Optimal Task Migration in Service-Oriented Systems: Algorithms and Mechanisms

Sebastian Stein and **Enrico Gerding** and **Nicholas R. Jennings** [1]

Abstract. In service-oriented systems, such as grids and clouds, users are able to outsource complex computational tasks by procuring resources on demand from remote service providers. As these providers typically display highly heterogeneous performance characteristics, service procurement can be challenging when the consumer is uncertain about the computational requirements of its task *a priori*. Given this, we here argue that the key to addessing this problem is *task migration*, where the consumer can move a partially completed task from one provider to another. We show that doing this optimally is NP-hard, but we also propose two novel algorithms, based on new and established search techniques, that can be used by an intelligent agent to efficiently find the optimal solution in realistic settings. However, these algorithms require full information about the providers' quality of service and costs over time. Critically, as providers are usually self-interested agents, they may lie strategically about these to inflate profits. To address this, we turn to mechanism design and propose a payment scheme that incentivises truthfulness. In empirical experiments, we show that (i) task migration results in an up to 160% improvement in utility, (ii) full information about the providers' costs is necessary to achieve this and (iii) our mechanism requires only a small investment to elicit this information.

1 INTRODUCTION

Service-oriented approaches promise to revolutionise the way computational resources are used and shared in distributed systems [10]. Specifically, emerging cloud, grid and peer-to-peer platforms allow consumers to procure such resources *on demand* to complete highly demanding computational tasks (such as video rendering, climate modelling or difficult optimisation problems). Providers, on the other hand, benefit from selling unused resources, and there is already a plethora of services that are being offered over the Internet.[2]

Now, a key feature in these systems is the highly heterogeneous nature of the available offerings, ranging from cheap (or even free) processing time on idle desktop PCs to the exclusive use of expensive supercomputers. This poses a critical decision problem for the consumer — when choosing a service, its cost and quality need to be balanced with the value and time constraints of the task. Doing this is particularly challenging when there is uncertainty about the computational requirements of the task, which occurs frequently in practice for many well-known computationally hard problems [3, 11].

In this paper, we will argue that the key to addressing this uncertainty is *task migration*, i.e., the process of transferring the current state of a running task from one resource to another [7], thus allowing several providers to contribute sequentially to the same task. When faced with uncertainty, migrating the task allows the consumer to proactively switch to faster resources as the deadline approaches. Additionally, the consumer can use migration opportunistically to exploit low resource costs when these fluctuate over time.

While task migration has received considerable attention in the literature, most of this focuses on balancing and predicting loads in closed systems, such as data centres or clusters [7, 2]. This is fundamentally different from the emerging open systems we consider, where resources are offered by self-interested agents that are not necessarily cooperative and that demand financial remuneration. First, it is necessary to reason about the costs of resources and balance this with the value and inherent uncertainty in the consumer's task. Second, it is vital to consider the incentives of the participants and address the possibility that they lie about their capabilities, for example by inflating costs to increase profits or by overstating their speeds.

Some of these issues are addressed by [5], who consider uncertainty in the behaviour of service providers and propose a mechanism to incentivise them to be truthful. However, they do not examine task migration or settings where the consumer is able to flexibly buy processing time (this is often known as *infrastructure as a service* rather than *software as a service*, where functionality is provided without control over the underlying implementation). Furthermore, they do not discuss potentially strategic behaviour by the consumer.

We address these shortcomings in this paper, and, more specifically, make the following three contributions to the state of the art. First, we prove that optimal task migration is an NP-hard problem. Second, we employ both novel and established search algorithms and analytical techniques to design two algorithms that can be used by an intelligent agent to optimally migrate tasks. Respectively, these deal with settings where the task can be processed at a given provider for any arbitrary amount of time, and where processing time is sold in discrete time slots. Third, we propose a payment mechanism that can be used by an intermediary agent to achieve optimal task migration in practice. Specifically, this mechanism is *efficient*, i.e., implements the optimal migration strategy, *incentive-compatible*, i.e., incentivises all participants to reveal their private information truthfully, and *individually rational*, i.e., both providers and the consumer expect to benefit.

Additionally, we evaluate our techniques empirically and show that task migration offers a significant benefit, leading to an up to 160% improvement in utility. We also demonstrate that full information about cost profiles allows the consumer to further benefit by using the most cost-effective provider at any time. Finally, we show that our proposed intermediary mechanism requires only a small investment in order to incentivise truthfulness.

In the following, we first formalise our model (Section 2) and then consider optimal migration (Section 3). In Section 4, we describe our mechanism and evaluate our work in Section 5. Section 6 concludes.

[1] University of Southampton, UK, email: {ss2,eg,nrj}@ecs.soton.ac.uk

[2] See, e.g., `aws.amazon.com/ec2`, `code.google.com/appengine` or `www.microsoft.com/windowsazure`.

2 SYSTEM MODEL

We consider a setting where a service consumer faces a computational *task* of uncertain *difficulty*, which corresponds to the number of computational cycles[3] required to complete it. Prior to completion, the consumer has a probability distribution over this difficulty, where we use F to denote its cumulative distribution function and f the corresponding density function. Thus, $F(x)$ is the probability that the task requires at most x cycles. Furthermore, the task has a *deadline* t_d and a *value* $V > 0$, which the consumer only receives if the task is completed and the results are returned within the deadline.

We assume there are n service *providers* that are able to work on the task on the consumer's behalf. Each provider $i \in \{1, 2, \ldots, n\}$ has a *quality of service* (QoS) q_i, which denotes the number of cycles it can execute in one time unit. While working on the task, the provider also incurs a cost. This represents both the running cost for the service, but also, and typically more importantly, the opportunity cost that arises from offering the resource to the consumer rather than use it for other purposes, including selling it to a different customer. As these costs can vary significantly over time, for example, during peak hours or when the provider requires the resource for its own computations, we represent this as a *cost profile* c_i, which maps time to a cost per time unit.[4] Using this, the cost for running the resource from time a to b is calculated as $\int_a^b c_i(t)\mathrm{d}t$.

Crucially, a running task can be interrupted at a certain provider i and *migrated* to another provider j, which then continues execution. This may happen several times during task execution, but to reflect network charges, the consumer incurs a monetary cost of $m_{i\to j}$ for each such migration.[5] These costs are also incurred when initially transferring the task to the first provider and when eventually receiving the results after the required number of cycles have been executed (using $i = 0$ to denote the consumer).

Given this formalisation, we represent the consumer's strategy for executing and migrating its task as a *migration schedule* $\rho = \langle \bar{s}, \bar{t} \rangle$, where $\bar{s}$ is a vector of service providers, and $\bar{t}$ is a vector denoting the *maximum execution times* at each provider i in the schedule before migrating to $i+1$.[6] We use $\eta = |\bar{s}| = |\bar{t}|$ to denote the number of providers in the schedule, and, w.l.o.g., we assume that the sum of execution times does not exceed the deadline, i.e., $\sum_{1 \le i \le \eta} t_i \le t_d$.

Next, we will describe in more detail the utility functions of the participants in these systems (assuming they are risk neutral). In doing so, we cover the two possible ways in which services may be provisioned in practice: continuous-time and discrete-time provisioning.

We first look at ***continuous-time service provisioning***, where the consumer can execute the task for *any* arbitrary amount of time at a given provider. When this applies, only costs for the actual time the task is executing are incurred, and the results are transferred back to the consumer as soon as the required number of cycles has been reached. Here, the scheduled times t_i may take on any values from $\mathbb{R}^+$, and so the consumer's expected utility prior to execution is:

$$\mathrm{E}[U_C(\rho)] = F\left(\sum_{i=1}^{\eta} t_i q_i\right) \cdot V - \sum_{i=1}^{\eta}\left(1 - F\left(\sum_{j=1}^{i-1} t_j q_j\right)\right) \cdot m_{i-1\to i} - \sum_{i=1}^{\eta}\left(F\left(\sum_{j=1}^{i} t_j q_j\right) - F\left(\sum_{j=1}^{i-1} t_j q_j\right)\right) \cdot m_{i\to 0} \quad (1)$$

where the first term is the expected reward from completing the task successfully, while the second and third terms comprise the total migration costs between providers and back to the consumer, respectively. Similarly, the expected utility for a given provider i is:

$$\mathrm{E}[U_i(\rho)] = -\int_0^{t_i} q_i \cdot f\left(x q_i + \sum_{j=1}^{i-1} t_j q_j\right) \cdot c_i\left(x + \sum_{j=1}^{i-1} t_j\right) - \left(1 - F\left(\sum_{j=1}^{i} t_j q_j\right)\right) \cdot c_i\left(x + \sum_{j=1}^{i-1} t_j\right) \mathrm{d}x \quad (2)$$

where the first term integrates over the cost when the task is completed at provider i (and thus a partial cost is incurred), while the second term adds the full cost when the task is not completed after t_i time units at provider i. While this service provisioning mechanism may offer more flexibility, it is rather uncommon in reality, mostly for practical reasons. Instead, services are typically allocated in fixed-size time slots, which eases the administrative burden on both providers and consumers.

In these ***discrete-time service provisioning*** mechanisms, execution time can only be allocated to the consumer in multiples of some fixed time slots Δt, i.e., it must hold that $t_i = x \cdot \Delta t$, where x is an integer. This is common in many real cloud or grid settings, where processing time is allocated in hourly slots. In these settings, once started, the resource is reserved for the user for the full time slot, and so the entire cost is incurred even if the task completes half-way through execution. For conciseness, we assume that Δt is a system-wide parameter and that migration may only take place at the end of a time slot. In these settings, the consumer's expected utility is still given by Equation 1, but $\mathrm{E}[U_i(\rho)]$ is now obtained by simply summing the expected costs of each time slot. This follows from Equation 1 (without the need to integrate over f).

Finally, for both the continuous and discrete provisioning scenarios, we define the expected *social welfare* of a migration schedule as the sum of all the participants' utilities:

$$\mathrm{E}[W(\rho)] = \mathrm{E}[U_C(\rho)] + \sum_{i=1}^{\eta} \mathrm{E}[U_i(\rho)] \quad (3)$$

In the following, we will be interested in finding the *optimal* migration schedule ρ^* that maximises the social welfare, i.e., $\rho^* = \mathrm{argmax}_{\rho} \mathrm{E}[W(\rho)]$. We focus our attention on this schedule, as it maximises the expected difference between the value of completing the task and the total costs incurred and so naturally represents a solution that best uses the available resources to complete the task.

3 OPTIMAL TASK MIGRATION

In this section, we consider the problem of finding the optimal migration schedule ρ^*. Initially, in Section 3.1, we characterise the computational complexity of this problem and then we describe two algorithms for solving it — the first, in Section 3.2, deals with continuous-time provisioning for a particular problem distribution, while the second, in Section 3.3, presents a general algorithm for any distribution when discrete provisioning is employed.

3.1 Problem Hardness

Unfortunately, finding the optimal migration schedule is a computationally hard problem, as we show in the following.

Theorem 1 OPTIMAL TASK MIGRATION [OTM] *is NP-hard.*

Proof. We prove this by providing a polynomial time transformation from an instance of the NP-complete KNAPSACK problem to an instance of OTM. The proof applies to both continuous and discrete provisioning. First, we consider an instance of KNAPSACK: we let $I = \{1, 2, \ldots, k\}$ be a set of items, w_i the weight of item i and v_i its value. The capacity is C and the target value is T. To transform this to an instance of OTM, we create one service provider for each item i. For each such provider i, we set its quality of service as $q_i = v_i$ and define its cost profile c_i such that $c_i(x) = 0$ if $i-1 \le x \le i$, and

[3] In practice, this might be measured in floating point operations (FLOP).

[4] When costs are subject to uncertainty, these correspond to *expected* costs.

[5] We assume migration costs are subadditive, i.e., $\forall i, j, k : m_{i\to j} \le m_{i\to k} + m_{k\to j}$, since we could simply route through intermediate nodes.

[6] For conciseness and w.l.o.g., we assume that $s_i = i$, unless noted otherwise.

$c_i(x) = \infty$ otherwise. We also define an additional provider $k+1$ with $q_{k+1} = 0$ and $c_{k+1}(x) = 0$ that corresponds to not choosing an item. Next, we define the migration costs such that migrating to a provider i (with $1 \leq i \leq k$) costs w_i ($m_{j \to i} = w_i$), while migrating to provider $k+1$ or back to the consumer is free ($m_{i \to k+1} = 0$, $m_{0 \to k+1} = 0$ and $m_{i \to 0} = 0$). Finally, we set the value of the task to $V = C + \epsilon$, where $\epsilon < \min_i w_i$, its deadline to $t_d = k$, and we set F, such that the difficulty is always T. This transformation can be performed in polynomial time, and it is straight-forward to see that the original instance of the KNAPSACK problem is satisfied if and only if the solution to the constructed OTM instance contains at least one provider other than $k+1$. $\square$

Despite this hardness result, we will propose two algorithms for solving OTM that work well in practice and that apply to the two different provisioning mechanisms outlined earlier.

3.2 Continuous-Time Provisioning

First, we consider continuous-time provisioning. Now, to solve OTM optimally in this case, we need to make some assumptions about the difficulty distribution of the task. For the purpose of this paper, we will assume that the difficulty follows an exponential distribution with rate parameter λ, i.e., $F(x) = 1 - \mathrm{e}^{-\lambda x}$ and $f(x) = \lambda \mathrm{e}^{-\lambda x}$. We chose this particular distribution, as there is evidence that the run-time of computionally difficult tasks often follows this in practice [11]. For the purpose of this section, we will also make the simplifying assumption that the cost of each provider does not fluctuate over time — which we denote by defining $c_i(x) = k_i$ for some constant cost k_i. We make these assumptions to keep the calculations manageable, but we note that we will consider arbitrary costs and distributions for the discrete mechanism in Section 3.3.

Given this, we can now calculate the expected social welfare as follows (using Equations 1, 2 and 3):

$$\mathrm{E}[W(\rho)] = \sum_{1=i}^{\eta} \mathrm{e}^{-\lambda \sum_{j=1}^{i-1} t_j q_j} \cdot \left(\left(V - m_{i \to 0} - \frac{k_i}{\lambda q_i} \right) \left(1 - \mathrm{e}^{-\lambda t_i q_i} \right) - m_{i-1 \to i} \right) \quad (4)$$

To find ρ^* that maximises this, we first show how to find the optimal processing times, $\overline{t}$, given a particular sequence of providers, $\overline{s}$. Here, we can use Lagrange multipliers to maximise Equation 4 subject to the deadline constraint. Due to space reasons, we omit the details and simply show the resulting closed form solution for all t_x with $x > 1$:

$$t_x = -\frac{1}{\lambda q_x} \ln \left(\frac{\frac{k_{x-1}}{\lambda} - q_{x-1} \left(m_{x-1 \to x} + m_{x \to 0} - m_{x-1 \to 0} + \frac{k_x}{\lambda q_x} \right)}{(q_{x-1} - q_x) \left(V - m_{x \to 0} - \frac{k_x}{\lambda q_x} - S_x \right)} \right) \quad (5)$$

$$\text{with } S_x = \sum_{i=x+1}^{\eta} \mathrm{e}^{-\lambda \sum_{j=x+1}^{i-1} t_j q_j} \cdot \left(\left(V - m_{i \to 0} - \frac{k_i}{\lambda q_i} \right) \left(1 - \mathrm{e}^{-\lambda t_i q_i} \right) - m_{i-1 \to i} \right) \quad (6)$$

The first processing time, t_1, is obtained as $t_1 = t_d - \sum_{i=2}^{\eta} t_i$. It is important to note here that each t_x only depends on the times of providers following x in the schedule. Thus, all times can be calculated efficiently using backwards induction.

This leaves the problem of determining the optimal *ordering* of providers, $\overline{s}$. However, we can exploit some characteristics of the optimal solution to design an algorithm that is fast in practice:[7]

1. The optimal solution never contains a dominated provider (i is dominated by j if $(k_i \geq k_j \wedge q_i < q_j) \vee (k_i > k_j \wedge q_i \leq q_j)$).
2. It is never optimal to migrate to a slower provider (i.e., $q_{i+1} \geq q_i$).
3. When Equation 5 is negative or cannot be solved for a particular x in a given schedule $\overline{s}$, this indicates that the optimal t_x is 0. Thus, the provider can be removed from $\overline{s}$ without decreasing the expected utility. Furthermore, since t_x depends only on the characteristics of $s_{x-1}, s_x, \ldots, s_\eta$, there must be an optimal schedule that does not *end* with these providers.

Algorithm 1 Continuous-Time Algorithm.

```
 1: $P \leftarrow$ FILTERDOMINATED                      ▷ Prune providers using item 1
 2: $\langle \overline{s}^*, \overline{t}^* \rangle \leftarrow \langle \langle\rangle, \langle\rangle \rangle$   ▷ Best ordering and times so far
 3: $Q \leftarrow \langle \overline{s}^* \rangle$                       ▷ Unexpanded orderings
 4: while $Q \neq \langle\rangle$ do                      ▷ More unexpanded?
 5:     $\overline{s} \leftarrow$ REMOVEHEAD($Q$)           ▷ Remove first element of $Q$
 6:     $\overline{t} \leftarrow$ OPTIMALTIMES($\overline{s}$)         ▷ Calculate times for $\overline{s}$
 7:     if FEASIBLE($\overline{t}$) then                ▷ Prune using item 3
 8:         for all $i \in P$ do                  ▷ Consider all providers
 9:             if $q_i < q_{s_1}$ then              ▷ Prune using item 2
10:                 $Q \leftarrow Q \oplus (\langle i \rangle \oplus \overline{s})$        ▷ Add new ordering
11:         if $\mathrm{E}[w(\langle \overline{s}, \overline{t} \rangle)] > \mathrm{E}[w(\langle \overline{s}^*, \overline{t}^* \rangle)]$ then   ▷ Best so far?
12:             $\langle \overline{s}^*, \overline{t}^* \rangle \leftarrow \langle \overline{s}, \overline{t} \rangle$      ▷ Update current best
13: return $\langle \overline{s}^*, \overline{t}^* \rangle$                  ▷ Return optimal
```

Using these, we search the space of possible orderings, discarding any that cannot be optimal. Critically, we perform this search by considering the last provider first and then build up the migration schedule from back to front. Doing this allows us to exploit item 3 above and prune all orderings that end with infeasible providers.

The full details are given in Algorithm 1.[8] In practice, the algorithm finds a solution for realistic settings with hundreds of providers in seconds on a standard PC. In the following, we now turn to the discrete-time provisioning mechanism and present an algorithm that can be used for arbitrary distributions and fluctuating prices.

3.3 Discrete-Time Provisioning

Due to the discrete time slots present here and the limited size of the state space, this problem is a natural candidate for *dynamic programming* [4]. Thus, we characterise the state of a task as $S = \langle d, t, i \rangle$, where d is the difficulty achieved so far, t is the elapsed time and i is the provider where the task is currently executing. Given this, we recursively define the *optimal* expected welfare achievable in state S:

$$\mathrm{E}[w^*(S)] = \begin{cases} 0 & \text{if } t + \Delta t > t_d \\ \max(0, \max_j (\mathrm{E}[w_j^*(S)])) & \text{otherwise} \end{cases} \quad (7)$$

where $\mathrm{E}[w_j^*(S)]$ is the optimal expected welfare achievable in state S *given* that the task is next executed at provider j:

$$\mathrm{E}[w_j^*(\langle d, t, i \rangle)] = -m_{i \to j} - \int_t^{t+\Delta t} c_j(x) \mathrm{d}x + F_j(d) \cdot (V - m_{j \to 0}) + (1 - F_j(d)) \cdot \mathrm{E}[w^*(\langle d + q_j \Delta t, t + \Delta t, j \rangle)] \quad (8)$$

where $F_j(d)$ is the conditional probability that provider j will successfully complete the task within the time interval Δt, given that d cycles have already been executed without completing the task, i.e., $F_j(d) = (F(d + q_j \Delta t) - F(d))/(1 - F(d))$. In the special case where $F(d) = 1$, we let $F_j(d) = 0$. We also assume here that $m_{i \to i} = 0$, which occurs when the task is not migrated.

Using this recursive formulation, we can find the optimal migration schedule by finding the solution to $\mathrm{E}[w^*(\langle 0, 0, 0 \rangle)]$ and noting the chosen decision variables j in Equation 7. The resulting list of providers indicates the optimal sequence of providers to use for each time slot. We omit a detailed listing of our algorithm to solve this here, as it follows standard dynamic programming practices (we first identify reachable states and then compute Equation 7 for all such states in a bottom-up manner starting from the deadline and then working backwards in time). Clearly, as with all dynamic programming approaches, the efficiency depends on how well the problem can be discretised. Specifically, the number of states to evaluate is bounded by $n \cdot \left(t^2 + t \right)/2 \cdot \max_i(q_i) / \gcd(q_1, q_2, \ldots, q_n)$, where

[7] Intuitively, these hold due to the time invariance of the exponential function.

[8] In the algorithm, $\oplus$ denotes concatenation, such that $\langle a \rangle \oplus \langle b \rangle = \langle a, b \rangle$.

n is the number of providers, $t = \lfloor t_d/\Delta t \rfloor$ the number of full time slots that can be utilised before the deadline and $\gcd(q_1, q_2, \ldots, q_n)$ is the greatest common divisor of all q_i. Yet, in practice, even large problems can be solved quickly. For example, when the deadline is $t_d = 24$ hours, a single time slot is $\Delta t = 1$ hour, there are 50 potential providers, and the values for q_i range in integer steps from 1 to 100, then the algorithm considers up to 1.5 million states. On a modern PC, this is solved in seconds.

To conlude this section, we note that both algorithms presented here cover a wide range of realistic settings and can efficiently compute optimal migration schedules when there are dozens or even hundreds of providers. However, we have assumed full information about the providers and that these willingly offer their services. In the next section, we address the more realistic case where participants are self-interested and might strategise about the information they reveal.

4 INCENTIVISING TRUTHFULNESS

In order to calculate the optimal migration schedule we have so far assumed that the consumer has access to complete information about the quality of service (QoS) and cost profiles of the providers. In practice, this information needs to be elicited and a strategic provider may misreport if this results in a higher expected utility. To this end, we turn to the field of mechanism design to find appropriate payments that incentivise the service providers to reveal their information truthfully. In this context, a well known mechanism is Vickrey-Clarke-Groves (VCG) [8]. The main advantage of VCG is that the resulting allocation is *efficient*, i.e., it maximises the social welfare in the system. However, as we will show, the VCG mechanism only works in our setting if we know or can verify the providers' quality of service, and the consumer is truthful about the properties of the task (i.e., the task difficulty distribution and the value it derives if the task is completed). To address the setting in which both sides can misreport, we first need to introduce a trusted intermediary agent (henceforth called the *centre*) who calculates and enforces the payments. Furthermore, we need to design appropriate payments to incentivise both sides. In the following, we first apply the VCG mechanism when the consumer is assumed to be truthful and the QoS is known. We then proceed to the setting where the QoS also needs to be elicited. Finally, we assume that the consumer is also strategic and consider the elicitation problem on both sides.

4.1 Strategic Providers and Known QoS

We first consider the case where a provider can only misreport its cost profile.[9] In this case, we can use the standard VCG mechanism, which calculates the payments to a service provider based on the *marginal contribution* of that provider [8]. Specifically, for our setting, the VCG payments or *transfers* to each provider i are given by:

$$\tau_i = \mathrm{E}[W_{-i}(\rho^*)] - \mathrm{E}[W(\rho^*_{-i})], \tag{9}$$

where ρ^* is the optimal migration schedule, ρ^*_{-i} is the optimal schedule if provider i did not exist, and $\mathrm{E}[W_{-i}(\rho^*)] = \mathrm{E}[W(\rho^*)] - \mathrm{E}[U_i(\rho^*)]$ is the social welfare excluding the expected costs incurred by i (but including the provider in the schedule ρ^* and its impact on the other agents' expected utilities). In words, the transfers to provider i are equal to the difference between the social welfare excluding the *costs* of i, and the social welfare when excluding i altogether. This is also known as its *marginal contribution* to the social welfare.

VCG has a number of desirable properties. First, it is *incentive compatible in dominant strategies*, which means that a provider is always (weakly) better off revealing its true cost profile, irrespective of the reports of others. Intuitively, it can be seen that, for this to hold, it is necessary that the payment does not depend on the (reported) costs of provider i, which is the case for Equation 9.[10] Second, the mechanism is *individually rational*, which means that the provider will always be better off (in expectation) participating than not. Note that this requires $\tau_i + \mathrm{E}[U_i(\rho^*)]$ to be positive always, and it is easy to verify that this is indeed the case.

4.2 Strategic Providers and Unknown QoS

If the service provider is also asked to report its QoS and this cannot be verified by the centre, then the VCG mechanism is no longer incentive compatible. To see this, note that the service provider can artificially inflate the expected social welfare, $\mathrm{E}[W_{-i}(\rho^*)]$ in Equation 9, by reporting a higher QoS, resulting in a higher payment (even if the optimal migration schedule remains unchanged).

Now, this problem can be avoided by calculating the payments based on the *actual* utilities of the other agents *after* execution, rather than using the *expected* utilities. This so-called execution-contingent VCG has been successfully applied to address similar problems, e.g., in [9, 5]. Specifically, the payments are here calculated as follows. Let $\rho' = \langle \overline{s}', \overline{t}' \rangle$ denote the executed schedule, where $\overline{s}'$ contains the service providers that have actually been used, and $\overline{t}'$ the actual time that they spent executing the task. Furthermore, let $\eta' = |\overline{s}'|$. Then the actual utility of the ith provider in the schedule is given by: $U_i(\rho') = -\int_0^{t'_i} c_i(x + \sum_{j=1}^{i-1} t'_j)\mathrm{d}x$. Furthermore, the utility of the consumer is $U_C(\rho') = V - \sum_{i=1}^{\eta'} m_{i-1\to i} - m_{\eta\to 0}$ if the task has succeeded, and $U_C(\rho') = -\sum_{i=1}^{\eta'} m_{i-1\to i}$ otherwise. Then the payment to i is given by:

$$\tau_i = \left[U_c(\rho') + \sum_{j\in\{1,\eta'\}\setminus i} U_j(\rho') \right] - \mathrm{E}[W(\rho^*_{-i})] \tag{10}$$

Note that the *expected* payment, $E[\tau_i]$, is identical to Equation 9 (given that the providers are truthful). However, Equation 10 no longer relies on the reported QoS values. It is straightforward to show that Equation 10 incentivises the providers to report their private information (including the QoS) truthfully. Essentially, this holds because doing so results in the optimal migration schedule, which in turn results (in expectation) in the highest payment to the providers. The fact that the migration schedule is calculated optimally is important, otherwise incentive compatibility is generally lost.[11] In the case that providers are asked to report their QoS, however, this requires all the providers in the system to be truthful. Now, since the optimal response of a provider depends on the fact that others are truthful as well, this means that the mechanism is no longer incentive compatible in *dominant strategies*, but rather in *ex post* implementation, which is a slightly weaker solution concept [1]. Furthermore, note that payments can be negative (e.g., when the task fails, payments will always be negative). Therefore, providers bear some of the risk, but, *in expectation*, payments are always positive. Therefore, given the assumption of risk neutrality, individual rationality still holds.

4.3 Strategic Providers and a Strategic Consumer

We now turn to the problem of the strategic consumer, who may misreport his valuation V, deadline t_d and the difficulty distribution F to the centre. A naïve approach is to simply let the consumer pay the sum of the transfers to the providers. However, it is easy to see that, since these payments depend on the reports of the consumer, this is not incentive compatible (e.g., the consumer may report a lower valuation in an attempt to lower these payments). If we instead apply the

[9] The QoS could be verified by the centre during execution, e.g., by modifying the task code to sample the speed of the processor. In this case, a penalty could be imposed (either monetary or virtual by using a reputation mechanism) if the observed QoS does not correspond to the reported QoS.

[10] The formal requirement is *monotonicity*, of which bid independence is a consequence. See, e.g., [6] for details.

[11] We note that our mechanism can be extended for certain suboptimal solutions, similar to those reported in [5].

VCG mechanism as in Section 4.1, the marginal contribution of the consumer becomes: $\mathrm{E}[W(\rho^*)] - \mathrm{E}[U_C(\rho^*)] - 0 = \sum_{i=1}^{\eta'} \mathrm{E}[U_i(\rho^*)]$ (noting that the social welfare without the consumer is 0). This is the sum of the expected utilities of the providers (which is always negative). Now, using the standard VCG, a consumer still has an incentive to misreport the problem distribution, since the payments are calculated based on *expected* utilities. For example, if the consumer reports that the task will finish quickly with high probability, then the expected utility of providers following the first one are likely to be close to zero, resulting in low payments for the consumer. Again, this problem can be solved by using the execution-contingent VCG. Then the (negative) transfers to the consumer simply become:

$$\tau_C = \sum_{i=1}^{\eta'} U_i(\rho') \tag{11}$$

We summarise the main properties in the following theorem:

Theorem 2 *For a given optimal schedule ρ^*, the transfers calculated by Equations 10 and 11 are ex-post incentive compatible (i.e., given that others are truthful) and individually rational w.r.t. the providers and the consumer.*

Although the (execution-contingent) VCG mechanism is efficient, note that the payments to the service providers do not correspond to the payments received by the centre from the consumer. That is, the mechanism is not *budget balanced*. In fact, while the consumer pays the true costs incurred by the providers, the centre has to pay them slightly more to elicit this information truthfully. Therefore, the centre has to *subsidise* the market.[12] This budget deficit could be recovered by charging a fixed subscription fee to consumers, providers, or both. Alternatively, if the mechanism is deployed within a company or by the government, they may be willing to pay the mechanism in return for obtaining an efficient market. In the next section, we will empirically evaluate the mechanism and measure the size of the budget deficit. We show that the deficit is small compared to the overall costs when there is sufficient competition.

5 EMPIRICAL EVALUATION

In this section, we evaluate our approach empirically by simulating a large range of distributed systems and realistic task distributions. To this end, we first describe our experimental setup (Section 5.1) and then outline a number of benchmark strategies (Section 5.2). In Sections 5.3 and 5.4, we show our results for the continuous and discrete provisioning settings, respectively.

5.1 Experimental Setup

We test our approaches over a wide range of settings to represent possible scenarios that may be encountered in reality. In doing this, we vary a number of system parameters and measure the expected social welfare obtained by each strategy (and, where applicable, also the expected utility of the intermediary agent, or centre).[13]

For consistency, we keep certain variables of the simulation fixed throughout this section (the trends continue to hold for other choices). First, the consumer faces a task with value $V = \$100$ and deadline $t_d = 24$ hours. We vary the distribution F, from which the difficulty of the task is drawn, but we generally choose one with a mean of around 100 (this could represent $100 \cdot 10^{15}$ CPU cycles).

We assume that each provider owns one of two possible resource types, chosen at random: cheap, slow *desktop PCs* (80%) or expensive, fast *supercomputers* (20%). Each of the former has a quality of service q_i that is drawn from the discrete uniform distribution $\mathcal{U}_d(1, 5)$, while each of the latter's q_i is drawn from $\mathcal{U}_d(5, 100)$. To generate costs, we let $\hat{c}_i$ denote the *cost per cycle* of i. As desktop PCs are more common, likely to be in lower demand and have a lower running cost, we draw each desktop's $\hat{c}_i$ from the continuous uniform distribution $\mathcal{U}_c(0, 0.01)$, while each supercomputer's $\hat{c}_i$ is drawn from $\mathcal{U}_c(0.35, 0.5)$. Using this, we calculate the *cost per time unit* of provider i as $c_i = \hat{c}_i \cdot q_i$.Thus, the desktop PCs may be virtually free, but even the fastest need 20 hours on average to complete the consumer's task. The fastest supercomputers, in contrast, do this in an average 60 minutes, but charge up to \$50 per hour.

To simulate realistic migration costs between providers in a global network, we place all agents uniformly at random on a unit sphere. We then compute the migration cost from i to j as the shortest distance between them along the outside of the sphere, multiplied by a constant $\hat{m}$. We vary this constant in our experiments, such that migration costs *at most* \$0, \$5, \$10 or \$25.

5.2 Benchmark Strategies

As discussed earlier, we are interested in measuring the relative benefit of considering migration and also of using full information about the cost profiles of providers. Thus, we use a set of benchmark strategies that we classify along the following dimensions:

- **Information:** *Myopic* strategies assume that costs do not change, i.e., at each point in time, t, they plan optimally assuming $\forall i, x > t : c_i(x) = c_i(t)$. Once costs change, however, the plan is adapted. *Informed* strategies use full information about c_i.
- **Migration:** *Single* strategies plan optimally but use at most a single provider to complete the task. *Migrating* strategies use migration when this is beneficial.

Throughout this section, we consider all combinations of these, noting that our approach corresponds to *informed migrating*.

5.3 Continuous-Time Provisioning Results

We begin by looking at the continuous provisioning setting, where the task difficulty is distributed according to an expenential distribution with rate parameter $\lambda = \frac{1}{100}$. Since our approach does not deal with fluctuating cost profiles, we do not examine the difference between *informed* and *myopic* strategies here, concentrating instead on the difference beween *single* and *migrating* strategies. To cover a range of settings, Figure 1 shows the results for various numbers of providers, n, and migration costs, $\hat{m}$.

The top half here shows the average social welfare obtained for the different strategies. Several trends immediately emerge here. First, it is clear that using migration is generally of significant benefit, as it consistently obtains a higher utility in all but one setting. This improvement arises because the consumer is able to first attempt execution on the slower, cheap providers and then only near the deadline switch to the faster ones (a typical optimal migration schedule here has 2–3 providers). The *single* strategy, on the other hand, immediately procures a fast, expensive provider to complete the task.

Over all these cases, migration yields an average improvement of over 15%, but is as high as 22% in some cases. Generally, the relative improvement is greater when network costs are low and when there are more providers. Intuitively, this is because there are more opportunities for migration and the costs are lower.

To conclude the continuous case, the bottom half of the graph shows the expected utility of the intermediary agent, or centre. As outlined in Section 4, the centre here incurs a *deficit*. However, this is relatively small (8.5% of the welfare on average) and drops to less than 1% in some settings. In general, we note that the deficit decreases as the number of providers rises and as migration costs drop

[12] In general, it is impossible to have mechanisms which are both efficient and budget balanced [8]. Here we focus on efficiency, and consider budget balanced mechanisms in future work.

[13] To obtain statistical significance, we repeat all experiments 1000 times and when reporting performance differences, we ensure their significance by performing ANOVA and pairwise t-tests with $p < 0.05$. As the 95% confidence intervals of the results are small, we omit them from the graphs.

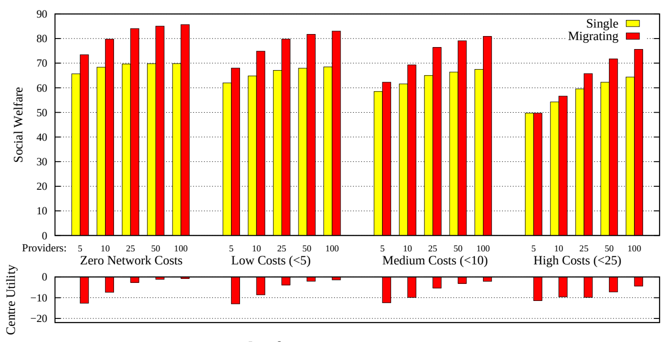

Figure 1. Results for continuous-time provisioning.

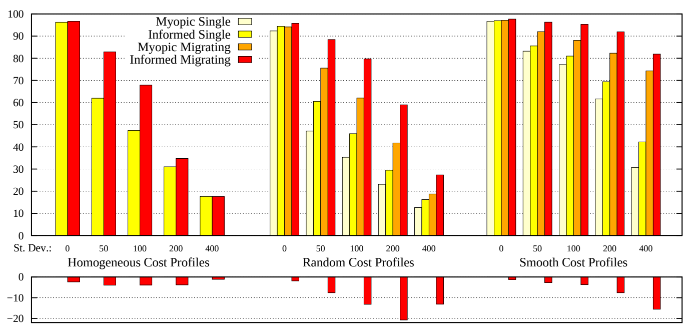

Figure 2. Results for discrete-time provisioning.

— this is because it is increasingly likely in these settings that a provider can be replaced by another similar one and so its marginal contribution is relatively low (i.e., there is more competition).

5.4 Discrete-Time Provisioning Results

Next, we consider the discrete-time case. As our approach for this deals with arbitrary distributions and cost profiles, we are here able to evaluate the effect of varying levels of uncertainty as well as cost fluctuations over time. Thus, we now assume the task difficulty is drawn from a normal distribution with a mean of 100 and standard deviations that we vary from 0 to 400 (truncated at 0). Furthermore, we consider *random* cost profiles for providers, where we perturb the cost per cycle $\hat{c}_i$ (described above) randomly once per hour by multiplying it by a random value drawn from $\mathcal{U}_c(0, 10)$. Second, we consider a *smooth* profile, where the cost of i is calculated as $c_i(x) = \hat{c}_i \cdot q_i \cdot 2.5 \cdot (\sin(\frac{\pi}{12} \cdot (\lfloor \frac{12}{\pi}\theta \rfloor + x)) + 1)$, where θ is the longitude of the provider on the unit sphere. Choosing this means that the cost varies smoothly between 0 and 5 times the normal cost during a full day, thus simulating peak and off-peak hours. Using θ creates an offset based on 24 equal time zones, reflecting the fact that peak times may be specific to the provider's location. As the trends discussed in Section 5.3 continue to hold, we now only consider 25 providers and fix the maximum migration cost at \$5.

The results of our experiments are shown in Figure 2, grouped by the cost profiles used (*homogeneous* refers to the unperturbed cost profiles, where no distinction between *informed* and *myopic* is necessary). Here, the benefit of task migration is more pronounced and evident over a large range of settings. Typically, the *informed migrating* strategy achieves a relative improvement of around 30–50% over the simple *myopic single* strategy, although for high uncertainty and fluctuating cost profiles, this rises to over 160%. The performance of the *informed single* is generally higher than its *myopic* counterpart, but usually significantly lower than either of the two *migrating* strategies. Thus, migration appears to be the key to performing well in these settings. In many settings, the *myopic migrating* strategy achieves 90% or more of the *informed migrating* strategy, but this drops to less than 70% for random cost profiles. Here, the random price fluctuations lead to wrong migration decisions that frequently leave the consumer unable to complete the task by the deadline. Thus, having full information is critical in some settings. Furthermore, we generally notice that migration initially becomes more beneficial when the uncertainty rises (i.e., as we increase the standard deviation of F). However, when this is too high, the consumer is increasingly forced to rely on a single fast provider again.

We also note that the centre's deficit can be relatively high in settings with random cost profiles. This is because the consumer often includes many providers in its schedule (up to 9 on average), to exploit the best offer at any point in time. Each provider here needs to be paid its marginal contribution, which results in a high overall payment. When a smooth cost profile is used, transfers are generally lower, because fewer providers are required and their marginal contribution is often lower too. Overall, the deficit is usually acceptable and, in all but two cases, outweighs the loss in welfare that would be incurred by switching to any other strategy.

6 CONCLUSIONS AND FUTURE WORK

The approaches outlined in this paper can be applied in a wide range of service-oriented settings. First, we envisage that users of cloud and grid systems can use our algorithms to execute their tasks more effectively, reducing overall costs and meeting time constraints despite uncertain requirements. Even in the absence of a mechanism, i.e., when the only information about costs consists of the current posted prices of resources, we showed that a myopic migration strategy still achieves a significant improvement over non-migrating strategies.

Second, our proposed mechanism can be used to implement efficient systems, where the best resources are selected to complete a given task. As this requires investment by an intermediary, we believe that this would most likely be offered by an organisation or government that has an interest in ensuring efficiency, e.g., to manage an open cloud or inter-organisational grid. Alternatively, an intermediary could offer the mechanism as a value-added service and reclaim costs elsewhere, e.g., through subscription fees or advertising.

In future work, we will consider the possibility of failures and settings where providers can invest a variable effort by dedicating only a proportion of its resources to the task.

ACKNOWLEDGEMENTS

This work was undertaken as part of the ALADDIN project, which is jointly funded by BAE Systems and EPSRC (EP/C548051/1), and as part of the EPSRC funded project on Market-Based Control (GR/T10664/01).

REFERENCES

[1] D. Bergemann and S. Morris, ‘Ex post implementation’, *Games and Economic Behavior*, **63**(2), 527–566, (2008).

[2] A. Blum and C. Burch, ‘On-line learning and the metrical task system problem’, *Machine Learning*, **39**(1), 35–58, (2000).

[3] P. Cheeseman, B. Kanefsky, and W. M. Taylor, ‘Where the really hard problems are’, in *Proc. IJCAI-91*, pp. 331–337, (1991).

[4] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*, McGraw-Hill, 2006.

[5] E. Gerding, S. Stein, K. Larson, A. Rogers, and N. R. Jennings, ‘Scalable mechanism design for the procurement of services with uncertain durations’, in *Proc. AAMAS2010*, pp. 649–656, (2010).

[6] A.V. Goldberg, J.D. Hartline, A.R. Karlin, M. Saks, and A. Wright, ‘Competitive auctions’, *Games Econ. Behav.*, **55**(2), 242–269, (2006).

[7] E. Huedo, R. S. Montero, and I. M. Llorente, ‘A framework for adaptive execution in grids’, *Softw. Pract. Exper.*, **34**, 631–651, (2004).

[8] A. Mas-Colell, M.D. Whinston, and J.R. Green, *Microeconomic Theory*, Oxford University Press, 1995.

[9] R. Porter, A. Ronen, Y. Shoham, and M. Tennenholtz, ‘Fault tolerant mechanism design’, *AIJ*, **172**(15), 1783–1799, (2008).

[10] M. P. Singh and M. N. Huhns, *Service-Oriented Computing : Semantics, Processes, Agents*, John Wiley & Sons, Inc., 2005.

[11] J-P. Watson, L. D. Whitley, and A. E. Howe, ‘Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search’, *JAIR*, **24**, 221–261, (2005).