Evolving Distributed Resource Sharing for CubeSat Constellations

Chris HolmesParker
Oregen State University
204 Rogers Hall
Corvallis, OR 97331
holmespc@onid.orst.edu

Adrian Agogino
UCSC at NASA Ames
Mail Stop 269-3
Moffett Field, CA 94035
Adrian.K.Agogino@nasa.gov

Kagan Tumer
Oregen State University
204 Rogers Hall
Corvallis, OR 97331
Kagan.Tumer@oregonstate.edu

ABSTRACT

Advances in miniaturization will allow for the commoditization of tiny satellites, known as "CubeSats." This commoditization in addition to reducing price and increasing the numbers of satellites, will also result in the "democratization" of small space missions where numerous institutions can launch their own satellites. However, current algorithms made for small tightly-managed space missions are ill-designed to take advantage of the huge amount of resources available in a decentralized collection of CubeSats. We believe that multiagent evolutionary algorithms are ideally suited to exploit the distributed nature of this new problem. This paper presents a solution where a customer in need of satellite observations can reliably obtain these observations at low cost, through the help of a multiagent system as an intermediary. Each agent in this system is assigned to a single CubeSat. Given a set of the customer's observational needs, and models of the CubeSats' salient properties, the agents evolve policies that attempt to purchase an appropriate set of observations at a low price. This system is especially flexible as it demands no centralized resource broker, contracts or commitments of resources. We perform a series of experiments on an Earth-observation domain. The results show that the evolutionary methods combined with multiagent techniques have three times the performance of a simple hand-coded allocation algorithm, and twice the performance of simple evolving agents.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

Keywords

CubeSat, Evolution, Multiagent Systems

1. INTRODUCTION

Currently, satellites are very expensive resources that need to be coddled carefully. The costs of satellite missions can exceed billions of dollars, with teams of engineers, managers and scientists working together to extract all the information they can out of these missions. These missions are carefully planned and orchestrated by large institutions over a period of many years. However, in the near future this traditional satellite paradigm could change dramatically with the introduction of very small satellites known as "CubeSats." The number of CubeSats will dramatically increase due to reduced costs coming from platform standardization, availability of COTS (commercial off-the-shelf) parts and reduced launching costs [14, 17]. These satellites will have numerous capabilities, including in situ measurements of the thermosphere, interferometry, communication and Earth observation [12]. Collaborative networks of CubeSats offer mission capabilities that are impractical for larger satellite platforms due to cost restrictions, including simultaneous in situ measurements of multiple locations in space and temporally separated measurements of precise points in space[16, 11]. In addition, they offer lower cost and increased robustness compared to traditional satellites due to system reconfigurability [3]. In addition, networking clusters of CubeSats together in order to boost performance is becoming a popular concept, similar to computer clusters[1, 8]. However, while having numerous advantage, making effective use out of large numbers of heterogenous CubeSats is a difficult problem.

1.1 Motivation

As an example, consider an instance where a small community needs to observe the realtime progress of a local forest fire. There are many aspects of this fire that can be observed from orbit, including fire intensity, distribution, and movement. A few dozen observations would be useful, but with diminishing returns beyond this number. Currently, making these observations is difficult and expensive due to the limited number of satellites. However, in the future, there may be tens of thousands of tiny CubeSats able to make these observations. How can this small community in an economical way take advantage of these resources?

A straight forward solution to this problem is a centralized satellite resource broker. Under this scenario, our small community would register its fire observation needs to the broker, and the broker would try to find the resources, trading off the costs and benefits of all the other requests that were registered. While this method is attractive for networks of large satellites, there are three main difficulties this centralized system might have with a large, but disorganized collection of CubeSats: 1) CubeSats are likely to be owned by many different countries and institutions that may not trust having their resources used by a centralized resource broker, 2) CubeSats will be in unpredictable states of repair and may be owned by institutions unable to make reliable commitments, 3) There may be so many CubeSats (perhaps millions), that a centralized system could simply not scale efficiently.

As an alternative to a centralized solution, our community could buy observations directly from the owners of the CubeSats. For this process to be effective, the community needs to do two primary things, 1) Buy the appropriate number of observations taking into account the unreliability of CubeSats, 2) Buy the observations at the lowest possible price. For these two things to happen, the reliability and expected cost of each of the CubeSats needs to be modeled so that an appropriate combination of request for observations can be made. While taking all these considerations into account would ordinarily be difficult for a small institution or community, an agent based system can help by modeling the satellites behavior and evolving policies to maximize value.

1.2 Decentralized Agent Solution

We propose to have a decentralized solution to this problem, through the use of an "agent" intermediary. But first we have to decide what an agent is in this domain.

1.2.1 What is an agent?

There are numerous ways agents can be used and defined. Here we explore a few alternative types of agents:

1. **Trivial**: Just pass information between CubeSats and customer.
2. **Owned by Customer**: Every customer has its own agent buying observations for that customer.
3. **Owned by CubeSat**: Every CubeSat has its own agent selling observation for that CubeSat.
4. **Independent**: One agent per CubeSat, buying observations for customer.

In the first definition, the agent is a simple intermediary. While this solution may work for a very sophisticated customer, in general it does not solve the problem of how a customer can buy an appropriate set of observations at a low price. In the second definition, each customer has an intelligent agent that tries to make these purchases for the customer. However, this solution has similar limitations to the trivial agents, as the agent would have to be sophisticated enough to know the properties of the thousands of CubeSats in existence and come up with a policy satisfying the customer's demands at a low price. In the third definition, the task of the agent is much simpler. It knows all the properties of the single CubeSat that owns it, and knows at what price points it can sell its observations for. However, the issue with this approach is that it can be very inefficient, since agents trying to maximize revenue for its CubeSat may try to sell observations that are not valuable to the customer.

In this paper we focus on the final definition for an agent, where *agents are independent*, there is one agent per CubeSat and the task of the agents is to buy an appropriate set of observations for a given customer. With this definition, the requirements of an agent is relatively simple. It needs to model the capabilities, reliability and price point of only a single CubeSat. Then when a customer makes an observational request to a set of agents, the agents coordinate to purchase an appropriate set of observations. This agent model has a number of advantages:

1. CubeSats with any price structure can participate.
2. Unreliable CubeSat can participate.
3. CubeSat owner can choose not to participate on case by case basis.
4. Customers can choose not to buy resources from particular CubeSats.
5. Agents can scale with number of CubeSats.

With independent agents, CubeSats of all types can participate. An agent can simply decide not to use its CubeSat if it is inappropriate for the task. The most difficult task for an agent is to coordinate effectively with other agents. In this paper, we will focus on this coordination problem, and how policies can be evolved that allow agents to coordinate well. Note that this paper does not cover the broader case where there are multiple customers at the same time. However we believe that a similar mapping could be made in this case, where the agents are trying to maximize a larger overall utility over all customers.

1.2.2 Proposed Solution

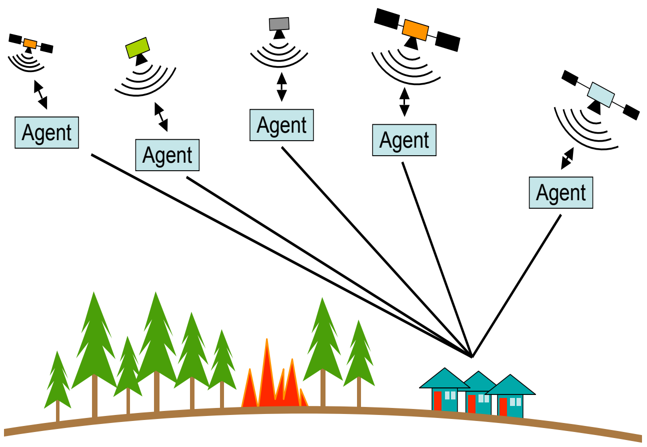

Figure 1: Small community needs CubeSat observations of a forest fire. Agents handle observation request. Using one agent per CubeSat, an agent bids for the observation of a particular CubeSat. As a collective agents as a whole must bid for an appropriate number of observations with minimal cost.

This paper proposes a multiagent solution, where independent agents help a consumer of satellite resources, buy an appropriate combination of resources at low cost (see Figure 1). In this algorithm an agent is assigned to every CubeSat, and is responsible for making a monetary bid to its CubeSat for its observation. The consumer makes a request to all of the agents for satellite observations, giving the agents a utility equation representing the value of the benefit it would receive from different numbers and types of observations. Each agent then makes a bid for an observations, using a bidding policy. This policy is evolved from a population of policies, using the value benefit equation given by the consumer, in combination with the agent's model of its CubeSat. These bids take into account the value of an observation, the likelihood that the CubeSat will be able to carry out an observation, and the likelihood that the CubeSat will be willing to carry out an observation given the value of a bid. All these values have uncertainty, making this a difficult problem. In addition the agents will need to coordinate the evolution of their policies so that the collection of observations derived from all the winning bids is beneficial to the customer and bought at a low cost. In this paper, we will explore these aspects in more detail.

Section 2 provides a background on distributed evolutionary methods and simple negotiation mechanisms related to the work performed in this paper. Section 3 provides an introduction to CubeSats, a popular small satellite platform. Section 4 outlines our problem domain, key learning issues, and our specific algorithm. Section 5, provides a detailed description and analysis of the experiments and experimental results. Section 6 provides a discussion and conclusion.

2. BACKGROUND

Evolution and genetic algorithms have been used extensively with satellites in a diverse set of domains, including observation scheduling, channel allocation and communication routing [7, 15, 9, 20]. In [7] a genetic algorithm is successfully used to schedule Earth-observing satellites under numerous constraints such as power limitations, thermal constraints and ground station communication. In [15] communication channels are efficiently assigned through a genetic algorithm used to solve global optimizations is combined with a hopfield network used to solve local optimizations. In [20] a genetic algorithm is used to perform communication routing among satellites to improve quality of service. In all these cases genetic algorithms are able to improve performance in a non-linear domain and maximize use of the limited resources available from a small set of satellites. However, they are not designed for allocating abundant resources likely to be available in the CubeSat paradigm.

In addition to evolution and genetic algorithms, agent methods have been used successfully to promote resource sharing within constellations of autonomous heterogeneous satellites. To date, agent-based satellite coordination research has included: resource allocation between on-board peripherals of individual satellites (processor and power allocation), autonomous coalition formation based upon negotiation mechanisms, and additional satellite-to-satellite coordination and resource sharing mechanisms to complete complex missions [4, 6, 13, 5].

3. CUBESATS

In this paper, we are concerned with collecting resources from a constellation of CubeSats. A CubeSat is a type of small satellite that measures 10cm x 10cm x 11cm and weighs 1 kg or less. These devices carry various scientific payloads and can be launched for around $100k per satellite [12]. Existing CubeSat missions include: Earth Quake Monitoring (Quake-SAT); Monitoring lightning storms in the Low-Earth atmosphere; and Solar flare and gamma ray burst detection. The diverse capabilities of these small satellites, coupled with their low mission costs make them an ideal space-based research platform for universities and small organizations, which historically have not been able to afford access to space.

Space-based research has gone away from large one-of-a-kind spacecraft, towards smaller, less expensive spacecraft [3]. In recent years, it has been demonstrated that coordinated teams of smaller spacecraft can carry out traditional satellite missions, as well as novel missions involving spatially and temporally separated measurements [3]. This shift towards smaller, less expensive devices could mirror the paradigm shift that has happened over the last few decades in the computer industry, as focus has shifted from large, expensive mainframes towards smaller, cheaper sets of coordinated workstations. Through coordination and resource sharing, teams of smaller spacecraft can have the same mission capabilities of a large spacecraft, at a fraction of the cost.

4. COORDINATING CUBESATS

Academic and industrial programs continue to launch CubeSats equipped with scientific instruments into Low Earth Orbit (LEO) [10]. The capabilities of individual CubeSats are fairly limited due to their size and mass restrictions. Yet, coordinating multiple CubesSats and collecting their combined resources greatly increases their overall value. In this paper, we focus on a particular instance of a CubeSat coordination problem where a customer can coordinate resource purchases for a set of existing CubeSats.

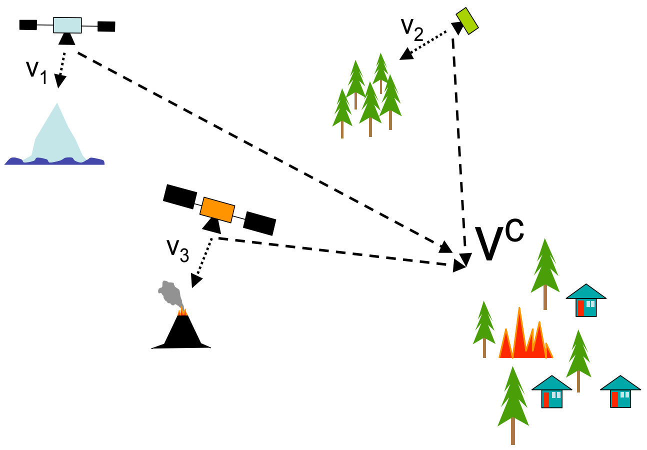

Figure 2: A set of CubeSats gain different levels of value, v_i, from observing their own point of interest. A potential customer would like satellites observations of its own point of interest. The value of these observations to the customer, v^c depend on mix and number and locations of satellites involved.

4.1 Observational Values

In this paper we assume that there is a set of Earth-observing CubeSats in low Earth orbits, where each satellite is owned by a separate institution [1]. Each of these CubeSats

[1]For simplicity, we will anthropomorphize the CubeSats by treating a CubeSat and the institutions that own it, as the same thing.

is interested in observations of a particular geographic region of interest for reasons such as crop monitoring, volcano monitoring, fire monitoring, reconnaissance, search and rescue, and weather monitoring. We assume that each CubeSat places some value on observing a particular point of interest (POI), but is able to observe any region of interest beneath its orbit (see Figure 2). Each CubeSat places a different value on monitoring its own POI. In addition, this value depends on the distance between the CubeSat and its POI. In general the further the CubeSat is from its POI the less value it will have in monitoring it. Formally we express this value for CubeSat i as:

$$v_i(d_i^p) \; , \tag{1}$$

where d_i^p is the distance between the CubeSat and its POI.

The goal of this paper is to figure out how a customer, with no satellites of his own, can make use of these existing satellites to observe a point of interest that this customer is interested in. In general the value of a set of CubeSat observations to a customer is a function of the observational capabilities of the CubeSats (e.g. resolution, heat sensing, particle sensing, etc.) and the distance of the satellite to the customer's POI. Formally we define this value function for the customer as:

$$v^c(d^c) \; , \tag{2}$$

where d^c is the vector of distances between the CubeSats and the customer's POI (note for simplicity the observational capabilities are rolled into v^c). In general, the more observations are better and observations closer to the customer's POI are better.

4.2 Customer Objective

The objective of the customer is find a set set of observations that have high value to him, v^c, at a low cost:

$$G(d^c, d^p) = v^c(d^c) - \sum_i c_i(d_i^p) \; , \tag{3}$$

where c_i is the cost paid to get an observation from CubeSat i. While we assume that there is no direct cost for a CubeSat to observe a POI, we assume that a CubeSat will not make an observation for a customer unless it is paid approximately its opportunity cost for not observing its own POI $v_i(d_i^p)$. Therefore in generally the cost paid $c_i(d_i^p)$ will be higher than opportunity cost $v_i(d_i^p)$ for successful bids. While in some cases this opportunity cost is very high and a CubeSat will never offer to make an observation for a customer, in other cases in could be close to zero, especially if the CubeSat is not within range of its own POI.

4.3 Agent Model

Our overall goal is to figure out how a customer can maximize G; i.e. purchase a set of observations that have high value to him at a low cost. In general this will be possible when a set of CubeSats with appropriate capabilities is close to the customer's POI, increasing his value, and far from their own POIs, reducing their opportunity cost. This leads to our central problem: How can a customer sensibly buy a set of satellite resources, when he knows little about the CubeSats' capabilities, their cost model, or even their willingness observe the customer's POI?

In this paper, we propose to address this problem using agents combined with evolution. In this paradigm a single agent is assigned to a single CubeSat, and the action of an agent is to bid on the observations from its CubeSat on behalf of the customer. As a whole, the goal of the agents is to balance the value of all the observations to the customer, v^c, with the cost of these observations. Note that to do this effectively each agent has to take into account both local and global considerations: 1) Locally an agent will make bids when its CubeSat has likely high value, such as when it is close to the customer's POI, and when its CubeSat has likely low cost, such as when the CubeSat is far from its own POI, 2) Globally an agent should only bid for an observation, when that observation increases the total value, v^c, for the customer - agents should not bid for observations that are not needed or have low marginal value.

For an agent to accomplish its task, we assume that each agent is given the value function for the customer $v^c(d^c)$. We also assume that each agent has some approximate model for the value of its CubeSat, $v_i(d_i^p)$. However, we make little assumptions of the quality of this model or where it came from. For some agents $v_i(d_i^p)$, may be given to it directly from the CubeSat. For other agents, its model for $v_i(d_i^p)$ may be a crude approximation generated through previous interactions with the CubeSat. Given these value functions, we have the agents maximize the customer's objective through evolution.

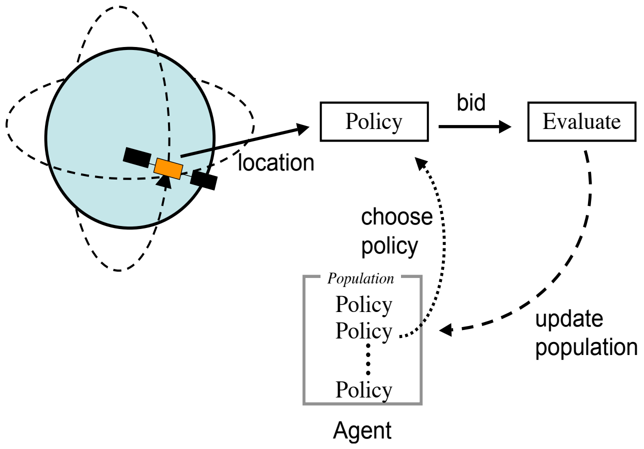

Figure 3: An agent evolves by cycling through a process where at each time step it 1) picks a bid policy from its population, 2) has its policy make a bid based on the location of the satellite, 3) evaluates the benefit of the bid against the observations bought by all the other agents, 4) updates its population based on the evaluation.

4.4 Agent Evolution

The job of each agent is to evolve a policy that makes bids for CubeSat observations in such a way as to maximize the overall customer's objective G. In this paper we evolve policies that map the location of the satellite (from which the distances d^c and d^p can be derived) to a bid value that the agent will make to the CubeSat. This process is shown in Figure 3. The agent is assumed to have a model of the satellite's orbit, a model of the satellite's value function, v_i, and the true value function for the customer, v^c. The evolutionary process of the agent is a fairly simple one used previously in online evolution [2]. At every time step, the agent chooses a policy from its population using an epsilon greedy selector. Then it uses this policy to determine a bid amount for its CubeSat, based on the location of the CubeSat. Next it samples its model to determine if it wins this bid. Then it looks at all the winning bids of all the agents and evaluates the effectiveness of the chosen policy. The policy's evaluation table is updated with a learning rate α such that $evaluation = \alpha * newevaluation + (1 - \alpha) * oldevaluation$. After updating the evaluation table, it updates its policy. This policy update is done by removing the worst performing member with probability P and replacing it with a mutated copy of the best performing member.

This process continues until the agents have produced policies that lead to high values of the customer's objective G. Critical to achieving this is to choose good evaluation functions. An agent's evaluation function has to take into account bid actions of all the other agents so that it can bid on observations that will be useful in the context of all the other observations that are bid on. However, it cannot have an evaluation function so complex and noisy that it will never evolve effectively. We shall explore these tradeoffs in the next section.

4.5 Agent Evaluations

Many different types of evaluation functions are possible for agent evolution. The first and most direct approach is to let each agent use the global system objective, G. However, in many domains, especially domains involving large numbers of agents, such an evaluation often leads to slow evolution [2]. This is because each agent has relatively little impact on its own evaluation. For instance if there were 100 agents and an agent takes an action that improves the system evaluation, it is likely that some of the 99 other agents will take poor actions at the same time, and the agent that took a good action will not be able to observe the benefit of its action.

Another possibility for an evaluation is to use a local agent-specific evaluation that only accounts for the action of the particular agent. While with such evaluations, an agent can easily see the impact of its action on its evaluation, in most domains, the local evaluations are not aligned with the global system objective G. In such domains, an agent can maximize its own local evaluation, but in doing so it can reduce the overall system performance. Local evaluations are primarily useful in problem domains in which the local evaluations can be created in such that they are directly aligned with the system performance. However, in many complex problem domains it is notoriously difficult to derive local evaluation that are well aligned with the system objective function.

Our approach to the problem is based on selecting an evaluation with the benefits of both the global and local evaluation functions, without the drawbacks associated with them. In this work, we focus on "*difference evaluations*" which aim to provide an evaluation that is both sensitive to that agent's actions and aligned with the overall system evaluation [2, 18, 19].

4.5.1 Difference Evaluation Function

The **difference** evaluation function used in this paper is of the form [18, 19]:

$$D_i = G(b) - G(b - b_i) \ , \qquad (4)$$

where b is a vector of the bid actions of all the agents, and $b - b_i$ are the bid actions of all the agents except with the bid action of agent i removed. Intuitively this causes the second term of the difference evaluation to evaluate the performance of the system without agent i and therefore D evaluates the agent's contribution to the system performance. There are two advantages to using D: First, bids taken by other agents that are not tightly coupled to agent i's bid will be subtracted out by the second term of D, significantly reducing the "noise" in the evaluation. Second, because the second term does not depend on the bids of agent i, any bid by agent i that improves D, also improves G. This means that agents that evolve policies that tend to maximize D will tend to also maximize G and will likely do so more quickly. In fact in cases when G is linear in the agents actions, D_i will only be affected by the action of agent i, and the agents will evolve very quickly. When G is non-linear, D_i will be affected by other agents, but there effects will usually be reduced and agents will still evolve quickly [2].

5. EXPERIMENTS AND RESULTS

We perform extensive simulations to test the effectiveness of different satellite evaluation functions and different agent types under a wide variety of environmental conditions. In all experiments a customer has a set of agents that bid on CubeSat observation. The goal is always to maximize the system evaluation function of the customer defined in Equation 5. However the goals of the agents may not be to directly maximize the system evaluation. Instead we test five different types of agents, to see their effectiveness in the overall maximization of the system evaluation.

5.1 Setup

In our experiments we simulate the movements of satellites within the 10x10 grid near a POI. Unless otherwise stated, all simulations have 100 satellites. The maximum distance satellites can move in a given time step is d_{max}, which corresponded to 1/100 of the individual satellites orbital angular period. In these experiments, the POI locations are randomly chosen, but remained constant over all time steps in a given trial. The allowed bid values are set between zero and nine, based upon a discretization of the maximum and minimum inverse-euclidean-distance between a satellite and a POI. The experiments are run for 10000 time steps, and the environment is static. All results are averaged over at least one hundred independent trials. All primary results are statistically significant with $p < 0.05$. For each experiment the 95% of the initial policy bids are set to 0, with the remaining 5% set to a random value between 0 and 9. For evolution $\epsilon = 0.1$, $\alpha = 0.2$ and $P = 0.2$.

In our experiments the system objective function is:

$$\begin{aligned} G &= v^c - \sum_i c_i \\ &= \sqrt{\sum_i (v_i^c)^2} - \sum_i c_i \ , \end{aligned} \qquad (5)$$

where v_i^c is the value of the *local* information gained from the use of CubeSat i, and c_i is the cost of acquiring resources from CubeSat i. Overall our model for the customer's value for a set of satellites $v^c = \sqrt{\sum_i (v_i^c)^2}$ provides diminishing returns for increasing levels of information. As in many real world problem domains, there exists a saturation point, beyond which additional information or resources become less beneficial for the system, even if the per unit cost remains constant. This problem is difficult, as agents are coupled. Each agent not only has to figure out what to bid for an observation, but has to figure out if an additional observation would actually be useful since its value reduces with the number of observations.

5.2 Agent Types

In these experiments, the five types of agents used are as follows:

1. **Random**: Agents evolve using random evaluation function.
2. **Strawman**: An agent's bid is precisely equal to the CubeSat's opportunity cost v_i .
3. **Local**: Agents try to maximize a local objective.
4. **Global**: Agents try to maximize the system objective.
5. **Difference**: Agents try to maximize difference objective.

These different types of agents are designed to test performance on a wide number of plausible solutions. The strawman agents are assumed to know the value the CubeSat places on observing its own POI and make a bid just above its expected value (equal to its value plus an arbitrarily small epsilon). If the CubeSat always accepts bids higher than its opportunity cost, then the strawman always wins the bid. However, if Gaussian noise is added to the CubeSats decision threshold then the strawman wins the bid approximately half the time. This is the model used in this paper.

While the policies of strawman agents are fixed, the other types of agents use evolution to try to maximize an evaluation function. Local agents try to maximize a local fitness evaluation that is a function of only the action of the agent:

$$L = v_i^c - c_i \ , \tag{6}$$

measuring the local net benefit of an observation from CubeSat i without taking into consideration any other observations. Global agents try to maximize the global evaluation functions directly as defined in Equation 5. Difference agents use the difference evaluation defined in Equation 4 applied to the global objective functions:

$$D = \sqrt{\sum_i (v_i^c)^2} - \sqrt{\sum_{i \neq j} v_i^2} - c_i \ . \tag{7}$$

Note that while most of the cost terms in the difference evaluation cancel out, the value terms do not.

5.3 Performance

In our first set of experiments, we test the performance of the five types of agents (R, L, S, G, D) in a noisy environment (zero-mean Gaussian noise with variance of 1.0) with 100 satellites. Figure 4 shows the performance of each evaluation function. In all cases, performance is measured by the same global objective function, regardless of the evaluation function used to evaluate the agents in the system. The figure shows a number of interesting results. Not surprisingly, the strawman bidders do poorly. This bidding strategy is

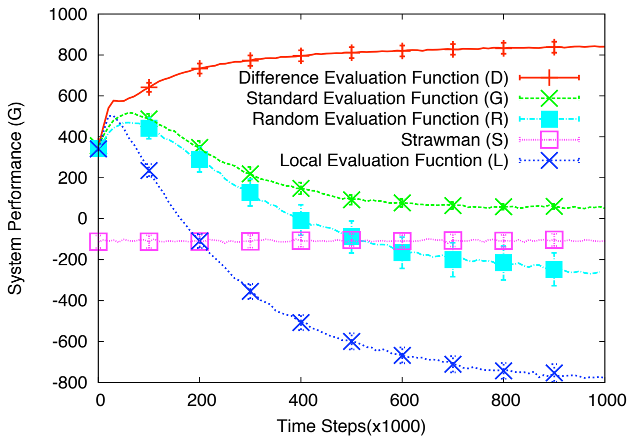

Figure 4: Performance of a 100 CubeSat system for 4 evaluation functions. Difference evaluation function provides the best system-wide performance because it is sensitive to the actions of its agent, yet still aligned with the system evaluation function. Results statistically significant $p < 0.05$.

not trying to maximize any sort of objective function and does not actually look at the total value of the observations purchased. Surprisingly, agents that use evolution to try to maximize a local evaluation function perform even worse. These agents evolve an extremely poor policy because their fitness evaluation function only tells them how they are performing in their local area and do not provide any information on how they impact the system performance. This shows that an agent can be evolving solutions that maximize its local evaluation, while simultaneously harming the global system performance. In particular, the local agents tend to overbid for observations, since they do not take into account the diminishing value of the observations. In contrast, the difference evaluation function, D, allows agents to simultaneously determine how they are impacting their evaluation as well as the system performance. Using D, agents are able to determine how their evaluation is affected by the action they take and how their action impacted the system as a whole. The ability of agents to see the impact of their actions on their own evaluation, as well as the system evaluation enables agents using the difference evaluation to evolve faster and converge to a higher system level performance.

Not surprisingly agents evolving using random evaluations perform poorly. In fact after a small initial increase, their performance actually goes down with time. This occurs because in this particular problem, most good solutions evolve relatively few bids. We chose a sparse initialization of the bidding tables to reflect this. As evolution goes on, agents using random evaluations tend to keep mutating the table, making it less and less sparse. Since the global evaluation does not provide a good measure of performance for an individual agent, an agent cannot readily tell that most mutations lead to worse policies. Agents using the difference evaluation are able to avoid this fate and perform much better.

Note that agents trying to maximize the global evaluation perform poorly, even though in principle the global evaluation function takes into account the value and costs of all the the satellite observations. This lower performance is a result of the low "signal to noise" properties of evolving using the global evaluation directly. In a 100 satellite system, whenever an agent chooses a policy, 99 other agents are choosing a policy at the same time. Even if an agent chooses a good policy, the global evaluation may not go up because some of the 99 other policies chosen by the other agents may have been poor. Therefore a choice of good or bad policy will rarely get the proper credit when the global evaluation is used. Similarly to agents using random evaluations, the performance of these agents tends to go down with time. However, since the global evaluation does give some useful feedback the performance is not quite as bad.

5.4 Scalability

To test whether our system will work with a wide range of satellites constellation, we perform experiments where we vary the number of satellites in the system. Figure 5 shows the performance of each evaluation function over 100 orbits in a noisy environment (zero-mean Gaussian noise with variance of 1.0) for constellation sizes ranging from 1 to 300 satellites. As the size of the constellation increases, an interesting trend emerges: The performance of L continues to grow more and more negative, the performance of G begins to degrade. However, the performance of D continues to increase slowly as the number of satellites within the system is increased. Again, this is because agents evolved using G cannot distinguish their individual impact on the system performance from the impact of other agents present within the system. The problem becomes more pronounced as the number of agents within the system increases. Similarly, as the number of satellites increases, agents evolved using L continue to optimize their own individual evaluation, while ignoring the state of the rest of the system become increasingly problematic. The random agents at least get an even mix of accepted and declined bids, which gives them a poor performance, but is still better than S. The agents using the difference evaluation on the other hand handle the increasing number of satellites well. Because the second term in Equation 4 removes the impact of other agents' actions from agent i's evaluation, increasing the number of satellites does very little to limit the effectiveness of the agent evaluation function. In fact the performance as the system using D even goes up with the number of satellites, as the degrees of freedom are increased. This is a powerful result suggesting that agents evolved using the difference evaluation are well suited to large collectives in this and similar domains where the interaction among the agents prevents both G and L from performing well.

5.5 Robustness

An important property for any system combining a heterogenous array of resources is to be robust against failures and unexpected occurrences. Especially in the CubeSat paradigm with various institutions owning satellites in different states of degradation, it should not be expected that all resources that are bought will actually work. Satellites working beyond their nominal lifespan can fail, or institutional bureaucracies may simply prevent information from being collected. To test that our evolving agent system is robust against these failures, we conduct a series of experiments where 20% of the satellites fail to make their observations, but where failures are refunded so that the average expected cost stays the same. The results shown in Figure 6 shows that our system is highly robust to such failures. In fact the results are almost identical to the experiments with no failures. This adaption to failures happens because policies are being evolved, instead of being implemented in a top-down fashion. In a system with numerous failures, policies are evolved that simply bid for more resources with the assumption that a certain percentage of them will fail.

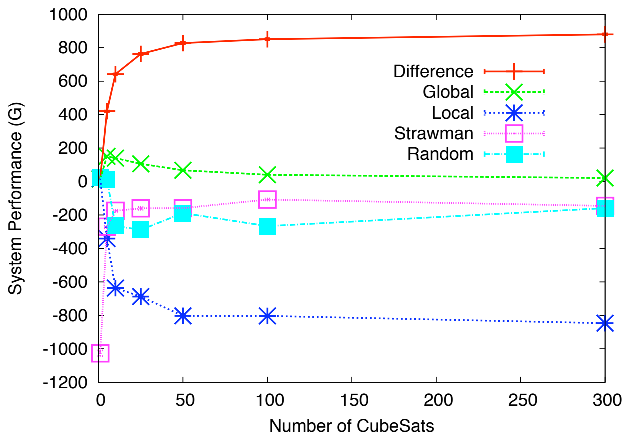

Figure 5: Performance of CubeSat system in a noisy, non-linear environment based upon the number of satellites present within the system. The D evaluation function is superior, allowing the system to maintain performance as the number of satellites within the system increases, even under noisy conditions. Results statistically significant $p < 0.05$.

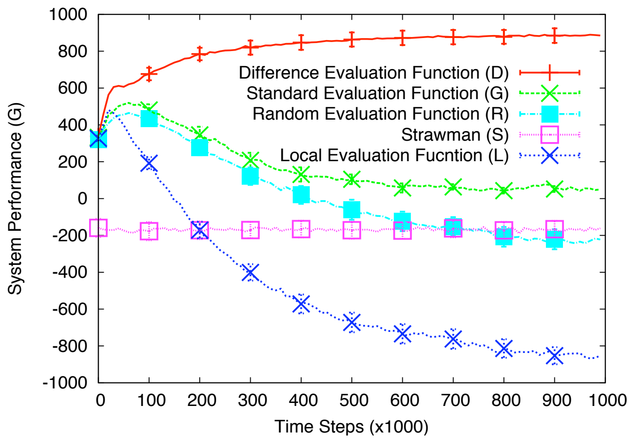

Figure 6: Performance of System with Failures. Here 20% of the satellites that are bid on fail to work. These failures have almost no impact on the system, since agents are able to evolve policies that bid for more resources, assuming some will fail. Results statistically significant $p < 0.05$.

6. DISCUSSION AND CONCLUSION

Evolution, combined with techniques from multiagent systems can play an important role in the new paradigm of

CubeSat missions. Instead of carefully managing the resource from a few expensive satellites in a top-down way, we can collect resources using classical market mechanisms more fitted to a paradigm where there is a large amount of spare capacity of satellites, owned by a growing collection of small institutions. The non-linear nature of collecting information from multiple satellites, and the on-demand nature of collecting this information in real-time, make evolution and multiagent systems a perfect fit.

In this paper, we show how evolution and agents can be used together to allow a customer to purchase observations from a large collection of CubeSats. In this problem, there are two simultaneous tasks: 1) effectively value a single satellite observation in the context of other observations, and 2) obtain these satellite observations in a cost-effective manner. The results show that our system can achieve high levels of performance even in the presence of high levels of noise and satellite failure. By using agents that are evolved with well-designed agent-specific fitness evaluation functions, the system is able to achieve up to a three fold increase in performance over a simple strawman allocation algorithm and up to double the performance of the system using standard evaluation functions (as compared to agents using random evaluations).

While our experimental domain shows the flexibility and robustness of multiagent systems when applied to satellite coordination, we do not intend for it necessarily to be a reference model to how satellite coordination will be implemented. For instance, in real applications, credits may be used instead of monetary exchanges. In addition it is likely that some teams of satellite owners will want to maximize a joint objective among themselves instead of selling resources to a customer. However, in most cases, the coordination algorithm will need to be flexible and robust. Our experiments show that evolution combined with multiagent systems are capable of being effective in this domain, and we believe that they will be used more frequently as small satellite constellations become increasingly common.

7. REFERENCES

[1] O. Abdelkhalik and D. Mortari. Satellite constellation design for earth observation. *15TH AAS/AIAA Space Flight Mechanics Meeting*, 2005.

[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.

[3] D. Baker and S. Worden. The large benefits of small-satellite missions. *EOS, Transactions American Geophysical Union*, 89(33), 2008.

[4] G. Bonnet and C. Tessier. Collaboration among a satellite swarm. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007.

[5] G. Bonnet and C. Tessier. Coordination despite constrained communications: a satellite constellation case. *3rd National Conference on Control Architectures of Robots*, 2008.

[6] S. Damiani, G. Verfaillie, and M. Charmeau. An earth watching satellite constellation: How to manage a team of watching agents with limited communications. *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, 2005.

[7] A. Globus, J. Crawford, J. Lohn, and A. Pryor. Scheduling earth observing satellites with evolutionary algorithms. In *International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, july 2003.

[8] M. Hapgood, S. Eckersley, R. Lundin, M. Kluge, and U. P. an P. Hyvonen. Nano satellite beacons for space weather monitoring. In *In Proceedings of the 5th ESA Round Table on Micro/Nano Technologies for Space*, 2005.

[9] S. Kim, H. Kim, V. Mani, and C. Kim. Genetic algorithm for satellite customer assignment. In I. King, J. Wang, L. Chan, and D. Wang, editors, *Neural Information Processing*, volume 4234 of *Lecture Notes in Computer Science*, pages 964–973. Springer Berlin / Heidelberg, 2006.

[10] B. Klofas, J. Anderson, and K. Leveque. A survey of cubesat communication systems. Technical report, California Polytechnic State University, 2008.

[11] Z. Li, J. Chen, and E. Baltsavias, editors. *Advances in Photogrammetry, Remote Sensing, and Spatial Information Sciences: 2008 ISPRS Congress Book*, London, United Kingdom, 2008. The Taylor Francis Group.

[12] A. Muteanu. Nanosat/cubesat constellation concepts. Masters thesis, Cranfield University, Bedfordshire, UK, 2009. Thesis.

[13] R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems*, 2003.

[14] A. Rogers and L. Paxton. Small satellite constellations for space weather and space environment measurements. *International Astronautical Congress*, 2008.

[15] S. Salcedo-sanz and C. Bousono-calzon. A hybrid neural-genetic algorithm for the frequency assignment problem in satellite communications. *Appl. Intell*, 22:207–217, 2005.

[16] R. Sandau, H. Roser, and A. Valenzuala, editors. *Small Satellite Missions for Earth Observations: New Developments and Trends*, New York, NY, 2010. Springer Heidelburg Dordrecht London.

[17] K. Schilling. Networked distributed pico-satellite systems for earth observation and telecommunication applications. Technical report, Julius-Maximilians Universitat Wurzburg, 2008.

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

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

[20] X. Zhang, L. Ding, and Y. Rao. Qos routing by genetic algorithm for leo satellite networks. In *Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on*, volume 1, pages 341 –344, Dec. 2009.