Adaptive Resource Modelling for Autonomous Planning and Scheduling

Andrew Carrel and **Phil Palmer**
Surrey Space Centre
University of Surrey
Guildford GU2 7XH
United Kingdom

Abstract

Autonomous planning and scheduling requires models of how spacecraft operations will affect resources. In practice the true behaviour may deviate from the model, either due to modelling inaccuracies or a change in behaviour. In this paper a neural network is used to learn about such modelling errors and provide corrections that can be used for forward planning. This enables adaptation to unexpected changes in the way resources are consumed, such as may occur when a subsytem's performance becomes degraded.

This learning method has been incorporated into the NEAT autonomous planner to enable it to adapt its resource models. Comparisons of NEAT's resource management performance with and without this new component have shown that using a neural network in this way greatly improves the robustness of autonomous operations when faults develop.

Introduction

The NEAT algorithm has previously been presented (Carrel & Palmer 2005) as an effective approach to onboard autonomy which could produce significant performance improvements over more conventional spacecraft operations. This Near-optimal, Evolutionary, Autonomous Taskmanager (NEAT) optimises its planned schedule of operations using an evolutionary algorithm, whilst enforcing temporal and resource constraints. Operations and resources are defined using a generic format such that different types of operation can be scheduled and multiple resources can be managed. This will make it possible to apply NEAT to a wide range of applications.

NEAT works on an entirely iterative basis, with the operations schedule evolving towards the optimum continuously. When a decision is to be made the best solution in the current population is used. Since the evolutionary process is continuous, NEAT is able to adapt to spurious events such as a failed operation or an unexpected loss of resource. Previous results have shown this method to be effective at repeatedly re-optimising the operations schedule in the presence of such events (Carrel & Palmer 2005).

To be able to manage future resource levels, any planning algorithm must have models describing the expected resource use of each operation and the availability of these resources at future times. In practice however, these models will generally be imperfect, either due to simplifying assumptions of inherent unpredictability. For this reason NEAT updates its estimates of resource levels as measurements become available and adapts its plans as necessary. However, this feedback does not learn about systematic modelling errors such as may occur when the spacecraft changes its behaviour. For example, if a solar array is degraded by micro-meteorite impacts, the available electrical power will be systematically less than the model may predict.

In this work the NEAT algorithm is augmented with a neural network which is able to learn about systematic changes in the availability or consumption of resources. Measured resource levels are compared with those predicted during planning to find systematic modelling errors (see figure 1). The neural network is then trained to predict these errors such that corrections can be applied for future planning. The training data is continuously updated such that the network adapts with changing resource behaviour.

This system has been applied to the case of scheduling imaging operations on the UK-DMC Earth observation satellite. Faults were simulated to create changes in the way operations affect resource levels. The neural network was then required to adapt so that NEAT could continue to make effective plans autonomously.

Generic Resources and Activities

A spacecraft may have many different resources, some of which are tangible such as fuel, and some of which are related to its circumstances, such as the availability of a ground station. For it to be possible to create a generic resource management system it is first necessary to divide resources into well-defined categories.

Two resource categories are used here:

Depletable resources are those that remain depleted even after an activity has finished using them. Examples include fuel and electrical energy. In some cases, such as electrical energy, it may be possible for an activity to replenish rather than consume the resource. Only a finite quantity of this type of resource will be available at any one time and this quantity will never exceed a fixed upper limit, for example a vehicle's battery capacity in the

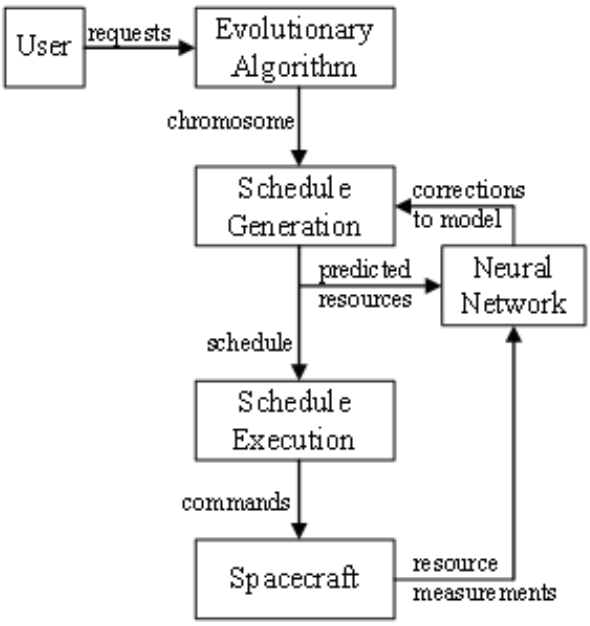

Figure 1: Scheme for adapting to changing resource behaviour. Measured resource levels are compared with predicted levels to create resource model corrections.

case of electrical energy. Depletable resources are depleted and replenished at finite rates over finite time intervals such that there are no step changes in the available quantity of the resource.

Non-depletable resources are only depleted for the period in which they are in use, after which they return to their original level. An example is computer processing power. Some activities may make more of this type of resource available for their duration rather than using the resource. The availability of this type of resource must not become negative and has fixed maximum limit. The effect of an activity using this type of resource will be a step change in the resource level at the start of the activity, which is reversed when the activity is complete.

These definitions are similar to those used in the the ASPEN system (Fukunaga *et al.* 1997; Chien *et al.* 2000). In that work the two are differentiated by the fact that a depletable resource "remains used even after the activity" (Chien *et al.* 2000), whilst "a non-depletable resource is used only for the duration of the activity". In a separate work, Brambilla et al. (Brambilla *et al.* 2005) also refer to these two resource types and define them in much the same way. Similarly, Funase & Nakasuka (Funase & Nakasuka 2005) consider "consumable" and "reusable" resources. Lemai & Ingrand (Lemai & Ingrand 2004) do not categorise resources but rather allow actions to "use, consume and produce" resources. When a resource is 'used' it is borrowed over an interval, producing a similar behaviour to a non-depletable resource. Consumption of a resource has an effect after the action is complete, similar to the effect generated by using a depletable resource.

Whilst the definitions of depletable and non-depletable resources given here are broadly the same as in other works, one difference is worth noting. Elsewhere it is assumed for planning purposes that an activity's depletable resource requirements are completely used from the instant the activity begins. In this work, however, activities consume (or produce) depletable resources at a defined rate over their duration. This approach is particularly useful when a depletable resource is being consumed or generated at a low rate over a relatively long period since the expected availability of the resource will be much more accurate.

In a later version of ASPEN (Sherwood & et al. 1999) two further resource types are included: atomic and concurrency. In both cases these resources may only be used by one activity at a time, the difference is simply that a concurrency resource is only available when it is explicitly made available whereas an atomic resource is available by default. In the MissionSwap system (Kramer & Smith 2004) another resource type is introduced. Here each resource has an integer value with each activity using one resource unit. Activities only deplete the level of resource availability for their duration, after which the unit becomes available again.

These additional resource types are in fact special cases of non-depletable resources where the minimum resource level is zero and activities always use the same fraction (or all) of the resource. Therefore, whilst these are useful concepts, it is sufficient to consider only depletable and non-depletable resources in creating a generic resource management system.

Both depletable and non-depletable resource have minimum and maximum levels. It is therefore straightforward to rescale each resource such that its minimum and maximum levels are zero and one respectively. This is very useful when dealing with generic resources since now the current state of any resource can be represented by a value in the range zero to one.

An activity is considered to be any action taken by the spacecraft that partly or completely achieves a goal. An operation is then simply the group of all the activities that relate to a goal the spacecraft has been set. Operations are mutually exclusive sets of activities. When all the constituent activities of such a set are successfully completed then the relevant goal will have been achieved.

To create an operation request (i.e. set the spacecraft a goal) it is first necessary to list all the activities that will be required. With each of these a finite set of one or more time windows must be given that specifiy when the activity may be scheduled. Activities may also have temporal constraints that are relative to other activities within the same operation (see table 1). For example image data must be transmitted to a ground station after the image has been captured and stored. If activity A has a relative temporal constraint dependent on when activity B occurs then it is not possible to know the absolute temporal constraints on activity A until a time has been set for activity B. Therefore activity B is said to be *superior* to activity A and an operation's activities are ordered from most superior to most inferior. If an activity's duration is not determined by relative constraints then it is

Constraint	Description
$S_I > S_S$	Start after the superior activity starts
$S_I > F_S$	Start after the superior activity finishes
$F_I > S_S$	Finish after the superior activity starts
$F_I > F_S$	Finish after the superior activity finishes
$S_I < S_S$	Start before the superior activity starts
$S_I < F_S$	Start before the superior activity finishes
$F_I < S_S$	Finish before the superior activity starts
$F_I < F_S$	Finish before the superior activity finishes
$S_I = S_S$	Start as the superior activity starts
$S_I = F_S$	Start as the superior activity finishes
$F_I = S_S$	Finish as the superior activity starts
$F_I = F_S$	Finish as the superior activity finishes

Table 1: Possible relative temporal constraints between the start or finish times of an inferior activity (S_I,F_I) and the start and finish times of a superior activity (S_S,F_S).

specified explicitly.

For each activity the required amount of each of the spacecraft's resources (or the amount generated) is specified. For depletable resources this will be the rate at which the resource will be consumed (or produced). Some activities may have to be included to account for essential or unavoidable changes in resource availability. For example charging of batteries using solar arrays is generally essential and some power consumption is usually unavoidable. These activities may not relate directly to high-level goals and so are goals in themselves. Operation requests must be created for individual or groups of activities of this kind so that forward planning is done in light of all expected resource changes.

Each goal is given a priority based on its perceived importance. Firstly goals are split into two groups, essential and non-essential. Essential goals relate to those activities just described that either must be performed to maintain the spacecraft's health or have unavoidable impacts on resources. All of the operations relating to these goals will be scheduled with equal priority. This should be possible without conflict, otherwise there is something wrong with the spacecraft. All other, non-essential goals are given a user defined priority, which is taken into account when optimising the schedule.

Schedule Generation and Optimisation

The continuous evolutionary optimisation of the operations schedule enables the spacecraft to be highly adaptive, however it is important to ensure that all the temporal and resource constraints are satisfied. Also the repeated projection of different planning options requires that schedules can be generated quickly. All the essential operations are used to create a baseline schedule. The optimisation component then searches the different orders in which the non-essential operations may be condidered for placement in the schedule. Each operation consists of a set of activities, which are considered for placement in the schedule in order of superiority, as defined earlier. These orderings do *not* mean that activities must be scheduled in a set time order.

An operations schedule is represented by a set of time-ordered events, representing the start and end of each activity. Resource availability and depletable resource rates are stored with each event. Depletable resource levels can then be projected directly from one event to the next without fixed time-step numerical integration to create a profile of the expected future resource availabilty (see figure 2).

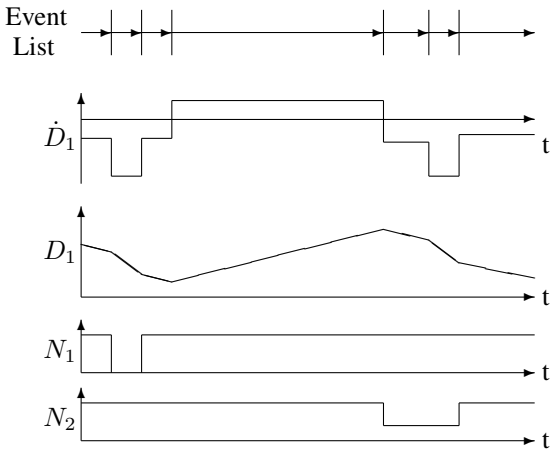

Figure 2: Projection of one depletable resource (D_1) and two non-depletable resources (N_1 & N_2) using schedule events.

The process of placing an operation in the schedule is based on backtracking. This is designed to ensure that all options have been exhausted before an operation is rejected. This does not, however, go as far as considering moving any activities relating to operations that have already been placed in the schedule, since these options are dealt with by the evolutionary optimisation.

Scheduling an operation simply requires finding a place in the schedule for each of its constituent activities, $a = 1 \ldots N_a$, such that all temporal and resource constraints are satisfied. The method for working through the activities is shown in figure 3. The process starts with the first activity ($a = 1$), applying the function $S(a)$, which attempts to place it in the schedule. If this is successful then the algorithm moves on to the next activity and repeats this scheduling until either an activity cannot be placed or all the activities are scheduled and so the process is complete. This is the forward loop marked 'F' in the figure.

If the function $S(a)$ fails to place an activity then the algorithm moves back to the previous activity, unless this is already the first activity ($a = 1$). In the latter case, the process terminates having failed either because it was not ever possible to place the first activity or after an exhaustive search in which many possible times were considered but in each case one of the other activities could not be scheduled. In the case where $S(a)$ fails and $a > 1$ the algorithm backtracks and removes the superior activity $a - 1$ using the function R. This activity is then re-scheduled by the function S, which works in such a way that the activity will be re-scheduled at a later time than before, if possible. This

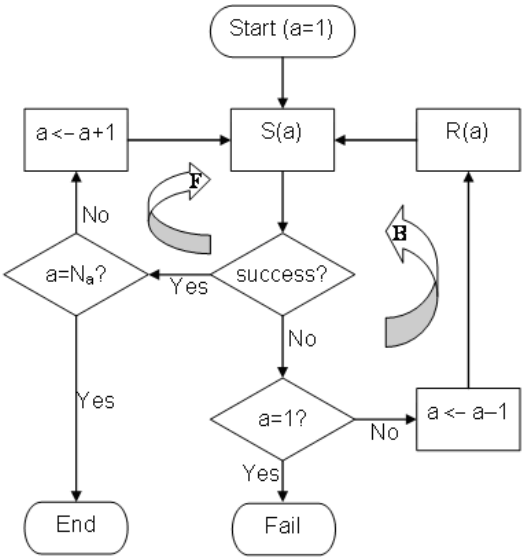

Figure 3: Scheduling an operation

backtracking loop is marked 'B' in figure 3.

The function $S(a)$, that schedules an activity, employs a three-stage backtracking process considering temporal, non-depletable resource and depletable resource constraints in turn (see figure 4). The function maintains a time window, $[t_E, t_L]$, which is the earliest known period when all the constraints considered so far are satisfied. At each stage the function checks that this is at least as long as the activity duration before proceeding to the next step. If it is not long enough then the function backtracks to the previous stage and looks for a later opportunity.

This schedule generation process is used to find the fitness of each of a population of orderings of the non-essential operations. Each of these permutations is a chromosome in the population of an evolutionary algorithm. The definition of the fitness of a chromosome can be set to whatever is appropriate to the application but will generally depend on the expected time taken for operations to be completed with greater weight being placed on higher priority goals.

In each planning cycle the evolution process is incremented G generations. The chromosome with the greatest fitness in the resulting population is then used to produce the working schedule for the next cycle. In this way the population will continually evolve towards producing the optimal allocation of resources but a valid schedule is always available.

Each generation the parent strings are selected for breeding using the 'roulette wheel' method (Goldberg 1989). Selected parents are then crossed with probability p_c. The permutation encoding of chromosomes requires a specialist crossover operator; the PMX crossover method (Goldberg & Lingle 1985) is used here. Mutations are applied to the child chromosomes with probability p_m using the inversion operator (Lin 1965).

When a new request arrives an additional gene is inserted

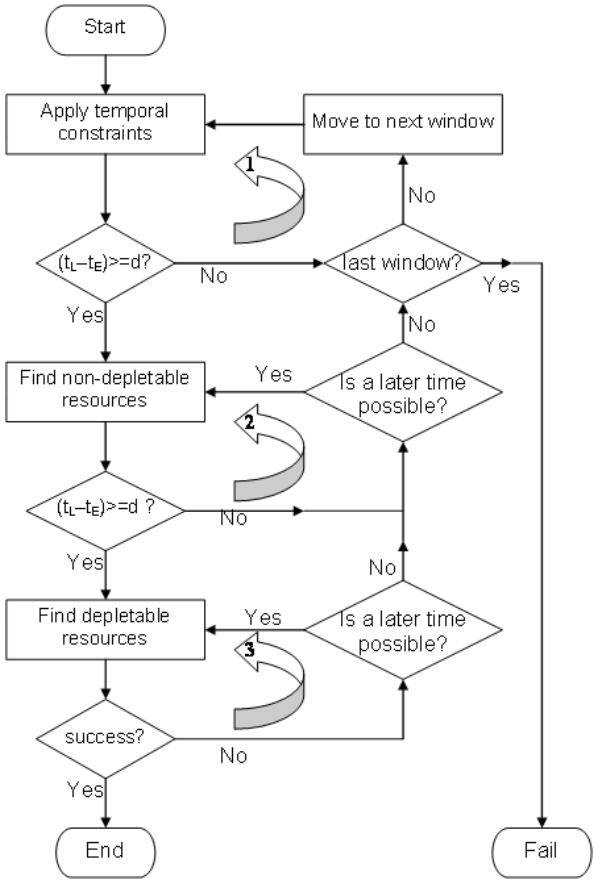

Figure 4: Scheduling an activity

at a random location in each chomosome in the current population. Similarly when a requested operation has been completed the relevant gene is removed from each string. In this way the ordering of other requests in the population is unaffected. Initially each chromosome in the population has length zero and they build up in length as operation requests arrive.

Neural Network Learning

The neural network used here consists of an input layer of size n_r (the total number of resources), a layer of $n_h = 2n_r$ hidden neurons and a layer of n_r output neurons as shown in figure 5. The network has one input and one output for each resource being managed. For non-depletable resources the inputs are the expected normalised resource levels whereas for depletable resources the nominal rate of consumption or production is used. The outputs of the network are corrections to be applied to the non-depletable resource levels and the depletable resource rates.

Both the inputs and outputs are rescaled to ensure that the network only has to deal with values comfortably in the range zero to one. For non-depletable resources input values originally in the full zero to one range are rescaled to the range 0.1 to 0.9. Depletable resource rates are not con-

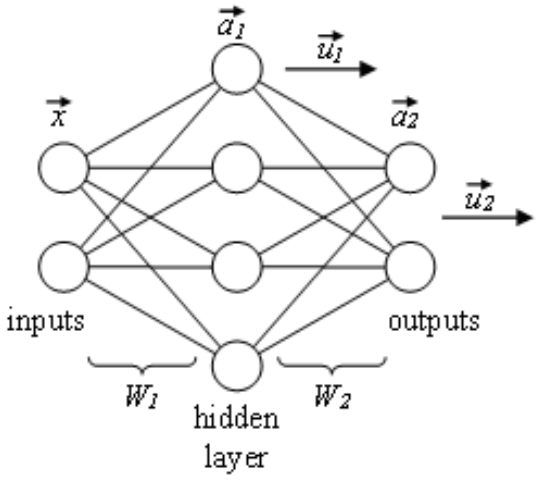

Figure 5: Neural network structure for two resources.

strained to a particular range and so each raw depletable input D is rescaled to x_D using:

$$x_D = \frac{1}{1 + e^{-\sigma D}} \tag{1}$$

The scale factor σ is set to a value appropriate to the problem in question. These scalings are applied in reverse to the network ouputs to get the final correction values.

Calculation of Weights

The hidden neurons are fully connected to the input layer with weight matrix W_1 and the output neurons are fully connected to the outputs $\vec{u}_1$ of the hidden neurons with weight matrix W_2. The activation vectors $\vec{a}_1$ and $\vec{a}_2$ of the hidden and output layers respectively are given by:

$$\vec{a}_1 = W_1 \vec{x} \tag{2}$$
$$\vec{a}_2 = W_2 \vec{u}_1 \tag{3}$$

For each neuron the output is related to the activation by:

$$u = \frac{1}{1 + e^{-a}} \tag{4}$$

This is used to generate the output vectors $\vec{u}_1$ and $\vec{u}_2$ from the activation vectors $\vec{a}_1$ and $\vec{a}_2$.

The network is trained using backpropagation (Rumelhart & et al 1986) to adjust the weights, which are initially set to zero. For the ith set of training inputs $\vec{x}_i$ and outputs $\vec{t}_i$, the incremental adjustments to the weights are given by:

$$\delta W_{1i} = \beta \vec{d}_1 \vec{x}_i^T \tag{5}$$
$$\delta W_{2i} = \beta \vec{d}_2 \vec{u}_1^T \tag{6}$$

where:

$$\vec{d}_1 = W_2^T \vec{d}_2 \otimes \vec{u}_1 \otimes (\vec{1} - \vec{u}_1) \tag{7}$$
$$\vec{d}_2 = (\vec{t}_i - \vec{u}_2) \otimes \vec{u}_2 \otimes (\vec{1} - \vec{u}_2) \tag{8}$$

Here $\vec{1}$ refers to a vector where each element equals 1 and the symbol $\otimes$ refers to element by element multiplication of two vectors to get a third vector of the same length. The parameter β is the learning rate. The weight adjustments are summed over the whole set of training data:

$$\delta W_1 = \sum_{i=1}^{N_{train}} \delta W_{1i} \tag{9}$$

$$\delta W_2 = \sum_{i=1}^{N_{train}} \delta W_{2i} \tag{10}$$

These adjustment matrices are then added to the weight matrices. This process is repeated until these weight changes have been calculated and applied for C_{train} training cycles.

Resource Model Adaptation

Once the neural network has been trained it can be used to modify the expected resource use of activities during the planning process. Each event in the schedule stores the nominal availability of non-depletable resources and rates of consumption or production of depletable resources. These values are passed as the input vector $\vec{x}$ to the neural network such that the output vector $\vec{u}_2$ can then be used to generate corrected values, which are stored alongside the nominal values at each event. In this way the corrected values can be used for projecting resource levels for planning and the nominal values can still be used for comparison with measurements to create training data.

Training the Network

Training data is accumulated each time an event in the schedule is reached. At these points the true resource levels are measured just before the scheduled event is acted upon. For non-depletable resources the difference between the expected and measured levels is recorded. For depletable resources the rate at which the resource was actually used is calculated from the measurement at the last event, the current measurement and the elapsed time. This is compared to the nominal rate expected over that interval and the necessary correction is stored. The corresponding network inputs that are stored are the nominal non-depletable resource levels and depletable resource rates expected over the interval between the two events.

Each time a new piece of training data is added to the training data set, the oldest data element is deleted and the network weights are readjusted. In this way the network is always trained using a recent set of data of a fixed size, N_T. This means that the resource use model used for forward planning is continually adapting in light of recent resource measurements. The size of the training data set will determine how responsive the neural network is to changes in resource behaviour. If the data set is very large then it may take a long time for new measurements to supplant old data. However, if the training data set is too small then the neural network may learn about noise or short term fluctuations and produce spurious corrections.

Test Case: UK-DMC Imaging

SSTL's UK-DMC satellite is a 700km altitude Sun-synchronous Earth observation satellite operated from SSTL

in Guildford, UK. It is one of five satellites that form the Disaster Monitoring Constellation (Chu *et al.* 2000), which provides medium resolution imagary with global coverage and a daily revisit period.

Here NEAT is applied to the UK-DMC case both with and without the neural network learning. Three verions of NEAT are tested: open-loop planning, planning with feedback of measured resource levels and planning with the neural network learning described above.

UK-DMC has three depletable resources that need to be managed: two solid state data recorders (SSDR0 and SSDR1) and the battery state-of-charge (SOC). The capacities of SSDR0 and SSDR1 are 1024 Mbytes and 512 Mbytes respectively. The battery consists of 22 4000mAh NiCd cells in series and has a nominal voltage of 28V. Therefore the capacity is taken to be 403.2 kJ, however its state of charge should not fall below 85% to prevent degradation of performance. A charging efficiency of 92% is used to account for losses in the power system.

Five non-depletable resources are also included, all of which are 'atomic' here in that they are either completely in use or fully available. These are used to represent the camera, antenna, solar arrays, the availability of the ground station and to limit power use. The last of these allows only one GPS, image capture or downlink operation to occur at any one time.

The different operations to be scheduled are listed in table 2 along with the impact they have on each resource. In this case each operation consists of just one activity. Four essential operations represent the routine functions onboard the satellite. Solar power generation is active whenever the spacecraft is in sunlight and the core platform operation encompasses the continuous maintenance activities such as attitude control. 10 minute GPS orbit determination operations are scheduled at 110 minute intervals such that GPS measurements are progressively staggered around the orbit (this is in line with current operations procedures). The non-depletable resource representing ground station availability is initialised as unavailable (i.e. set to zero) and operations are scheduled that provide availability for each visibility window.

Image capture and downlink operations are the two types of non-essential operations. The evolutionary algorithm searches for the optimal way in which to schedule these operations. User requests take the form of a target latitude and longitude, the latitude and longitude extent of the area to be imaged and a time by which the image is to be taken. ImPredict (Mai & Palmer 2001; Wu, Brewer, & Palmer 2001) is used to calculate time windows for image capture and for ground station visibility. This is a fast overpass calculation algorithm, based on the analytical epicyle orbit model (Hashisa & Palmer 2001), developed at Surrey. The memory rates are variable due to the nature of the UK-DMC imaging system. The full width of the field of view is around 51.9°, however this is split between two CCD arrays, one port and one starboard facing. These feed directly into SSDR0 and SSDR1 respectively. Therefore since UK-DMC is a nadir-pointing satellite the proportions of the image that will be stored by the two data recorders will vary from one imaging opportunity to the next.

To provide a realistic test scenario a set of imaging targets has been taken from the imaging history of UK-DMC. The 94 targets imaged by UK-DMC during the month of September 2004 are used as the image request targets for testing NEAT in this case. These target areas are spread widely over the globe and are shown in Fig. 6. The scheduling for this period was performed by ground station staff using the existing Mission Planning System, which also calculates imaging windows using ImPredict. The Mission Planning System automatically enforces operational constraints such as resourse limits but maximising the throughput of the system is conducted manually.

The same 94 image requests were sent to NEAT in a simulation of UK-DMC over the same period, with the orbit set to its state at that time. Requests were sent such that there were always 50 outstanding and should all 94 requests have been sent the simulation started sending the requests again from the top of the list but with a much lower priority ($P = 1$ as oppose to $P = 5$).

For this application the fitness, Q, of a trial schedule in the evolving population is given by:

$$Q = \sum_{i=1}^{L} P_i e^{-\gamma \tau_i / T} \tag{11}$$

where P_i is the priority of the ith image request, τ_i is the time taken to return the ith image, T is the maximum image return time that will be accepted and L is the chromosome length, corresponding to the number of outstanding requests. This function ensures that higher priority operations are given more weight. Also the parameter γ determines how much pressure there is to select earlier imaging times where possible ($\gamma = 4.0$ is used).

When NEAT is used with the neural network the training parameters applied are:

- Learning rate, $\beta = 1.0$
- Number of training cycles per update, $C_{train} = 10$
- Size of training data set, $N_T = 50$

These values have been chosen so that the time taken for the resource models to be corrected will be sufficient for only a small number of imaging operations.

Scenario 1: Damaged Solar Array

In this scenario the power produced by the solar arrays drops suddenly by 22W two days into a 10-day simulation. This could have been caused by an unfortunate micrometeorite impact, radiation damage or component failure. Firstly NEAT is tested without any feedback of resource levels. The resulting progression of the battery state-of-charge is shown in figure 7. At the 2-day event the model of solar power generation that NEAT is using departs from reality and so in this open-loop configuration the battery SOC rapidly falls far below the 85% minimum. For this reason the original NEAT algorithm regularly updates its estimates of resource availability using measurements. With this basic feedback in place the performance is improved (see figure 8) but the battery SOC is still too low much of the time.

Operation	Non-depletable Resources					Depletable Resource Rates		
	Camera	Antenna	Gnd Station	Solar Arrays	Power	Bat. SOC	SSDR0	SSDR1
Solar Power Gen.	0.0	0.0	0.0	-1.0	0.0	+57.3W	0.0	0.0
Platform maint'nce	0.0	0.0	0.0	0.0	0.0	-23.1W	0.0	0.0
GPS orbit det.	0.0	0.0	0.0	0.0	1.0	-5.7W	0.0	0.0
G.S. visibility	0.0	0.0	+1.0	0.0	0.0	0.0W	0.0	0.0
Image capture	-1.0	0.0	0.0	0.0	1.0	-32.0W	≤ 0.0	≤ 0.0
Downlink	0.0	-1.0	-1.0	0.0	1.0	-39.1W	+3.9Mbps	+3.9Mbps

Table 2: Resource requirements for UK-DMC operations. Positive values indicate generation of a resource, whilst negative values indicate consumption. The rate of memory consumption during image capture depends of the geometry with respect to the target.

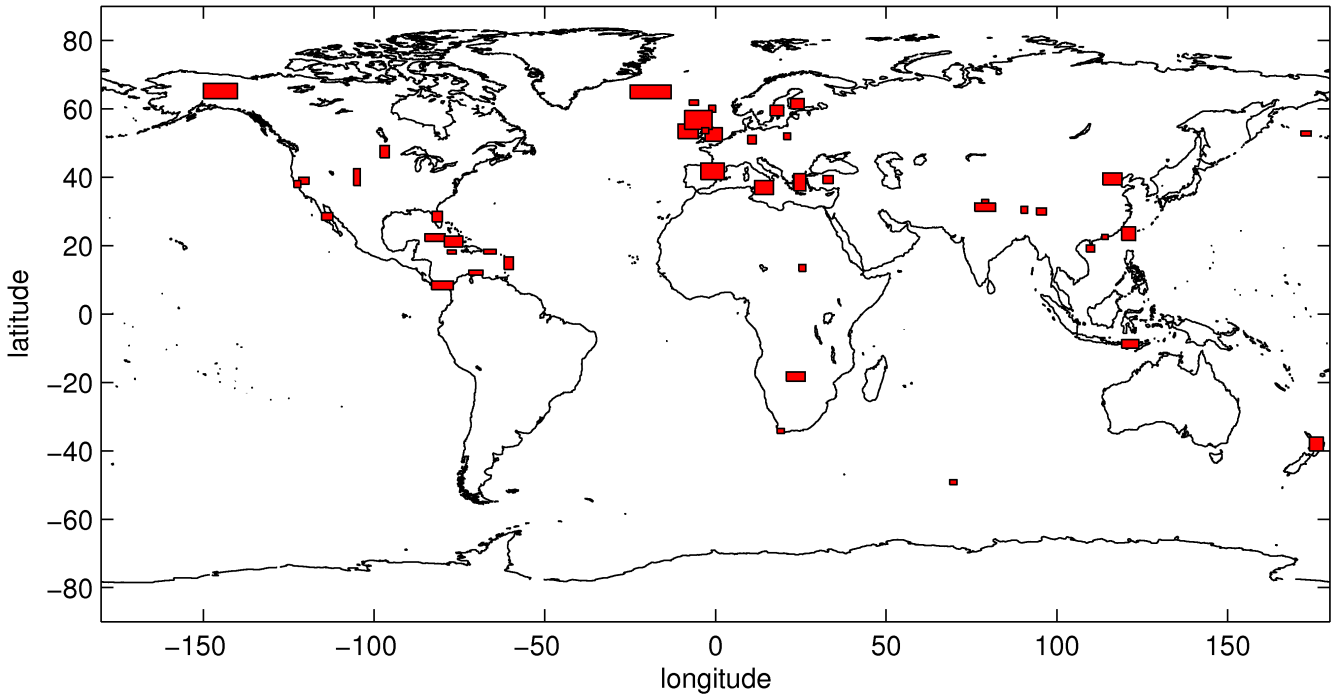

Figure 6: Target image areas. Some of these targets occur more than once in the set of targets to be imaged.

The same scenario was run again but with the neural network learning included. As can be seen from figure 9 the management of the battery SOC is greatly improved, although it still falls slightly below the 85% level for occasional brief periods. Here the neural network has enabled autonomous operations to continue despite a severe drop in the supply of power. Although the small breaches of the 85% battery SOC are undesirable, they are unlikely to cause serious harm, whereas the profile without the neural network would be of great concern.

Scenario 2: Faulty Antenna

Here the effect of a greatly reduced downlink data rate is considered. The same 10-day scenario is used but in this case the downlink data rate is reduced by a factor of 10, due to a simulated fault with the antenna two days into the simulation. With NEAT operating open-loop, without any feedback of resource information, this causes repeated failure of image takes as shown in figure 10. These are caused by there being insufficient memory to store the images. Without any feedback and an incorrect model of the downlink data rate, NEAT inevitably overestimates the available memory and schedules too many images.

When the basic feedback normally used is introduced these failed image takes are greatly reduced because NEAT becomes aware of the lower than expected level of data storage capacity shortly after each downlink. There are still some failures (see figure 11) when an image capture is commanded immediately after a downlink and so before NEAT has received feedback that there is less available memory than expected. Furthermore, NEAT continues to make future plans based on what is now a hugely over-optimistic model of downlink capacity, with imaging operations often being removed from the schedule at the last moment when

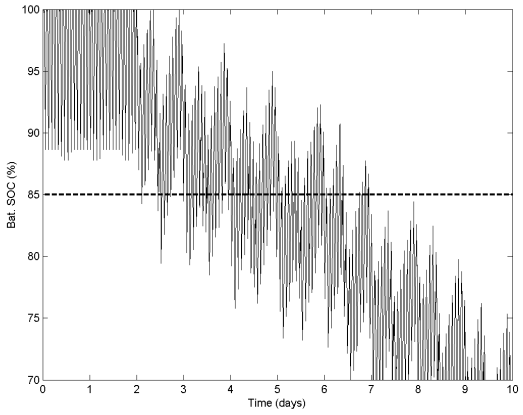

Figure 7: Battery SOC without resource level feedback.

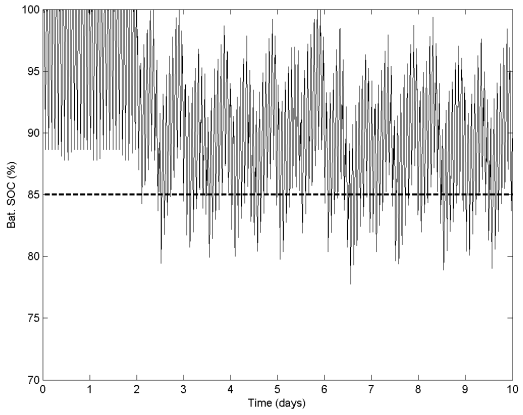

Figure 8: Battery SOC with basic resource feedback.

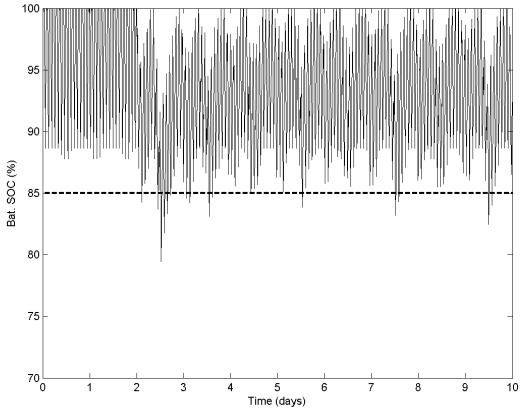

Figure 9: Battery SOC with neural network learning.

feedback data arrives.

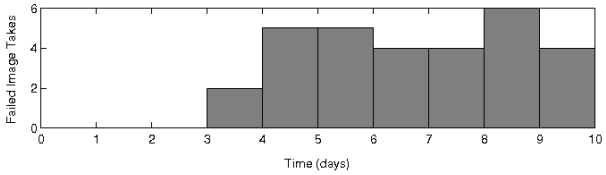

Figure 10: Failed image captures after a fault develops with the antenna when no resource feedback is used.

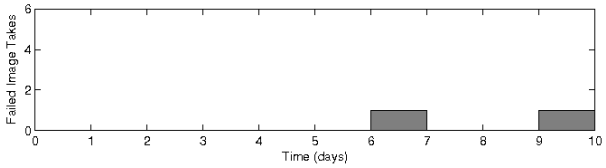

Figure 11: Failed image captures after a fault develops with the antenna and basic resource feedback is used.

When the neural network is included it quickly learns about the reduced downlink data rate, as shown in figure 12. Over the course of a day the old training data is replaced with new data showing the reduced data rate. After this period of adaptation the corrected model of the downlink capacity closely follows the true value. Consequently there are no failed image captures in this test case, although it would not have been surprising if there had been a few whilst the neural network was still adapting.

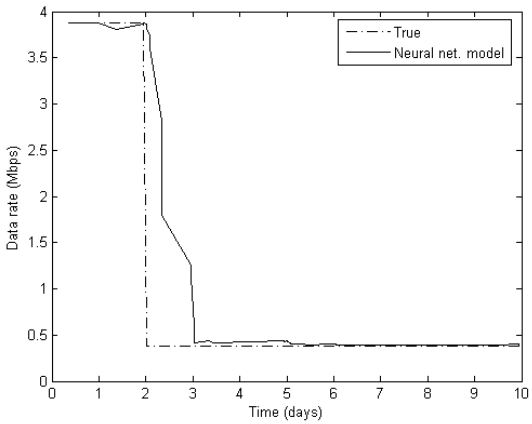

Figure 12: Neural network learning about a step change in the downlink data rate.

Conclusions

It is essential that planning algorithms maintain up-to-date estimates of resource levels, based on measurements, to prevent modelling errors accumulating. However, this will not

reduce future modelling errors unless a learning element is introduced. This work has shown that a neural network can be used make resource use models adapt when the behaviour of a spacecraft changes. Continuously updating training data with measurements of true resource levels ensures that corrections to resource models are always up-to-date.

This approach has been incorporated into the NEAT algorithm and applied to the case of SSTL's UK-DMC imaging satellite. The results show that the neural network was able to learn about changes in resource behaviour. This made planning more robust to faults developing on the spacecraft and enabled effective autonomous planning and scheduling to continue even after a sudden change in system performance.

Acknowledgements

This research was funded by the UK Defence Technology Centre in Point & Image Data Fusion as part of the Autonomous Decision Making project. Thanks to Surrey Satellite Technology Ltd. with their help relating to UK-DMC and to Dr. Steve Mackin for additional mentoring.

References

Brambilla, A.; Lavagna, M.; Da-Costa, A.; and Finzi, A. E. 2005. Distributed planning and scheduling for space system flotillas. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Carrel, A. R., and Palmer, P. L. 2005. An evolutionary algorithm for near-optimal autonomous resource management. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; and Tran, D. 2000. Aspen - automated space mission operations using automated planning and scheduling. In *International Conference on Space Operations (Space Ops 2000)*.

Chu, V.; Da-Silva-Curiel, R. A.; Sun, W.; and Sweeting, M. 2000. Disaster monitoring constellation. In *51st International Astronautical Congress*.

Fukunaga, A.; Rabideau, G.; Chien, S.; and Yan, D. 1997. Towards an application framework for automated planning and scheduling. In *4th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Funase, R., and Nakasuka, S. 2005. Cooperation between onboard executive and planner on ground and its demonstration on remote-sensing nano-satellite "prism". In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Goldberg, D. E., and Lingle, R. 1985. Alleles, loci, and the travelling salesman problem. In *International Conference on Genetic Algorithms and Their Applications*.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Hashisa, Y., and Palmer, P. L. 2001. Epicyclic motion of satellites about an oblate planet. *Journal of Guidance, Control, and Dynamics* 24(3):586–596.

Kramer, L. A., and Smith, S. F. 2004. Task swapping: Making space in schedules for space. In *4th International Workshop on Planning and Scheduling for Space (IWPSS04)*.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution for an autonomous rover. In *4th International Workshop on Planning and Scheduling for Space (IWPSS04)*.

Lin, S. 1965. Computer solutions of the traveling salesman problem. *Bell Systems Technical Journal* 44:2245–2269.

Mai, Y., and Palmer, P. L. 2001. Fast algorithm for prediction of satellite imaging and communication opportunities. *AIAA Journal of Guidance, Control, and Dynamics* 24(6):1118–1124.

Rumelhart, D. E., and et al. 1986. *Parallel Distributed Processing*. MIT Press. 318–362.

Sherwood, R., and et al. 1999. Using aspen to automate eo-1 activity planning. In *IEEE Aerospace Conference (IAC 1999)*.

Wu, S. F.; Brewer, A.; and Palmer, P. 2001. Impredict: a fast image prediction software and its application in the sstl off-axis image scheduling system. In *15th AIAA/USU Conference on Small Satellites*.