Learning agents for distributed and robust spacecraft power management

Stéphane Airiau
Mathematical & Computer Sciences Department
Tulsa, Oklahoma, USA
stephane@utulsa.edu

Kagan Tumer and Adrian Agogino
NASA Ames Research Center
Moffett Field, California, USA
{kagan | adrian}@email.arc.nasa.gov

ABSTRACT

Power generation and distribution in spacecraft presents an interesting resource allocation for learning agents. In this problem, not only does the life expectancy of the batteries and the power demands of various devices need to be balanced but sudden power failures need to be managed. Fixed, centralized strategies offer adequate results in reliable and static environments, but are not well suited to harsh conditions where because of malfunctioning batteries and changing device needs, the power needs to be adaptively balanced. In this work, we propose a multi-agent system to coordinate power distribution for spacecraft systems. The use of reinforcement learning agents at each device and battery provides the adaptiveness and robustness that the domain requires. The fundamental problem we need to solve is to determine how to coordinate agents so that their independent actions lead to good global behavior. We present results showing that agents using agent-specific reward functions that promote cooperation learn to use power in a globally beneficial manner and significantly outperform both utilities aimed to solely optimize device power consumption and utilities aimed to solely optimize battery life. Furthermore, proper agent utility design overcomes device jams, device resets, battery degradation and battery failures, and provides satisfactory power distribution even in extreme cases where over half the devices/batteries are inoperational.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

collective, learning, robustness

1. INTRODUCTION

The management of modern power systems is becoming increasingly labor-intensive and error prone as the size, complexity and dependence between subsystems increases. Consequently, managing such systems using fixed logic algorithms is becoming less and less desirable. This is even truer in spacecraft power systems as both the generation and distribution of power has significantly more restrictions than in terrestrial power grids. For such systems, power management policies that dynamically adapt to accommodate changes in user demand (e.g., sudden device needs) and changes in system states (e.g., faults) are not only desirable but also imperative. Four fundamental system characteristics of modern system control and autonomous computing are crucial in achieving those goals[3]. They are:

1. self-configuration, or the ability to adapt automatically to dynamically changing environments,
2. self-healing, or the ability to discover, diagnose, and react to disruptions,
3. self-optimization, or the ability to monitor and tune resources automatically, and
4. self-protection, the ability to anticipate, detect, identify, and protect themselves from any attacks.

Though all four are relevant to the power distribution domain, in this paper, we will focus on the first three, as the intrusion and/or attack scenario is generally not a concern with spacecraft power systems. In a battery-powered spacecraft, power management requires a delicate balance between the life expectancy of the batteries and the power demands of different subsystems. Deep discharges of a battery limit life expectancy, and therefore, it might be globally preferable to protect the battery and save power even when some subsystems require power. A priori, it is possible to design fixed solution that would only deliver power up to a fixed limit for the battery. However, such an approach is too rigid to trade-off potentially important subsystem usage against the loss of battery life. Furthermore, in cases of device or battery failures, such solution not only become suboptimal, but also in some circumstances, can place the entire spacecraft in jeopardy. It is therefore crucial to use an approach that can ensure the overall safety and reliability of the spacecraft even under severe breakdowns in the system.

In the agent-based solution we present, autonomous agents act as local controllers in order to bring the system to a globally desirable state. This approach is based on first translating the designer preferences into a global utility function. Though allowing the agents to make their control decision independently to optimize that global utility function is appealing, this approach is only suitable in a small system. In larger systems, the impact of a single agent on the full system is swamped by the actions of other agents: agents have a difficult time learning which actions are beneficial. Providing agents with local utilities is an alternative, but this approach is only successful if the agents' utilities are aligned with the global utility. Otherwise, the agents' actions can lead to conflict, inconsistencies, and worse adversarial situations. The approach we present relies on agents making decisions in a coordinated manner using collectives [2, 8]. This framework addresses how to design agent rewards so that the agents collective action optimizes a preset global utility function that evaluates the performance of the overall system.

In this paper, we show how learning agents can be an appropriate solution in distributed spacecraft power management. In particular, we investigate robustness in an environment that contains faults: not only we want to the multiagent system to recover from faults, but we also want it to respond quickly. Multiagent learning has focused on performance in terms of speed of convergence and level of utility, but we believe we should also take into account robustness. As a preliminary step, we present results of response to various faults in the system. In Section 2 we describe the details of the space power system domain used in this paper. In Section 3, we show how a system of multiple learning agents can be applied to this domain to provide robust, efficient control. In particular, we show how the agent's learning problem can be simplified with agent-specific reward functions. In Section 4, we present experimental results and show that the system exhibits our desired properties of self-configuration, self-optimization and self-healing.

2. SPACE POWER SYSTEMS

The space power domain used in this paper is concerned with the resource allocation problem of assigning a set of devices to a set of batteries. Devices have varying power needs and draw power from at most one battery. Each battery has a capacity of available energy. However, using the full capacity of the batteries is harmful to the battery, because deep discharges limit the battery life expectancy. Unlike many resource allocation problems (e.g. bin packing), the cost to use the resource is not uniform.

2.1 System Model

In our model, we assume that the batteries are fully recharged periodically, and we are only interested in the system behavior during the battery discharge period. The power network structure is a set of independent buses; each bus is powered by a single battery. To prevent cascading effects (that yield blackouts in a grid structure) this is the current solution used in spacecraft. During a discharge period, a device/task can draw power from one of the power buses (in our model, it is not possible to change power bus during the discharge period). We model the distribution of power using a simple flow algorithm.

Each battery j is modeled by a rating $R(j)$ (total amount of deliverable energy) and a cost function. To ensure that the battery can perform for given number of charge/discharge cycles, the level of discharge must remain below a depth of discharge threshold δ_M. For example, batteries on board of the International Space Station are designed to operate at a maximum 35% depth of discharge during normal operation [4].

2.2 System Utility

When a task is performed, it contributes to a benefit at the system level. In our model, the benefit is proportional to the amount of power required to process the task. Hence, for each devices i requesting and obtaining power from a bus, it contributes a benefit $\mathcal{B}(i)$. The power cost is modeled by a function penalizing deep discharges. There is a tradeoff between the cost of using the battery and the benefit of using the power: one extreme is to maintain the battery health by limiting power, the other is to perform more tasks, at the expense of risking to damage the batteries. The choice for the tradeoff can be expressed by single utility function that we call *world utility*. This function is defined by the system designer, and the goal of the control mechanism is to maximize this function.

We translate this requirement by a cost function $\mathcal{C}$ that is a logistic function of the depth of discharge δ (see Figure 1):

$$\mathcal{C}(\delta) = \frac{A_1}{A_2 + e^{-A_3(\delta - \delta_M)}}, \tag{1}$$

where A_i and δ_M are constants that control the harmful effects of a discharge affect battery performance. For low δ, the cost is close to zero since the energy drained is not harmful for the batteries. As δ increases up to δ_M, getting closer to the harmful region, the cost increases with an increasing rate. After δ_M, when δ keeps on increasing, the rate of increase of the cost slows. This is because the battery is already operating in a harmful regime: though extra usage has deleterious effects, the damage has already been done.

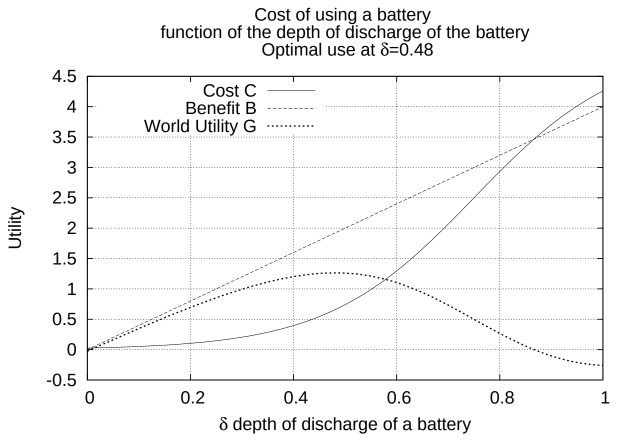


Figure 1: Cost, benefit, utility of a battery

The choice of a particular benefit function and a particular logistic function determine the *world utility function* G as in Figure 1:

$$G = \sum_{\text{device } i} \mathcal{B}_i - \sum_{\text{battery } j} \mathcal{C}(\delta(j)) \tag{2}$$

This function measures the overall system performance, and is the function that the multiagent system as a whole has to optimize. The evaluation of G requires a global knowledge of the system: in particular, it is required to know the depth of discharge of all the batteries, and the demand of the devices that received power. This information may not be available to the devices. However, it can be measured by the system designer.

3. APPLICATION OF AGENTS

As system designers, we require that the devices can be added and removed from the system seamlessly. A centralized control mechanism would require tracking and monitoring a potentially large number of devices, which is error-prone. Instead, to implement these plug-and-play devices, a device comes with its own decision mechanism in the form of an agent. This one decides on the bus that powers the device for the entire discharge period. For now, we consider that each agent is independent and can sense only feedback. We are not considering in this work the possibility of agents sharing knowledge, or forming teams, which might lead to improvement of performance for example.

3.1 Learning in a Collective

From the point of view of the entire system, the goal of the agents is to maximize the world utility G. However, the agent may not be able to know G (because of partial observation, limited communication, etc), and even if it is possible to know G, it is hard to optimize G directly. More importantly, we will show that it is possible to do better without the knowledge of G. An agent may access limited information of the state of the system, and from it, an agent i, it can compute a private utility g_i. Coordination should be embedded in a utility, and two other properties are crucial in the definition of a private utility. First, optimizing independently the private utilities must not lead to working at counter purpose. When an increase in g_i leads to an increase in G, we say that the utilities are factored. The second property concerns with a signal to noise issue. To decide what action is beneficial for itself, an agent must be able to determine the impact of its action on its private utility. An agent will find it difficult to estimate the best decision when changes in other agents' decisions greatly affect the value of a private utility. In [2], measures of the factoredness and learnability are provided.

3.2 Private Utilities of the Agents

To compute the private utility, each agent receives/senses a feedback. This feedback must be inexpensive (in terms of computation, power and implementation) and fault tolerant. The feedback must be simple enough, but carry enough information for the agent to make good decision. We now introduce the different private utilities that we tested in the experimental section. In the following, η denotes an agent, d_η is the power obtained by the agent by choosing to connect to the battery b_η.

•**Team Game (g_{TG})**: The world utility G can be used as private utility, and we refer this utility as a **team game**: all the agents are working together to optimize the same global function. By definition, g_{TG} is guaranteed to be factored. However, it is likely to have a poor learnability since its value depends on the actions of all the agents: it will be difficult for an agent to identify the impact of its decision on this utility. In addition, the knowledge of G requires a global knowledge of the system, which may be difficult and costly to implement. For example, multiple devices can be in charge of computing the value and broadcasting it to all the devices.

•**device - (g_{eg})** A device can estimate its contribution to the world utility and can try to greedily optimize this contribution. We assume that an agent knows the benefit it generates. We assume that an agent can observe the depth-of-discharge of the battery at the end of the discharge period. From this observation, it can estimate its contribution to the cost of the battery: it knows the energy it obtained, and assuming the knowledge of the battery rating, it can estimate the total energy used by all the devices connected to the same bus. For this particular utility, the agent considers that its contribution to the cost is proportional to the amount of power it received from the battery. For an agent η, the value of the utility is

$$g_{eg}(\eta) = \mathcal{B}(\eta) - \frac{d_\eta}{R(b_\eta)\cdot\delta(b_\eta)} C(\delta(b_\eta)).$$

This utility constitutes a partition of the world utility G to all agents: $\sum_{\eta\in\mathcal{A}} g_0(\eta) = G$.

•**device-uniform** g_{unif}: Instead of considering that the contribution on the cost of a battery is proportional to the amount of power received, an agent can consider the cost is uniformly divided between all the agents using the battery.

$$g_{unif}(\eta) = \mathcal{B}(\eta) - \frac{C(\delta(b_\eta))}{n_{b_\eta}},$$

where n_{b_η} denotes the number of agents using the same battery as agent η. The knowledge of n_{b_η} requires a mechanism to count the number of agents connected to the bus. As g_{eg}, g_{unif} is also a partition of the world utility.

•**Wonderful Life Utility (g_{WLU})** Finally, the agents can use a marginal cost to estimate their contribution to the cost of the battery: its contribution is the difference of cost when the agent uses the battery or not. An agent η can use:

$$g_{WLU}(\eta) = \mathcal{B}(\eta) - \left(C(\delta(b_\eta)) - C\left(\delta(b_\eta) - \frac{d_\eta}{R(b_\eta)}\right)\right).$$

An equivalent expression of g_{WLU} is $g_{WLU}(\eta) = G - \mathcal{G}_{-\eta}$ where $G_{-\eta}$ denotes the value of the world utility when all the agents remain fixed, but agent η is removed from the system. This utility is the Wonderful Life utility introduced in [8]. This utility is factored (since the second term is independent of agent i, the derivative of g_{WLU} with respect to agent η is the derivate of the world utility with respect to agent η). This utility has proven effective in many multi-agent system domains including network routing, rover control, job scheduling and congestion games [1, 5, 7, 6]. Note that in this domain, the agents only require the knowledge of the rating of the battery and the observation of the depth-of-discharge of the battery at the end of the discharge period.

•**battery-utility** Instead of optimizing a greedy utility (difference of the benefit of an agent and an estimate on its contribution on the cost), the agents can optimize the utility of a battery. All agents using the same battery can uniformly share the utility obtained using this power. Each agent can receive:

$$g_{bat}(\eta) = \frac{\sum_{j=1}^{n_b} u(j) - C(\delta_b)}{n_b}.$$

This utility requires each device to publish the utility it generated. A mechanism is required to aggregate the benefit obtained by each agent, compute the cost incurred by the battery, and broadcast the utility to the corresponding devices.

3.3 Learning algorithm

Each device uses a simple reinforcement learning algorithm that learns an expected utility for each available action, i.e. it learns the expected utility of using each battery. We require an algorithm that always performs some exploration because of the possibility of failure or changes in the system. Without exploration, a learner would not adapt rapidly to changes in the system. That is why we use Q-learning algorithm with the ϵ-greedy exploration scheme. At the beginning of the discharge period, each agent chooses the bus it is going to use during that period. At the end of the period, the agent receives a feedback according to the level of information accessible, and the agent computes its reward r and updates its estimate using a Q-update rule: $Q(b) \leftarrow (1-\alpha)Q(b) + \alpha r$, where α is the learning rate. The learners use the ϵ-greedy exploration scheme: $\epsilon\%$ of the time, the learner picks a random action, and the rest of the time, the learners pick the action leading to the maximum expected utility. The agents keep an estimate $Q(b)$ of the expected utility of choosing a particular battery b.

4. RESULTS

We present the experimental results of our simulator. As self-interested entities, the agents are competing to maximize their utilities. We describe and analyze the environment with different power demands (which modify the pressure of the competition to obtain power) but no faults. Finally, we present experiments testing the robustness of the system.

We experiment with a system of eight batteries and different numbers of devices. For the experiments we present, the batteries are identical, sharing the same rating (100 units) and the same cost function. We use the following cost function (see Figure 1):

$$c(\delta) = \frac{5}{1 + e^{-7(\delta - 0.75)}}.$$

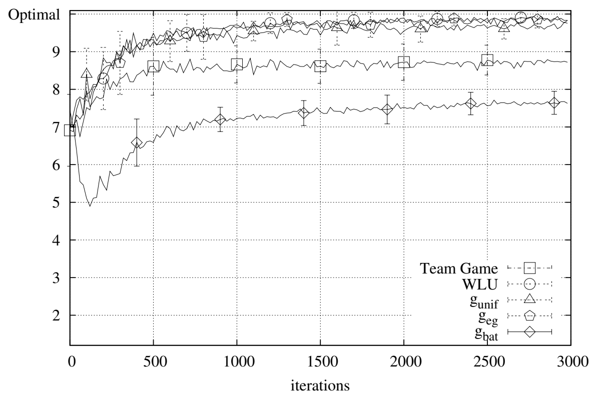
The benefit of performing a task requiring d units of power is set to $\frac{d}{25}$. The optimal usage of the battery is at a 48% depth of discharge, which represents 48 units of power. A load of 384 units will produce an optimal usage of the battery. We chose to experiment with a load of 400 (normal load) or 800 units (overload). The normal load is achieved by using a set of 50 agents: half of them require 2 units, the others 14 units. For the overload system, the set of agents is doubled. With the normal load, the agents have enough "room" to distribute the load and the difficulty of the problem lies on an appropriate allocation. In the overload case, some of the agents must decide not to be plugged in the system at all. This asymmetry in the load of the agents makes the problem harder to solve. Each agents uses Q-learning with the epsilon-greedy exploration scheme with the following parameters: $\alpha(t) = \frac{1.0}{5+\frac{t}{50}}$ and $\epsilon(t) = \frac{1.0}{1+\frac{it}{50}}$

4.1 Non-faulty environment

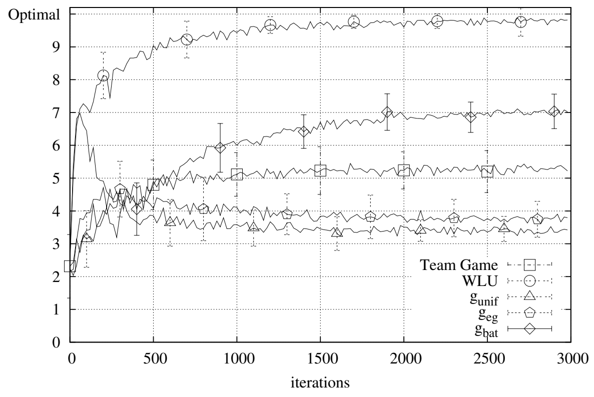
We first compare the different private utility functions in a system where the load is normal, i.e. there exists optimal allocations of the agents where all agents can complete their task. The results in Figure 2(a) show that three utilities are able to perform at close to optimal level: g_{WLU}, g_{eg} and g_{unif}. The remaining two utilities are not. TG suffers from a poor learnability: although perfectly aligned, the agents are not able to determine their impact on the system. The battery-utility is performing the worse. We observe that the agents are under-using the power. Despite the possibility for all agents to get power with optimal performance, a number of agents requiring lots of power still prefer not to be plugged. In typical runs, we observe that for most batteries, the difference between the optimum load and the final load is larger than 14. Because of the division by the number of agents using the particular battery, the agents believe the optimal use of the battery is much less than it is. For example, if all agents demand 2 units, the optimal number to optimize g_{bat} is 7 agents. From there, the utility decreases and is concave down. This utility over-protects the batteries.

Results with an overload are presented in Figure 2(b). First, this curve illustrates that the system is self-configuring: the learners explore the different possibilities and find a way to adapt to ensure high level of performance in the system. Because it is factored and learnable, g_{WLU} still performs close to optimal, but all other utilities suffer a great drop in performance compared to the situation with the normal load. Two factors contribute to the increase of the problem difficulty: a greater number of agents (scalability issue) and the fact that some agents must decide not to be plugged to the power network, which greedy agents do not. The utility g_{unif} and g_{eg} are the ones with the biggest drop: the agents are requesting too much power. Each agent is greedy in a sense and tries to get some utility. Agents prefer to be unplugged (getting a utility of 0) only if the use of a battery provides them a negative utility. Hence, they use the battery up to a depth of discharge providing positive utility. A deeper analysis of the results shows that with g_{eg}, agents with both high and low load get powers whereas with g_{unif}, only the agents with a high load request power. With g_{unif}, the cost incurred by each agent is proportional to the number of agents using the battery, hence all the agents sharing the same battery incur the same cost. Agents requesting more power receive more utility; hence the agents with high demand tend to get more utility than the agents with smaller load. The agents with a higher load converge faster than the agents with a small load, and are able to get their power. The greater number of agents is the cause for the decrease of performance of Team Game: because of the increase of the number of agents, the signal to noise ratio has decreased, making it harder for an individual agent to understand the impact of its action.

Although under-using the system can be seen as a weakness under a normal load, this can turn into an advantage under an overloaded system: battery-utility performs comparatively better that in the normal load case, mainly because the other utilities behave much worse. The decrease in performance is small compared with the normal load case. This is because few more agents with a small request use the battery.



(a) Normal load (50 agents)



(b) Overload (100 agents)

Figure 2: Non-faulty System.

4.2 Robustness

Robustness, fault tolerance, low maintenance are important properties of complex systems. In a case of a large number of devices, it might not be possible to maintain or replace faulty devices when a problem occurs. In addition, if a power source is faulty, or if the total capacity decreases over time, the devices must respond to these changes of power availability. We believe robustness should also be considered as an important metric of performance of multiagent learning. We now provide experiments to show that the learning agents can make the system self-healing and self-optimizing: we are testing the robustness of the system by experimenting different faults that can affect the devices or the power sources:

•**device jam:** We model a problem in the device which switches between the different bus to power the subsystem. The agent is faulty and gets locked to a particular bus in the system. For the experiments, the bus powering the device is chosen at random. If too many devices get locked to the same bus, the system may no longer be able to work at the optimal level, and in this case, the non-faulty agents must avoid this bus.

device reset: This fault models a surge in the power system. The agent controlling the choice of the power bus accidentally resets itself. In this case, the agent must re-learn a policy.

battery degradation: The battery fails to re charge completely. Typically, a battery is composed by many cells, and we model a sudden failure of few of these cells: as a result, the total power delivered by the battery is reduced. However, the cost function for this battery is unchanged: after the accident, the battery starts anew with a $\delta\%$ depth-of-discharge. If δ is high, the cost of using even a small quantity of power might be expensive. In our experiments, $\delta = 25\%$

In the following experiments, devices becomes faulty once, one at a time, over a window on n charge cycles, when the learning algorithm has converged. The speed of reaction of the learning algorithm plays an important role on the drop of performance and on the time to converge again. We study the system across two dimensions: the error rate ρ and the length n of the window where the faults occur, or length of the "accident". We present results where the faults can occur in a window of 100, 500 and 1000 cycles for various error rates ranging from 10% to 90%. Each graph is an averaged over 20 runs. The system for these experiments is the same than previously (Figure 2).

4.2.1 Device jam

During the accident, devices get locked to a particular batteries. It may happen that many faulty agents use the same battery, with a very negative effect on G. Under this scenario, non-faulty agents must not worsen the world utility. In Figure 3, we compare the utility obtained after the accident and convergence to a new equilibrium. For the three utilities that performs close to optimal in the non-faulty environment, we observe a similar trend. For TG, which does not perform optimally, the influence of the fault is less since the agents are not able to distinguish between noise and faults. For the battery-utility, we see some improvement with increasing faulty agents: the agents tend to under-use the batteries. With the faults, more agents use the power, which results in relative better performance.

4.2.2 Battery degradation

An accident corresponds to a diminution of the total amount of energy delivered by a battery. Agents must reduce their demand to use the battery at the optimal depth-of-discharge, which remained unchanged. In Figure 4, we present the world utility once the agents have adjusted their demand. For example, when there are two faulty

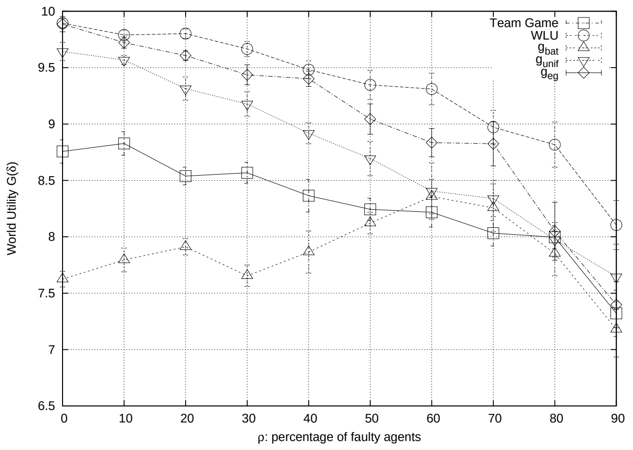


Figure 3: Device jam: comparison of the performance.

batteries and the agents are using g_{WLU}, the world utility of the system is 8. If the agents use the battery at their optimal depth-of-discharge, the cost incurred by the batteries does not change. As benefit is proportional to power consumption in our model, the decrease of performance is linear. This explanation is verified for g_{WLU}, which performs optimally. When the agents are using g_{bat}, the batteries are used at a constant level, which explains the same behavior. Team game suffers the same way, but because of high level of noise, the performance is not very sensitive to the faults. We can notice that when 5 batteries fail, the agents are using almost all the available power from the remaining batteries.

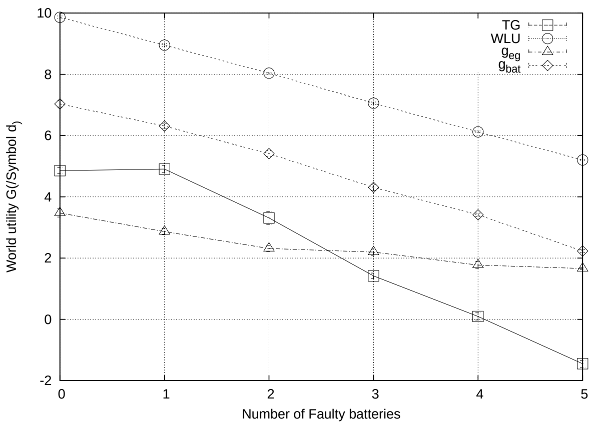


Figure 4: Battery degradation - Comparison of the performance.

4.2.3 Device reset.

During the accident where $\rho\%$ of the agents are resetting their learning algorithm, G drops, since the system is no longer at the optimal distribution. The depth of he drop depends on the length of the accident and of the private utility used. In Figure 6, we compare the behavior of all the utilities with an overload system and either 20% or 50% of the agents are resetting. The drop in performance is greater for g_{WLU}, because it is the only one to reach optimal performance. For the battery-utility, the immediate effect of the faults is not a drop, but a raise. Again, this is because the faulty agents will explore and use more power, resulting in an increase of G. Then, the agents learn to under-use the battery. Note that the new equilibrium reached is slightly better than the previous one. We observe a similar phenomenon for g_{TG}: interestingly, we noticed that in the case of the overloaded system, the reset helps the system: the environment is more static and the agents can understand better the impact of their decisions on G. This reset help the system to move out of a local maximum. In Figure 5, we present a

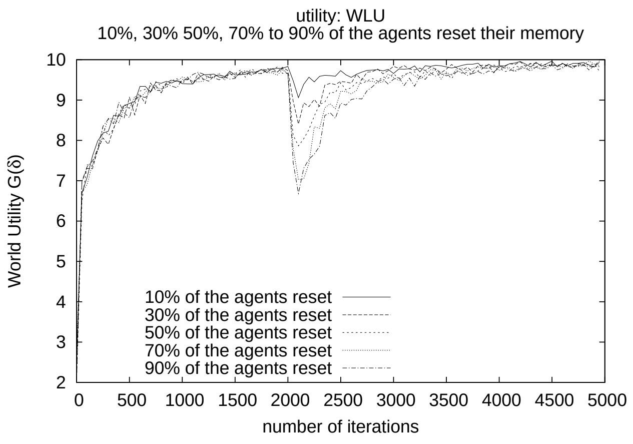


Figure 5: Device reset - private utility is WLU.

typical set of curves. This particular chart occurs with an overload and the agents are using g_{WLU}. The accidents occur between iteration 2000 and 2100. The greater the value of ρ, the deeper the drop, and the longer it takes to reach the previous level of performance.

5. FUTURE WORK AND CONCLUSION

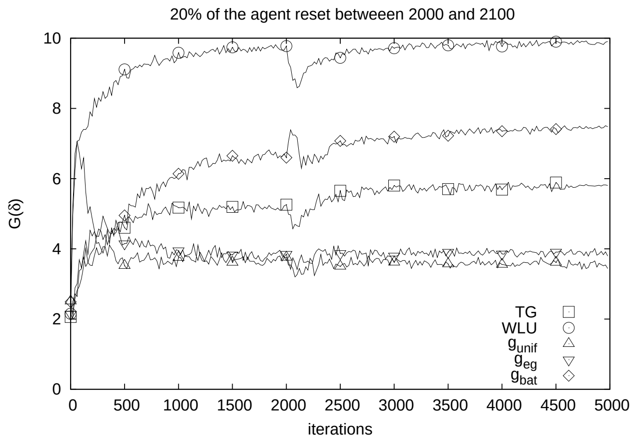
In this work we addressed power distribution in spacecraft where the life expectancy of the batteries and the power demands of various devices need to be balanced. This is a challenging resource allocation problem where, because of the long-term harmful effects of deep battery discharges, it is beneficial for devices not to request power even in cases where power is available. Our results show that agents using well-designed agent-specific reward functions learn to use power in a globally beneficial manner and significantly outperform both utilities aimed to solely optimize device power consumption and utilities aimed to solely optimize battery life. Furthermore, such agents overcome device jams, device resets, and battery degradation providing good power distribution even in extreme cases where over half the devices/batteries are inoperational.

Finally, in this work, we assumed that all the devices have equal importance. However, in most real world application, this assumption is incorrect. Some systems are more important than others are, and therefore their power needs supersedes those of other systems. In space vehicles for example, life support systems have the highest priorities. We are currently expanding these results to incorporate device priorities so that critical systems are ensured of getting power in case of a breakdown in the system.

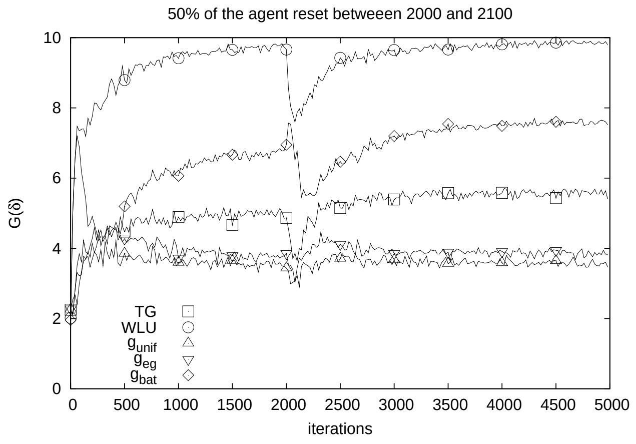
6. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] A. Agogino and K. Tumer. Multi agent reward analysis for learning in noisy domains. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Utrecht, Netherlands, July 2005.



(a) Comparison when 20% of the agents reset



(b) Comparison when 50% of the agents reset

Figure 6: Device reset

[3] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Syst. J.*, 42(1):5–18, 2003.

[4] T. B. Miller. Nickel-hydrogen battery cell life test program update for the international space station. Technical report, NASA Glenn Research Center, 1999.

[5] D. H. W. Stéphane Airiau Sandip Sen and K. Tumer. Providing effective access to shared resources: a coin approach. In *Engineering Self-Organizing Applications, First International Workshop, ESOA 2003. Melbourne, Victoria, July 15th, 2003. Workshop Notes*, pages 29–36, 2003.

[6] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[7] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[8] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.