Experience-Based Trust: Enabling Effective Resource Selection in a Grid Environment*

Nathan Griffiths[1] and Kuo-Ming Chao[2]
[1]Department of Computer Science, University of Warwick,
Coventry, CV4 7AL, UK
nathan@dcs.warwick.ac.uk
[2]School of MIS, Coventry University,
Coventry, CV1 5FB, UK
k.chao@coventry.ac.uk

Abstract

The Grid vision is to allow heterogeneous computational resources to be shared and utilised globally. Grid users are able to submit tasks to remote resources for execution. However, these resources may be unreliable and there is a risk that submitted tasks may fail or cost more than expected. The notion of trust is often used in agent-based systems to manage such risk, and in this paper we apply trust to the problem of resource selection in Grid computing. We propose a number of resource selection algorithms based upon trust, and evaluate their effectiveness in a simulated Grid.

1 Introduction

Distributed computer systems can be viewed as multi-agent systems in which individual autonomous agents cooperate to achieve their objectives. It is through cooperation that such agents are able to function effectively, since they typically lack the knowledge, capabilities or resources to achieve their objectives alone. Grid computing can be viewed as a multi-agent system that allows users to discover, select and use remote resources [15]. The process of a user submitting a task to a resource, and that resource performing the task on the user's behalf, can be viewed as cooperation. Both users and resources have control over their own behaviour, and so are autonomous agents having their own decision making mechanisms. By definition, however, autonomous agents that control their own behaviour also control how they cooperate. In particular, autonomous agents determine for themselves when to initiate cooperation, when to offer it, and when to rescind cooperative commitments. Consequently, when agents

*Research Report CS-RR-409, Department of Computer Science, University of Warwick, December 2004

cooperate, any one of them can cease cooperation at any time, (typically) causing the whole interaction to fail. In a Grid computing environment autonomy manifests itself through users and resources being able to change the nature of their cooperation, or even ceasing to cooperate, at any time. For example, a resource may reduce the priority of a user's task causing it to overrun, or may cease processing one user's task in favour of another.

When entering into cooperation, agents in a multi-agent system are entering into an uncertain interaction in which there is a risk of failure due to the decisions and actions of others. To function effectively agents must manage this risk, and the notion of trust can used to provide a suitable mechanism. In this paper, we describe how the concept of trust can be used to manage cooperative interactions, in the context of Grid computing. In particular, we provide a mechanism for user agents to use trust for resource selection, i.e. to choose the most appropriate resource for a task. Although several researchers have considered the problem of resource selection in a Grid environment from the perspective of minimising cost [3, 16], relatively few have been concerned with minimising task failure [1, 2]. Those that do consider failure tend to use restrictive trust models, and often do not consider the cost of task execution[1].

In the following section, we introduce the Grid context in more detail and illustrate the problems that trust can address. In Section 3 we describe the notion of trust, and the trust model that we use, and in Section 4 we describe how trust can be compared using a strict, stratified, or windowed approach. The application of trust to the resource selection problem is described in Section 5. Our approach has been implemented in a simulated Grid as described in Section 6, and in Section 7 we give details of our results. Finally, in Section 8 we give our conclusions and outline ongoing work.

2 Context: Grid Computing

Grid computing provides a mechanism for users to discover, select and utilise remote resources. Although most existing Grids are fairly localised, the Grid vision is to allow resource sharing on a global scale, where resources charge users for executing their tasks. Resources are heterogeneous, geographically distributed and locally controlled, and have specific individual capabilities in terms of their processor architecture, network bandwidth etc. that are made available for an associated cost. Grid resources may be unreliable and may erroneously fail to execute a task, or circumstances may cause them to delay or drop execution of a remote user's task. An example of the former is where hardware or network problems cause a task to fail. The latter case is illustrated by a resource dropping a task submitted by one user, in favour of that submitted by another user who is considered to be of higher priority. For example, remote users' tasks may be dropped in favour of local users, or tasks submitted by low paying users may be dropped in favour of a user that pays a higher rate.

When determining which resource to submit a task to, a Grid user will typically try to minimise the cost of executing the task (possibly considering time in addition to financial cost) and to minimise the risk of the task failing. These aims may conflict

[1]There has been other work in minimising failure in Grid computing but not from a Grid user's perspective (e.g. [10]).

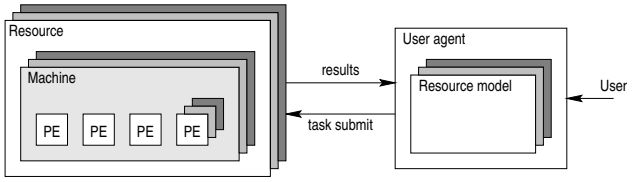

Figure 1: A single user's perspective of the Grid as a multi-agent system.

with each other, since cheap resources may be unreliable and reliable resources may be expensive. Therefore, a user must balance these criteria according to their own preferences. Additionally, users should generally aim to use a broad selection of resources, to avoid reliance on a narrow set of resources. If a user interacts only with a small set of resources that subsequently become unreliable, that user will not have alternative familiar resources to submit tasks to, but must search for alternatives and rebuild resource familiarity.

The factors that a user considers when selecting a resource depend both on its individual decision making mechanism, and on the nature of its individual Grid context. The context is important since it determines, among other things, the significance of failure in terms of cost. In this paper, we assume that all of a users tasks must be completed and failure has an associated cost penalty. Thus, if a resource fails to complete a task the user incurs a cost penalty (i.e. a fine) and must find an alternative resource to submit the task to. The overall cost of processing a task is therefore related to the risk of failure, since in addition to the fundamental execution cost there may be penalties for submitting tasks to unreliable resources.

The Grid environment is illustrated in Figure 1, where resources (which comprise a set of machines that in turn comprise a set of processing elements) are shown as resource agents and human users are represented by user agents. Users submit tasks to resources via their corresponding user agents. All of a user's interactions with resources are performed via its user agent which assists in managing these interactions. In particular, the responsibility of selecting an appropriate resource for a task is devolved to the user agent, which is configured to reflect the user's preferences (e.g. in balancing cost and risk). Our focus in this paper is primarily concerned with the use of trust in resource selection and, although cost is clearly relevant, we are not concerned with the details of specific economic models of Grid computing. Therefore, we adopt a simple flat commodity market model in which resources specify their service price and charge users according to how much resource is consumed [3].

3 Trust

The notion of trust is recognised as a means of assessing the potential risk in interactions [5, 7, 12]. Trust represents an agent's estimate of how likely another agent is to fulfil its cooperative commitments. When considering uncertain interactions, an agent can use its trust of potential cooperative partners to evaluate the risk of failure. There are two main categories of trust: experience-based and recommendation-based.

In the former, an agent's assessment of trust is purely based on its own experiences, while in the latter trust is based on information provided by other agents (possibly supplemented by individual experience). Experience-based trust fits naturally in the Grid context. Users interact with resources and infer trust based on their experiences and, over time, improve their trust models. Recommendation-based trust requires users to share information about their experiences with a resource with other potential users of the resource. Although this is a potentially useful mechanism, there are a number of obstacles to its use for a Grid domain. In particular, there is no intrinsic motivation for information sharing. Indeed, if a user gives positive feedback about a resource this may jeopardise any ability for its future use since others are more likely to use the resource (reducing its availability). There are also general issues to address regarding recommendation-based trust concerning the subjectivity and context-specific nature of feedback. Other researchers are, however, considering these problems to allow utilisation of recommendation-based trust in the Grid domain [9, 14]. Our work is orthogonal to this, and we envisage experience-based and recommendation-based trust being combined in the future to provide a single trust mechanism. For the purposes of this paper, however, we are solely concerned with experience-based trust.

3.1 Trust Framework

We base our model of trust on Gambetta's theoretical work [7], Marsh's formalism [12], and the work of Griffiths [6, 8], and define the trust T in an agent α, to be a real number in the interval between 0 and 1: $T_\alpha \in [0, 1]$. The numbers merely represent comparative values, and have no strong semantic meaning in themselves. Values approaching 0 represent complete distrust, and those approaching 1 represent complete blind trust. There is an inverse relationship between trust and the perceived risk of an interaction: cooperating with a trusted agent has a low perceived risk of failure, while there is a high risk associated with distrusted agents. Trust values represent the view of an individual agent, subjectively based on its experience, and are not directly comparable across agents.

Trust values are associated with a measure of confidence, and as an agent gains experience its confidence increases. With no prior experience, trust takes an initial value according to an agent's disposition: optimistic or pessimistic. Optimistic agents ascribe a high initial trust value to others (implying a low perceived risk), and pessimists ascribe low values. This disposition also determines how trust is updated after interactions [13]. After successful interactions, optimists increase their trust more than pessimists and, conversely, after unsuccessful interactions pessimists decrease their trust more than optimists. An agent's disposition comprises: the initial trust, $T_{initial}$, ascribed prior to interacting and functions for updating trust after successful and unsuccessful interactions, $update_{success}$, and $update_{fail}$ respectively. These functions for updating trust are simple heuristics, and there is no standard definition for them. Instead, it is the responsibility of the system designer to choose an appropriate heuristic. In this paper we use the following definitions for the update functions:

$$update_{success}(T) = T + ((1 - T) \times (d_s \times T))$$

$$update_{fail}(T) = T - ((1 - T) \times (d_f \times T))$$

where d_s and d_f are weighting factors defined by the disposition[2].

Over time trust values may become inaccurate and outdated if the experiences that gave rise to them are no longer relevant. The environment may change, and a resource that was reliable previously may no longer be so. To address this problem, we apply a decay function to converge each trust value to $T_{initial}$ in the lack of any subsequent experience. Thus, unless reinforced by recent cooperative activity, the positive effect of successful interactions reduces over time, as does the negative effect of failed interactions. The decay function is defined as:

$$decay(T) = T - ((T - T_{initial})/d_d)$$

where d_d a decay rate defined by the agent's disposition.

To improve its trust models an agent may also explore alternative resources for low priority tasks. Agents typically only interact with others that are considered trustworthy. Thus, there tends to be a subset of resources that are interacted with, and it is this subset for whom the agent will be confident about its trust values. To reduce the risk of a reliable agent being outside this subset, agents can periodically select resources from outside the set to verify its judgement is correct and attempt to discover additional trusted resources[3].

4 Numerical, Stratified and Windowed Trust

In our approach trust is modelled as a numerical value, however some researchers note that using continuous numerical values can introduce ambiguity since the semantics are hard to represent [1, 12]. The alternative approach is to divide the trust continuum into a set of labelled strata. Abdul-Rahman and Hailes, for example, take this approach, providing four distinct trust strata ("very trustworthy", "trustworthy", "untrustworthy", and "very untrustworthy") that they argue provides a clear semantics for different trust values [1]. The advantage of this approach is that "trustworthy" for one agent should correspond to "trustworthy" for another, avoiding the problem of defining the meaning of a numerical value. However, these semantics are still subjective, and different agents may ascribe the same experiences to different strata; experiences that rate as highly trustworthy for one agent may only rate as trustworthy for another. A further problem with stratifying trust is a loss of sensitivity and accuracy, since comparisons become coarse grained with no way to distinguish between agents within a stratum. For this reason Marsh rejects the use of strata in favour of numerical values [12]. In our approach, to avoid loss of sensitivity and accuracy, we also represent trust by numerical values. Furthermore, updating trust after interactions is simple for numeric values, whilst those models that use stratification gloss over how agents determine a trust strata from their experiences [1, 2].

The advantage of trust strata is that selection between resources is simplified. In our model, suppose that a user must select between two resources with trust values 0.5

[2]It is beyond the scope of this paper to discuss the impact of different dispositions, however our experiments have obtained similar results to those presented in Section 7 with different dispositions.

[3]This approach is analogous to the simulated annealing method for local search, where potentially suboptimal options are considered to avoid local minima [11].

and 0.50001. The user must either conclude that this difference is insignificant and the resources are equally trusted, or that there is a real difference in trust and the latter resource is more trustworthy. The use of strata avoids this problem, since there are no numerical values for consideration. Ideally, a trust model would have the sensitivity and accuracy of a numerical approach, combined with the ease of comparison from a stratified approach. To this end, we propose two simple modifications to our trust model *at the time of trust comparisons*: variably stratified and windowed trust.

In the former, trust values are translated into strata immediately before comparison. The number of strata is variable, with fewer strata providing the simplest but least precise comparison, while more strata reduces the comparison advantage but retains precision. For example, Abdul-Rahman and Hailes' strata can be modelled by dividing the trust interval into the following equal parts,

very trustworthy:	$0.75 <= trust <= 1$
trustworthy:	$0.5 <= trust < 0.75$
untrustworthy:	$0.25 <= trust < 0.5$
very untrustworthy:	$0 <= trust < 0.25$.

The disadvantage of stratifying trust is that an equal difference in values is not significant within a stratum, but is when it lies across strata. For example, using the above strata a difference of 0.1 is not significant for agents with trust values 0.6 and 0.7 (i.e. both are "trustworthy"), but it is for agents with values of 0.7 and 0.8 (i.e. the former is "trustworthy" and the latter "very trustworthy").

Our second modification is to use a variable size window when considering trust values, such that values within the window are considered equal. This is a similar to stratified trust, but avoids the problems across strata boundaries. Windowed trust can be thought of as stratified trust with movable boundaries. For example, if we use a window size, or *trust distance*, of 0.1, then agents with trust values 0.6 and 0.7 are considered equal, as are agents with trust values of 0.7 and 0.8. However, the agent will still distinguish between trust values of 0.6 and 0.8. The trust distance is variable, and similarly to stratified trust, a larger distance provides the simplest but least precise comparison, while smaller values reduce the comparison advantages but retain precision.

5 Resource Selection

When selecting a resource a user must first consider whether the resource has the required capabilities (e.g. processor architecture, network bandwidth etc.), and second whether it is likely to complete the task successfully. The first of these considerations is simple — either a resource has the required capabilities or it does not. Those that do not can be rejected, otherwise factors such as cost and reliability must be considered. These factors are more complex, since they have varying degrees and may be based on uncertain information. Trust enables an agent to consider these factors. A user is likely to prefer a resource that generally completes tasks on time and rarely fails, to one that frequently fails. This is embodied by the trust ascribed to a resource. To select a resource an agent should balance trustworthiness against cost. Selecting a resource

in this manner is a heuristic decision and there are several possible approaches. This paper proposes six fundamental options, as introduced below.

5.1 Cost

The simplest approach to minimising execution cost is to select the resource that has the lowest expected cost, based on its published cost-per-second. However, this will only be successful if resources are honest and reliable. If a resource takes longer than expected to process the task, or fails to execute it, then this approach will not result in the lowest overall cost. If execution takes longer than expected then the user will be charged for the additional processing time. Should task execution fail then an alternative resource must be found, and the user will be charged a penalty for failure. The problem is that the risk of resource failure and unreliability is not considered when selecting a resource.

5.2 Strict Trust

To minimise the risk of failure, an agent can select the most trusted capable resource (provided the agent is confident of its models[4]). If several resources are equally highly trusted, then one is selected arbitrarily. For low priority tasks the agent periodically selects a resource that is not trusted, by way of exploration as described in Section 3. This approach ensures that the resource that is perceived to be the lowest risk is selected. However, the user is forced to distinguish between resources where there is only a small numerical difference in trust. Since trust is solely based on experience, small differences in the numerical value may not be sufficient to make *meaningful* judgements. Although small differences may arise from genuine differences in reliability, they may also arise from slight differences in the extent or recency of prior experience. Thus, there is a risk of overfitting by drawing conclusions from trust values where differences arise from irrelevant artifacts of the data.

A further disadvantage to using strict trust is that it tends to lead to a very narrow set of resources being used. Since users may select resources based on small numerical differences, these small differences are reinforced by the resulting interactions. Over time, an insignificant difference in trust can be magnified significantly. For example, if resource R_1 is trusted only slightly more than R_2 then after a period of successful interactions the trust of R_1 may be significantly more than R_2. Thus, in future interactions R_1 is likely to remain more trusted that R_2 even if in reality there is no significant difference. This is problematic since, as described in Section 2, a broad selection of resource should be used to ensure resilience.

5.3 Stratified Trust

To avoid the narrow resource usage that results from strict trust we can use stratified trust at the comparison stage. The resource that is most trusted according to the trust

[4]If the agent is not confident, it can do little more that choose arbitrarily to gain experience. In the remainder of this section we assume such confidence.

stratum that the numerical value lies in is selected. Where several resources are equally most trusted, one is selected arbitrarily. Again, where the task is of low priority the agent periodically selects a resource that is not trusted by way of exploration. This approach avoids the problem of overfitting small numerical differences, but at the cost of a loss of precision. The number of strata is variable, and the effect of strata size is discussed in Section 7.

5.4 Stratified Trust with Cost

Using stratified trust for resource selection minimises the risk of failure, and avoids overfitting, however no consideration is given to the cost of task execution. Therefore, we propose a fourth selection method that initially uses stratified trust as described above, but then selects the cheapest of the most trusted resources, based on their advertised cost-per-second. Thus, an agent first determines the capable resources, then extracts the most trusted of these, and finally selects the cheapest.

The choice of strata size is crucial to the effectiveness of this approach. If too few strata are used, an agent is unable to distinguish between resources according to reliability; if too many strata are used there is likely to be a single most-trusted resource leaving no opportunity to distinguish based on cost. The number of strata effectively determines the emphasis given to reliability versus cost — more strata emphasises reliability and reduces the influence of cost, and visa versa. Again, the effect of different strata sizes is discussed in Section 7.

5.5 Windowed Trust

Stratified trust avoids the overfitting associated with strict trust, however, as noted in Section 4, it suffers from problems at strata boundaries. The alternative is to use windowed trust, in which the most trusted resource is selected according to the trust window that the numerical value lies in. Where several resources are equally most trusted, one is selected arbitrarily. Again, agents periodically select a resource that is not trusted by way of exploration. This approach avoids both the overfitting problem and the effects of strata boundaries. Similarly to stratified trust, the trust distance (i.e. the window size) is variable, with a larger distance giving higher precision, but at the risk of overfitting. Section 7 discusses the effects of trust distance.

5.6 Windowed Trust with Cost

Our final selection approach combines windowed trust with cost. Similarly to stratified trust with cost, a user first determines the capable resources, then determines the most trusted of these according to windowed trust, and finally selects the cheapest. Again, the size of trust distance is important, with smaller distances emphasising reliability and reducing the influence of cost, and visa versa. The effects of different trust distances are discussed in Section 7.

6 Experimental Scenario

The resource selection mechanisms described above have been explored using the GridSim simulation toolkit [4]. This toolkit provides a feasible way to experiment with such algorithms on a reasonably large-scale Grid of heterogeneous resources. The overheads and resource requirements of investigating these algorithms in a physical Grid are prohibitive, since significant access to Grid resources would be required. Our future aim is to instantiate the most effective of our resource selection mechanisms in a real Grid environment after validation via simulation.

In the GridSim toolkit, resources comprise a set of machines that in turn comprise a set of processing elements (PEs). Each resource has certain capabilities, defined by its communication bandwidth and the configuration and processor architecture of its PEs. At run-time resources have a variable load, due to tasks being processed on behalf of users and due to background processing. Resource load, at least in part, determines how long it takes to process a given task. In GridSim each resource has an associated "calendar", specified when configuring a simulation, that defines how background load varies according to time of day (i.e. peak and off-peak loads), day of the week and holidays etc. We have extended the GridSim toolkit to allow the simulation of unreliable resources, and in our GridSim extension resources have an additional characteristic defining their reliability in terms of a failure rate.

To test the validity of our approach, and the resource selection algorithms described above, a series of experiments were performed. A range of user agents with different dispositions, from optimistic to pessimistic, were used in a selection of Grid contexts, ranging from the majority of resources being reliable, through a mixed set, to the majority of resources being unreliable. Each of the resource selection proposed algorithms was used, with varying numbers of trust strata (from 1 to 5000) and trust distances (from 0.0001 to 1). For control purposes a random selection algorithm was also used, where user agents simply submitted tasks to random resources. Various numbers of users and resources were experimented with, however most of the results presented below are for 30 resources and 10 users. Each user generates a random set of 500 tasks to be completed with varying lengths, bandwidth requirements, and priorities. For each configuration of the experimental domain we performed 10 runs, and the results shown below represent the average values across those runs.

7 Results and Evaluation

In this section we present results obtained from using the resource selection algorithms described above. We are primarily concerned with three factors: failure rate, execution cost, and total cost (i.e. execution cost plus any failure penalties). We begin by first comparing the stratified trust methods with others and second considering the windowed trust methods, in both cases investigating the impact of strata size and trust distance. Finally, we consider the effects of the Grid context in terms of the proportions of reliable and unreliable resources, and attempt to identify the best resource selection algorithms.

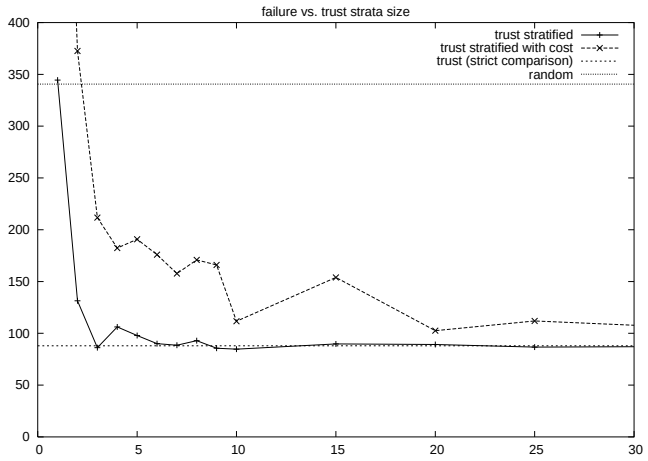

Figure 2: Failure rate for the stratified trust and stratified trust with cost resource selection methods.

7.1 Comparing the Stratified Trust Approach

Figure 2 shows the failure rate for the stratified trust and stratified trust with cost resource selection methods for a varying number of strata. These results represent the average of 10 runs in a "mixed" Grid resource environment, i.e. there is a mix of reliable and unreliable resources[5]. Figure 2 also shows the failure rate for the strict trust and random selection methods. The failure rate for the cost based selection method is not shown, since it is too high (an average of 987) to appear on the graph. It can be seen immediately that the lowest failure rate is achieved for the strict trust method and, with the exception of very low numbers of strata, the random selection method gives the highest failure rate. It can also be seen that provided at least 3 trust strata are used then both strata-based methods perform better than a random selection (and significantly better than a cost-based approach). In both strata-based approaches lower failure rates are achieved as more strata are used. The simple strata-based approach tends to perform as well as the strict trust approach for 10 or more strata, while the strata-based with cost approach is consistently only around 10-15% worse than strict-trust for 20 or more strata (the trace for both methods continues to be flat for 30-5000 strata, but for clarity is omitted from the graph).

Recall from Section 5 that one significant disadvantage of using strict trust for resource selection is that a narrow set of resources tends to be used. In the experiments illustrated in Figure 2, the strict trust approach led to a user generally interacting with a single trusted resource. The strata-based approaches led to a wider range of resources being used, where fewer strata gave a wider set of resources (a single strata leads to all resources being used equally). At the point where the failure rate for the strata-based approaches levels off (around 10 strata without cost and 20 with cost) then, as for strict

[5]It should be noted that the effect of the number of strata is broadly the same regardless of the mix of resources, as discussed later in this section.

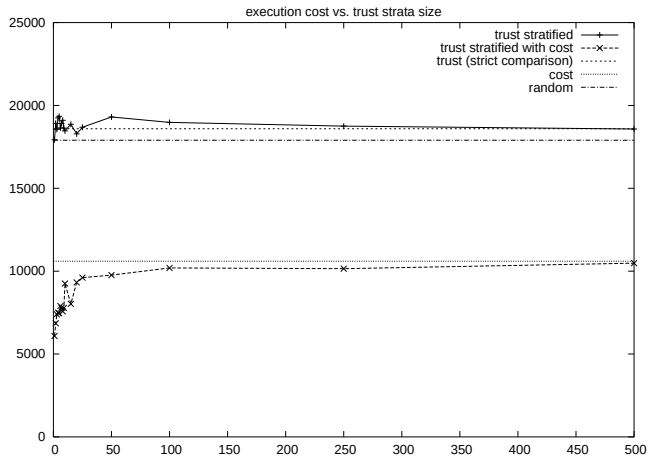

Figure 3: Execution cost for the stratified trust and stratified trust with cost resource selection methods.

trust, there is a single trusted resource being used. However, before this point there is a wider set of used resources, and so a higher resilience to change and a reduced likelihood of negligible differences being interpreted as significant.

The execution cost for the stratified trust methods is shown in Figure 3, along with the execution cost for the cost-based, strict trust and random selection methods. It can be seen that a stratified trust with cost approach gives the least cost, followed by a cost-based approach, and then the random, strict trust, and stratified trust methods respectively. The reason that stratified trust with cost gives a lower execution cost than a pure cost-based approach is that the execution is the *actual* execution cost, and the reliable resources that are favoured by the stratified trust with cost approach are those whose actual execution cost is likely to be closer to the advertised cost.

It can be seen from Figs. 2 and 3 that in general the stratified trust with cost approach gives better results in terms of execution cost than the strict trust, cost-based and random approaches. However, the stratified trust approach gives a lower failure rate but only by around 10-14%.

7.2 Comparing the Windowed Trust Approach

Figure 4 shows the failure rate for the windowed trust and windowed trust with cost selection methods for varying trust distances. Again, results are averaged over 10 runs in "mixed" resource environment. It can be seen that the cost-based approach produces the highest failure rate while a strict trust approach gives the least. For a trust distance of less than 0.35 the windowed trust with cost method gave a lower failure rate than a random approach, while windowed trust was always better. The windowed trust approach gave significantly fewer failures than windowed trust with cost, and for both approaches larger trust distances increased failures. With a trust distance of 1, the windowed trust approach is equivalent to random selection, and the windowed trust

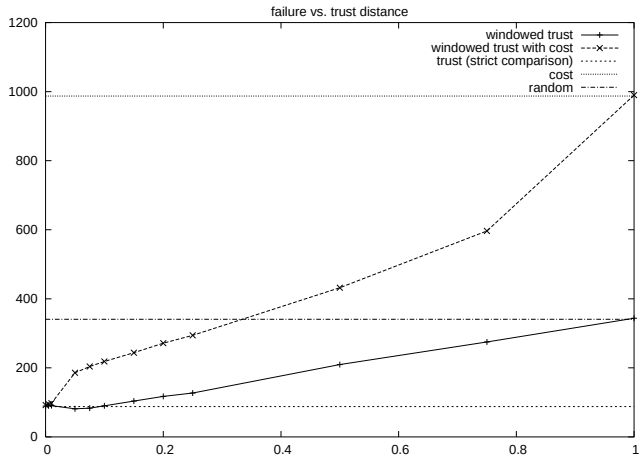

Figure 4: Failure rate of the windowed trust and windowed trust with cost resource selection methods.

with cost approach is equivalent to the cost-based approach.

In a similar manner to the stratified approaches, a larger trust distance leads to more resources being used. For a trust distance of 1, the windowed trust approach uses all resources and the windowed trust with cost uses all cheapest-cost resources. As the trust distance is reduced, so is the set of used resources, such that a very small distance (< 0.001) is equivalent to using strict-trust.

The execution cost for the windowed trust methods is shown in Figure 5, along with the execution cost for the cost-based, strict trust and random selection methods. It can be seen that a cost-based approach gives the least cost, followed by windowed trust with cost, random and strict trust. The windowed trust method is the highest cost for a trust distance of less than 0.25, while for larger window sizes it tends toward the cost of the random approach.

From Figs. 4 and 5 it can be seen that for trust distance of less than 0.35 both windowed trust approaches give a lower failure rate than a random or cost-based approach. The windowed trust with cost method gives a lower execution cost than all alternatives except cost-based, while windowed trust gives a lower failure rate than all except strict trust.

7.3 Resource Setting

The above results illustrate the characteristics of each of the resource allocation methods. However, in order to select the most appropriate method for a given situation we must consider both the nature of that situation (in terms of Grid resources) and the level of penalty imposed for failure. To this end we consider three distinct Grid environments, with differing resource reliabilities. Suppose that there are six levels of reliability, defined as follows.

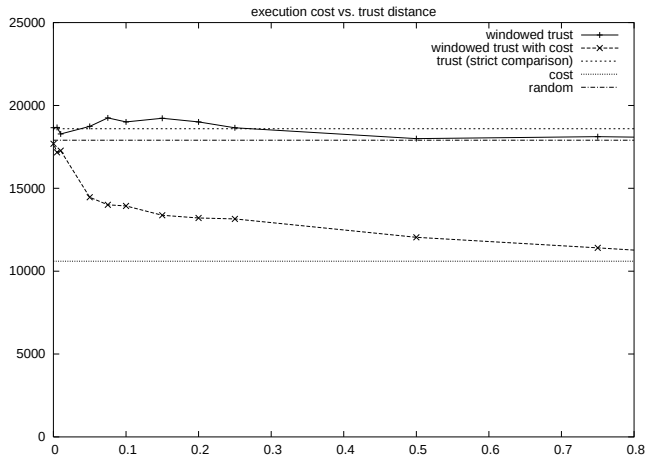

Figure 5: Execution cost of the windowed trust and windowed trust with cost resource selection methods.

reliability	failure rate
highly reliable (HR)	$0 \leq$ failure rate $\leq 3\%$
reliable (R)	$3\% <$ failure rate $\leq 10\%$
marginal unreliable (MU)	$10\% <$ failure rate $\leq 30\%$
unreliable (U)	$30\% <$ failure rate $\leq 60\%$
highly unreliable (HU)	$60\% <$ failure rate $\leq 90\%$
very highly unreliable (VHU)	$90\% <$ failure rate $\leq 100\%$

Using these definitions we define the following three sets of resources, according to the proportions of reliable and unreliable resources present.

	HR	R	MU	U	HU	VHU
reliable	60%	20%	10%	4%	3%	3%
mixed	20%	20%	20%	20%	10%	10%
unreliable	5%	10%	30%	20%	20%	15%

Based on these resource sets, we have experimented with each allocation method (for various strata sizes and trust distances) and obtained the average failure rates, execution costs, and total costs. The failure penalty was set to a cost of 100. Figure 6 shows how the failure rate varies for selected allocation methods according to resource set (reliable, mixed, and unreliable). In the figure, S_x and SC_x represent stratified trust and stratified trust with cost allocation methods with x strata respectively. Similarly, D_x and DC_x represent windowed trust and windowed trust with cost with a trust distance of x. As expected, the failure rate is least with reliable resources, rises in a mixed situation, and is highest where resources are unreliable. Furthermore, as expected from the above discussions, stratified trust and windowed trust have lower failure rates than stratified trust with cost and windowed trust with cost, and increasing the number of strata or decreasing the trust distance also reduces the failure rate.

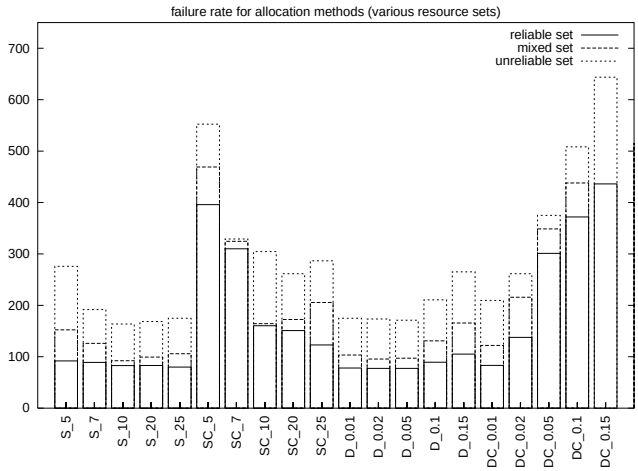

Figure 6: Failure rate for different allocation methods in various resource sets (reliable, mixed, and unreliable).

Figure 7 shows how the total execution cost varies according to resource set[6]. Again, it can be seen that the total cost is smallest where resources are reliable, rises in a mixed situation, and is highest when resources are unreliable. The total cost is determined by both failure rate (in terms of penalties) and execution cost. It can be seen that the lowest cost is obtained by the stratified trust with cost and windowed trust with cost algorithms, but only for certain values of number of strata and trust distances. In particular, the best results are obtained for 10-20 strata or trust distances of 0.01-0.02. Overall, for these values of strata and trust distance there is little to choose between the two approaches; stratified trust with cost is marginally better in a reliable resource context and windowed trust with cost is marginally better in an unreliable context.

8 Conclusions

In a Grid environment resource selection is fundamental to minimising the cost and failure rate that user incurs. We have shown how the notion of trust can be leveraged to implement a suitable resource selection method that minimises both cost and risk. In this paper we have presented several resource selection algorithms based on alternative methods for trust comparison namely, strict, stratified and windowed trust. We have demonstrated the use of these algorithms in a Grid context. Our results show that the use of trust significantly reduces the failure rate encountered by users when utilising Grid resources. Furthermore, where cost is factored into resource selection the overall cost of performing a task is reduced. We have shown that the most effective approaches are stratified trust with cost and windowed trust with cost with moderate

[6]Random and cost-based allocation methods are not included because the total cost is too high. The strict trust method is not included since it is not practical due to the narrow set of resources that are used (although in pure cost terms it performs fairly well). Finally, note that the origin of the y-axis is non-zero.

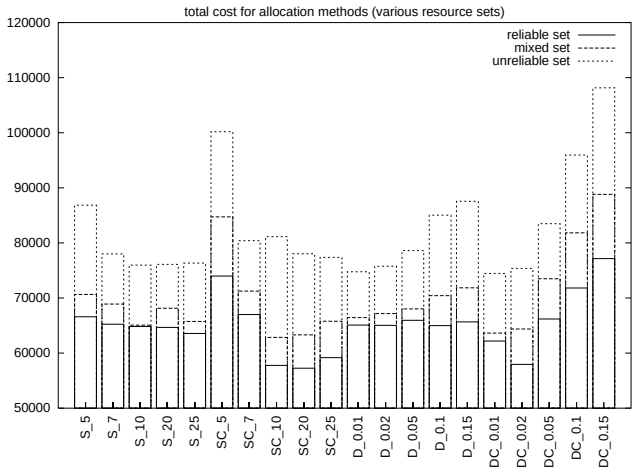

Figure 7: Total execution cost for different allocation methods in various resource sets (reliable, mixed, and unreliable).

strata numbers and trust distances (around 10-20 and 0.01-0.02 respectively). Both of these approaches ensure that users interact with a much wider set of resources than would occur with a strict trust approach, and so they are more resilient to environmental change or incorrect information. Our ongoing work is concerned with investigating further approaches to combining trust and cost information, in particular through the use of fuzzy logic, along with explicitly including the desire to maximise the number of resources used. Future work also includes the instantiation in a real Grid environment of the most effective of our proposed mechanisms, namely stratified trust with cost and windowed trust with cost, Finally, we also intend to apply the trust framework and resource selection algorithms to a peer-to-peer scenario.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the Hawaii International Conference on System Sciences 33*, 2000.

[2] F. Azzedin and M. Maheswaran. Integrating trust into Grid resource management systems. In *Proceedings of the International Conference on Parallel Processing*, pages 47–54, 2002.

[3] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience*, 14:1507–1542, 2002.

[4] R. Buyya and M. Murshed. GridSim: A toolkit for the modelling and simulation of distributed resource management and scheduling for Grid computing. *Journal of Concurrency and Computation: Practice and Experience*, 14(13–15):1–32, 2002.

[5] C. Castelfranchi and R. Falcone. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 72–79, Paris, France, 1998.

[6] J. R. D. Dyson, N. Griffiths, H. N. Lim Choi Jeung, S. A. Jarvis, and G. R. Nudd. Trusting agents for Grid computing. In *Proceedings of the IEEE SMC 2004 International Conference on Systems, Man and Cybernetics*, pages 3187–3192, 2004.

[7] D. Gambetta. Can we trust trust? In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988.

[8] N. Griffiths and M. Luck. Coalition formation through motivation and trust. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems*, pages 17–24, 2003.

[9] T. D. Huynh, N. R. Jennings, and S. Shadbolt. Developing an integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 7th International Workshop on Trust in Agent Societies*, pages 65–74, 2004.

[10] S. Hwang and C. Kesselman. A flexible framework for fault tolerance in the Grid. *Journal of Grid Computing*, 1:251–272, 2003.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[12] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[13] S. Marsh. Optimism and pessimism in trust. In *Proceedings of the Ibero-American Conference on Artificial Intelligence*, 1994.

[14] S. D. Ramchurn, C. Sierra, L. Godo, and N. R. Jennings. A computational trust model for multi-agent interactions based on confidence and reputation. In *Proceedings of the 6th International Workshop of Deception, Fraud and Trust in Agent Societies*, pages 69–75, 2003.

[15] O. F. Rana and L. Moreau. Issues in building agent based computational grids. In *Proceedings of Third Workshop of the UK Special Interest Group on Multi-Agent Systems*, 2000.

[16] K. Subramoniam, M. Maheswaran, and M. Toulouse. A micro-economic model for resource allocation in Grid computing systems. In *IEEE Canadian Conference on Electrical and Computer Engineering*, pages 782–785, 2002.