognitive load" on the agent increases (this is indicated by the number of goals it is currently attempting to achieve as well as the length of the current plan), the introduction of new goals might either be encouraged or suppressed by lowering or raising the thresholds associated with the motivations. For example, if the agent is currently idle, or has only a small number of goals and a short plan, lowering the thresholds will encourage more new goals to be generated.

An important component of the MADbot system involves

[1]http://members.deri.at/ joergh/metric-ff.html

[2]http://zeus.ing.unibs.it/lpg/

[3]http://manip.crhc.uiuc.edu/programs/SGPlan/

generating goals in response to changes to its motivational values. In the Mars rover domain, the motivations currently generate single goals, for example once the battery charge has fallen below a certain threshold, a goal to recharge is generated. In some situations however, it may be appropriate to generate disjunctive goals. For example, when the rover is motivated to obtain an image, (either a panoramic image, or an image of some objective), the resulting images are currently stored on a single data storage device, and, when the device gets full, the rover is further motivated to send the image data to a lander. If the rover has n storage devices where $n > 1$, a disjunctive goal could be generated such as *(or (have-image* $image_1$ $device_1$*) (have-image* $image_1$ $device_2$*) .. (have-image* $image_1$ $device_n$*))*.

Another area for further investigation is that of goal arbitration which is the problem of determining, amongst a set of competing goals, which to attempt to solve. This problem, known as the over-subscription problem (Smith 2004; Sanchez & Kambhampati 2005), is important because there may not be sufficient time or resources available to achieve all of the goals that have been generated. Goals may have very different costs in terms of resource usage or they may be mutually inconsistent, either because they imply directly contradictory states for the system or else because they demand more resources than are available. In addition, if motivations cause disjunctive goals to be generated it will be necessary to choose which goal disjunct to achieve. Although achieving each goal disjunct might satisfy the motivation to the same degree, the costs of achieving each goal disjunct might differ widely. (Coddington & Luck 2004) present a model in which the values associated with each motivation could in some way determine the relative priorities of the goals generated in response to that motivation which means that at any moment, the set of motivations could cause a number of goals with different priorities to be generated. The intended role of the *Goal Arbitration* component of figure 1 is to select which of these goals will be achieved.

Conclusions

In this paper we have introduced the idea of motivations and described how changes to an agent's motivations might cause new goals to be generated. The reason for introducing motivations into our architecture is to provide a solution to the problem of where goals come from, and to thereby improve the effectiveness of remote autonomous systems such as a Mars rover system in which the system is only able to operate by communicating with ground control. We presented two approaches to motivated goal generation. The first approach explicitly models the agent's motivations which change in value in response to changes that occur to the agent's state variables, and cause goals to be generated whenever such changes satisfy certain constraints. The second approach involves modelling the motivations implicitly by encoding them as resource constraints in the planner's domain model—in this approach it is the planner's responsibility to ensure that the implicit motivational values remain within predefined bounds. We described the advantages and disadvantages of both approaches and proposed a hybrid model in which resource-critical motivations are modelled implicitly as resource constraints in the planner's domain model, and in which the remaining non-resource-critical motivations are modelled explicitly in the *Motivations* component of the MADbot architecture. We believe the MADbot architecture to be a promising platform for further work in the areas of goal generation and goal arbitration.

Acknowledgements

The MADbot architecture was developed with Derek Long, Jonathan Gough (University of Strathclyde) and Ivan Serina (University of Brescia).

References

Balkenius, C. 1993. The roots of motivation. In Meyer, J. A. Roitblat, H. L., and Wilson, S. W., eds., *From Animals to Animats II*. MIT Press.

Chien, S.; Engelhardt, B.; Knight, R.; Rabideau, G.; Sherwood, R.; Hansen, E.; Ortiviz, A.; Wilklow, C.; and Wichman, S. 2001. Onboard autonomy on the three corner sat mission. In *International Symposium on AI, Robotics and Automation in Space*.

Coddington, A. M., and Luck, M. 2004. A motivation-based planning and execution framework. *International Journal on Artificial Intelligence Tools* 13(1):5–25.

Coddington, A.; Fox, M.; Gough, J.; Long, D.; and Serina, I. 2005. MADbot: A Motivated And goal Directed robot. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*.

Ferber, J. 1999. *Multi-Agent Systems: An Introduction to Artificial Intelligence*. Addison Wesley Longman.

Gerevini, A., and Serina, I. 2000. Fast plan adaptation through planning graphs: Local and systematic search techniques. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the 5th Int. Conf. on AI Planning and Scheduling*, 112–121. AAAI Press.

Knight, R.; Rabideau, G.; Chien, S.; Englehardt, B.; and Sherwood, R. 2001. Casper: Space exploration through continuous planning. *IEEE Intelligent Systems* 16(5).

Kunda, Z. 1990. The case for motivated reasoning. *Psychological Bulletin* 108(3):480–498.

Maslow, A. 1971. *The farther reaches of human nature*. New York: Penguin Books.

Moffat, D., and Frijda, N. H. 1995. Where there's a will there's an agent. In Wooldridge, M., and Jennings, N. R., eds., *Intelligent Agents, ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, 245–260. Springer.

Sanchez, R., and Kambhampati, S. 2005. Planning graph heuristics for selecting objectives in over-subscription planning problems. In *Proceedings of the 15th Int. Conf. on AI Planning and Scheduling*, 192–201.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the 14th Int. Conf. on Automated Planning and Scheduling*, 393–401.