# Flexible Procurement of Services with Uncertain Durations using Redundancy

**Sebastian Stein**[1], **Enrico Gerding**[1], **Alex Rogers**[1], **Kate Larson**[2], **Nicholas R. Jennings**[1]

[1]School of ECS, University of Southampton, Southampton, UK. {ss2,eg,acr,nrj}@ecs.soton.ac.uk
[2]Cheriton School of CS, University of Waterloo, Waterloo, ON, Canada. klarson@cs.uwaterloo.ca

## Abstract

Emerging service-oriented technologies allow software agents to automatically procure distributed services to complete complex tasks. However, in many application scenarios, service providers demand financial remuneration, execution times are uncertain and consumers have deadlines for their tasks. In this paper, we address these issues by developing a novel approach that dynamically procures multiple, redundant services over time, in order to ensure success by the deadline. Specifically, we first present an algorithm for finding optimal procurement solutions, as well as a heuristic algorithm that achieves over 99% of the optimal and is capable of handling thousands of providers. Using experiments, we show that these algorithms achieve an improvement of up to 130% over current strategies that procure only single services. Finally, we consider settings where service costs are not known to the consumer, and introduce several mechanisms that incentivise providers to reveal their costs truthfully and that still achieve up to 95% efficiency.

## 1 Introduction

Increasingly, participants in large distributed systems are able to discover and automatically procure the services of others. This allows service consumers to complete complex computational tasks on demand, but without the need to invest in and maintain expensive hardware. Already, such a service-oriented approach is gaining popularity in a large range of application areas, including Grids, peer-to-peer systems, cloud and utility computing [Singh and Huhns, 2005].

Despite its benefits, flexible service procurement poses new challenges that have not been addressed satisfactorily by current research. In particular, services offered by external providers are beyond the consumer's direct control and may therefore display uncertainty in their behaviour. Thus, the execution time of services can be highly uncertain, due to concurrent access by other consumers, hardware or network problems and the provider's scheduling policies. This is particularly problematic when services take a long time to complete, as is common for many computationally-intensive tasks, and when consumers need to obtain their results by a certain deadline. Furthermore, in large systems, many different providers may offer functionally equivalent services that are heterogeneous in their quality and costs. This requires consumers to make appropriate decisions about which services to procure, balancing the probability of success with the overall cost. It also necessitates the design of appropriate economic mechanisms that incentivise providers to truthfully reveal their private information, such as their costs and the estimated execution time, thus resulting in good procurement decisions and removing the need for strategic behaviour.

Related to this work is the literature on task allocation under execution uncertainty such as [Porter *et al.*, 2008]. Here, researchers have studied problems where providers have private information about both their costs for executing tasks, as well as the probability that they will successfully complete their tasks. However, this and similar works do not consider redundancy to increase the overall success probability of a task. Now, there is some work that employs redundancy, combining several unreliable services to achieve a higher probability of success. This includes deployed systems, such as Google's MapReduce [Dean and Ghemawat, 2008], but the techniques used for determining how many services to procure are typically ad hoc, and they also do not consider costs. A decision-theoretic approach for addressing the latter is described in [Stein *et al.*, 2008], but this work assumes that costs are known and focusses on heuristic techniques for complex workflow scenarios. A slightly different approach is taken by work on restarting Web queries, which examines when such queries should be timed out and re-issued (possibly to a different provider) to ensure timely completion [Chalasani *et al.*, 1998]. However, such work typically assumes that only one query is active at any time and the costs of multiple queries are not explicitly balanced with the resulting benefit.

To address these shortcomings, we present an abstract model of a procurement scenario for services with uncertain durations. We begin by outlining a generic approach for finding an optimal procurement strategy when service costs and duration distributions are known. This approach is the first to employ redundancy in a flexible and optimal manner as to balance the probability of completing within its deadline and the costs for doing so. More specifically, our approach allows a consumer to invoke multiple services in parallel for executing the same task, and it can dynamically procure further services during execution as the deadline draws closer. To

find the optimal strategy, we combine analytical expressions with computational search methods. As brute force is computationally intractable, we present a novel branch-and-bound algorithm that reduces the search, on average, by over 99.9%. We also discuss a heuristic algorithm capable of handling problems with thousands of heterogeneous service providers, and we show that its solutions are, on average, within 0.12% of the optimal. For a range of settings we then experimentally demonstrate that dynamic redundancy achieves an improvement of up to 130% over current approaches.

Next, we examine settings where service costs are private and known only by the providers. For this scenario, we propose a VCG-like mechanism that is incentive-compatible, i.e., that incentivises rational, self-interested participants to reveal their true costs. In this context, it is the first such mechanism that allows consumers to procure multiple, redundant services to increase its probability of success. Moreover, we show that this mechanism can achieve an average 95% efficiency when some prior information about service cost distributions is known to the consumer. To address settings where this is not available, we propose two further novel mechanisms, which have lower information requirements, but still achieve an average 86% efficiency.

In the remainder of this paper, we first present the procurement problem (Section 2) and discuss its optimal solution (Section 3). This is followed by our mechanisms (Section 4) and an empirical evaluation (Section 5). Section 6 concludes.

## 2 Problem Specification

We consider a single service consumer $A$, which needs to complete a task $T$. The consumer derives a utility $V \in \mathbb{R}^+$ if the task is successfully completed within a given deadline $D \in \mathbb{R}^+$, and 0 otherwise. Furthermore, there are $m$ service providers, given by the set $M = \{1, \ldots, m\}$, which can complete the task on the consumer's behalf. A consumer can invoke a provider $i \in M$ at any time in the interval $[0, D]$. We assume that, once invoked, the provider remains committed to the task until it is completed (possibly beyond the deadline), and incurs an (expected) cost $c_i$, where this cost may represent both the running costs of its computational resources and opportunity costs from not being able to use these resources for other tasks. To compensate for these costs, the consumer pays the provider a transfer $\tau_i \in \mathbb{R}^+$ on invocation of $i$, which is paid regardless of whether the task is completed by the deadline $D$. Although a provider will always successfully complete the task, the execution time is uncertain, and is given by a continuous cumulative distribution function $F_i(t)$. This denotes the probability that provider $i$ completes the task within time $t$, and we assume this includes any time needed for pre-/post-processing, queueing and data transfers. We also assume that the execution times of different service providers are independently drawn.

Although we only consider a single task in this paper, crucially we allow multiple providers to execute it concurrently and independently. In this case, the task is considered successful if at least one provider completes it by time $D$. We assume that all participants are expected utility maximisers.

Now, the key problem is to find an optimal procurement strategy that determines which providers should be invoked and when, such that the consumer's utility is maximised. We compactly represent such a strategy as a vector $\rho = ((s_1, t_1), \ldots, (s_n, t_n))$ with $n \leq m$, where each element represents the invocation time $t_i \in [0, D]$ of a provider $s_i \in M$. A provider $i$ is then only invoked at time $t_i$ (and only receives $\tau_i$) if no provider has so far completed the task. Without loss of generality, we assume that $t_i \leq t_{i+1}$, and $s_i \neq s_j$ if $i \neq j$. For example, assume there are four providers and $\rho = ((2, 0), (3, 0), (1, 2.5))$, i.e., providers 2 and 3 are invoked immediately. Then, if the task has not been completed by $t = 2.5$, provider 1 is also invoked, causing the three providers to run concurrently. Provider 4 is never invoked.

Given a strategy $\rho$, the consumer's expected utility is the difference between the expected value of completing the task and the expected total transfers to the invoked providers:

$$\begin{aligned} U_A(\rho) &= V \cdot \left(1 - \prod_{i=1}^{n} (1 - F_{s_i}(D - t_i))\right) \\ &\quad - \sum_{i=1}^{n} \left(\tau_{s_i} \cdot \prod_{j=1}^{i-1} (1 - F_{s_j}(t_i - t_j))\right) \end{aligned} \quad (1)$$

Furthermore, the expected utility of each provider $s_i$ is:

$$U_{s_i}(\rho) = (\tau_{s_i} - c_{s_i}) \cdot \prod_{j=1}^{i-1} (1 - F_{s_j}(t_i - t_j)) \quad (2)$$

if $s_i$ is included in $\rho$, and zero otherwise. Furthermore, although our main concern is maximising the consumer's utility, as a measure of how well the available services are utilised, we define the overall *efficiency*, also referred to as the social welfare, of a procurement strategy as:

$$\begin{aligned} U(\rho) &= U_A(\rho) + \sum_{i=1}^{n} U_{s_i}(\rho) \\ &= V \cdot \left(1 - \prod_{i=1}^{n} (1 - F_{s_i}(D - t_i))\right) \\ &\quad - \sum_{i=1}^{n} \left(c_{s_i} \cdot \prod_{j=1}^{i-1} (1 - F_{s_j}(t_i - t_j))\right) \end{aligned} \quad (3)$$

This measures the overall quality of a strategy for all participants and therefore ignores any transfers, as these only re-distribute utility between the agents.

## 3 Optimal Service Procurement

We now consider the problem of finding the optimal procurement strategy that maximises the consumer's expected utility, given that the consumer has full information about both the providers' costs $c_i$ and the duration distributions $F_i$. This corresponds to a service-oriented system where providers advertise their services at a fixed price, and thus $c_i$ denotes the advertised price. In this case, we set the transfers to the providers so that they equal these prices, i.e., $\tau_i = c_i$.

More formally, let $\rho^* = \text{argmax}_\rho U_A(\rho)$. Finding the optimum, $\rho^*$, is non-trivial since it involves selecting an appropriate subset of providers, ordering them and then determining invocation times. To solve this, we initially assume that the optimal subset of providers and their ordering is given. That is, we are given an ordered set of providers $\rho_s^* = (s_1, \ldots, s_n)$ where $s_i$ is invoked before $s_{i+1}$. To compute the optimal procurement schedule, we must determine $\rho_t^* = (t_1, \ldots, t_n)$, where $t_i$ is the invocation time of $s_i$. To this end, we compute the gradient of the expected welfare, $\nabla U(\rho_t^*)$, and find its root, i.e., $\nabla U(\rho_t^*) = \mathbf{0}$. This results in a system of $n$ simultaneous equations, with one equation for each $t_i$, with constraints, $\forall i : 0 \leq t_i \leq D$, and $\forall i, j : i \leq j \leftrightarrow t_i \leq t_j$. Solving these equations (and checking the appropriate second order conditions) depends on the family of duration distributions and can be done either analytically or numerically using standard optimisation software. In what follows, we focus on the exponential distribution as this is commonly used for modelling uncertain service durations [Trivedi, 2001].

### 3.1 Exponentially Distributed Durations

We now derive analytical expressions for the invocation times $\rho_t^*$, given $\rho_s^*$ and given that the duration distributions of providers $i \in M$ are given by $F_i(t) = 1 - e^{-\lambda_i t}$, where $\lambda_i > 0$ is a rate parameter. Re-writing Equation 1 with these distributions, and computing the gradient, allows us to compute the optimal invocation time $t_i$ of provider $s_i$ by solving:

$$0 = -V \cdot \lambda_{s_i} \prod_{j=1}^{n} e^{-\lambda_{s_j} D} \prod_{j=i+1}^{n} e^{\lambda_{s_j} t_j} + c_i \sum_{j=1}^{i-1} \lambda_{s_j} \prod_{k=1}^{i} e^{-\lambda_{s_k} t_i} - \lambda_{s_i} \sum_{j=i+1}^{m} \left( c_{s_j} \prod_{k=1}^{j-1} e^{-\lambda_{s_k} t_j} \prod_{k=i+1}^{j-1} e^{\lambda_{s_k} t_k} \right) \quad (4)$$

Here, we note that $t_i$ is independent of any $t_j, j < i$, i.e., the invocation time of a provider does not depend on the invocation time of those already running. This is a result of the exponential function being memoryless, i.e., the probability of completing the task within the next time interval $\Delta t$ is independent of when it was invoked. Hence, we can calculate each $t_i$ by backward induction, starting with the last provider, $n$. The invocation time of this can be obtained directly by taking the derivative with respect to $t_n$ (as in Equation 4):

$$t_n = \frac{\ln\left(V \cdot \lambda_{s_n}\right) - \ln\left(c_{s_n} \cdot \sum_{j=1}^{n-1} \lambda_{s_j}\right) - D \cdot \sum_{j=1}^{n} \lambda_{s_j}}{\sum_{j=1}^{n} \lambda_{s_j}} \quad (5)$$

Furthermore, we can obtain a simpler closed-form solution for the remaining invocation times by combining and manipulating the partial derivatives for $t_i$ and $t_{i+1}$, resulting in:

$$\frac{c_{s_i}}{\lambda_{s_i}} \sum_{j=1}^{i-1} \lambda_{s_j} \prod_{k=1}^{i-1} e^{-\lambda_{s_k}(t_i - t_k)} - \sum_{j=i+1}^{m} \left( c_{s_j} \prod_{k=1}^{j-1} e^{-\lambda_{s_k}(t_j - t_k)} \right)$$
$$= \frac{c_{s_{i+1}}}{\lambda_{s_{i+1}}} \sum_{j=1}^{i} \lambda_{s_j} \prod_{k=1}^{i} e^{-\lambda_{s_k}(t_{i+1} - t_k)} - \sum_{j=i+2}^{m} \left( c_{s_j} \prod_{k=1}^{j-1} e^{-\lambda_{s_k}(t_j - t_k)} \right)$$

Then, using algebraic manipulations, we isolate $t_i$, and derive an expression that is based solely on $t_{i+1}$:

$$t_i = t_{i+1} - \frac{1}{\sum_{j=1}^{i} \lambda_{s_j}} \ln\left( \frac{c_{s_{i+1}} \lambda_{s_i} \sum_{j=1}^{i+1} \lambda_{s_j}}{c_{s_i} \lambda_{s_{i+1}} \sum_{j=1}^{i-1} \lambda_{s_j}} \right) \quad (6)$$

Note that Equation 6 is not well defined for $t_1$, and the optimal here is to set $t_1 = 0$. This is because the cost will be incurred in any case and any delays would only reduce its probability of success by the deadline. Furthermore, we note that the equations can yield negative values for some $t_i$, indicating that the optimal values lie outside the constraints of the problem (i.e., before the task can be started). In this case, as $t_i$ does not influence the procurement times of later providers, the optimal choice is to set $t_i = 0$, i.e., the provider is invoked at the earliest possible time. Furthermore, the equations can sometimes yield inconsistent values, i.e., $t_i > D$ or $t_i > t_{i+1}$ for some $i$, but this only occurs when the ordering and/or the set of providers was non-optimal in the first place. Finally, we also note that the partial second derivatives are always negative, and since each variable is found uniquely one at a time using backward induction, the final result is optimal.

So far, Equations 5 and 6 allow us to efficiently calculate the optimal procurement times for a given, optimal ordered sequence of service providers $\rho_s^*$. However, it is not obvious how to find this order. Related work on economic search, such as [Weitzman, 1979], does not apply to this case, due to the overlap of concurrently invoked providers. Furthermore, our problem includes a fixed time constraint, by which the task has to be completed. Other greedy approaches that order services by increasing costs, decreasing rate parameters, the ratio of these, or approaches that first select providers who individually yield a higher expected utility, also do not always find optimal solutions. This is because it is often best to select cheaper, slower providers first and only invoke the more expensive and faster ones later, to ensure that the task is completed successfully. However, when the deadline of the task is particularly short, the consumer may be forced to immediately invoke the faster, expensive providers.

As a simple example of this, we consider a set of two providers, $M = \{1, 2\}$. The first is cheap and slow with $c_1 = 0.2$ and $\lambda_1 = 0.1$, while the second is expensive and fast with $c_2 = 5$ and $\lambda_2 = 10$. If we then assume that a consumer has a task $T$ with deadline $D = 1.5$ and utility $V = 100$, the optimal procurement strategy is $\rho^* = ((1, 0), (2, 0.75))$. However, if we decrease the deadline slightly to $D = 1$, the optimal strategy becomes $\rho^* = ((2, 0), (1, 0.84))$, thereby reversing the order of invoked providers.

This observation suggests that a simple greedy search for the optimal strategy is insufficient. Hence, in the following sections, we present an optimal branch-and-bound algorithm. As this becomes slow when there are dozens of providers, we also discuss a fast heuristic algorithm.

### 3.2 The Branch-And-Bound Algorithm

Finding an optimal subset and ordering of providers $\rho_s^*$ using a brute-force search is clearly infeasible[1] when the number of

[1] More generally, we believe that this problem is NP-hard due to its similarity to the well-known Stochastic Integer Packing and the

**Algorithm 1** Branch-And-Bound Algorithm.

```
1:  ρ*_s ← ()                                          ▷ Best ordering found so far
2:  u_lower ← 0                                        ▷ Best current lower bound
3:  Q ← {ρ*_s}                                         ▷ Unexpanded orderings
4:  while Q ≠ ∅ do                                     ▷ More unexpanded?
5:      ρ_s ← argmax_{ρ_s ∈ Q} LOWER(ρ_s)              ▷ Pick best
6:      Q ← Q \ {ρ_s}                                  ▷ Remove ρ_s from Q
7:      P'_s ← EXPAND(ρ_s)                             ▷ Expand ρ_s
8:      P'_s ← FILTERDOMINATED(P'_s)                   ▷ Remove dominated
9:      for all ρ'_s ∈ P'_s do
10:         ŭ ← LOWER(ρ'_s)                            ▷ Find lower bound
11:         û ← UPPER(ρ'_s)                            ▷ Find upper bound
12:         if û > u_lower then                        ▷ Sufficient upper bound?
13:             if ŭ > u_lower then                    ▷ Better lower bound?
14:                 ρ*_s ← ρ'_s                        ▷ Keep as current best
15:                 u_lower ← ŭ
16:             Q ← Q ∪ {ρ'_s}                         ▷ Keep for future expansion
17:     Q ← {x ∈ Q | UPPER(x) > u_lower}               ▷ Filter orderings
18: return FINDTIMES(ρ*_s)                             ▷ Return best strategy
```

providers rises beyond a handful, as the number of possible orderings for $m$ providers is $\sum_{i=0}^{m} \binom{m}{i} \cdot i! = \sum_{i=0}^{m} \frac{m!}{(m-i)!}$.

Now, it is possible to significantly reduce the number of provider orderings that need to be searched by noting that we can use information about some examined orderings to exclude others. For example, assume we have three providers, and we have just considered the ordering $\rho_s = (2, 1)$. This already promises a high utility, and, in fact, we note it is higher than what could possibly be achieved by invoking provider 3 first (e.g., if $V - c_3$ is low). Hence, we can immediately discard all five orderings starting with 3.

This intuition is generalised in our branch-and-bound technique given by Algorithm 1. In more detail, we begin with an empty ordering $\rho^*_s = ()$ (line 1), and then repeatedly consider any new ordering that can be created by appending a single provider *to the end* of an existing ordering. This is implemented by keeping a set of orderings, $Q$ in line 3, that have not yet been expanded in this manner. During each iteration of the main loop of the algorithm (lines 4–17), we then remove one[2] ordering $\rho_s$ from $Q$ (lines 5 and 6) and expand it. Here, EXPAND in line 7 takes an ordering $\rho_s$ and returns the set of all orderings that can be obtained by appending a single remaining service provider from $M$ to $\rho_s$. From this set of new orderings, we then remove any that include providers that are dominated by others not currently in the ordering (line 8).[3]

For each new ordering $\rho'_s$, we now find both a lower bound and an upper bound for the expected utility that is achievable by any procurement strategy beginning with the providers in $\rho'_s$ (lines 10 and 11). Finding these allows us to exclude any orderings starting with $\rho'_s$ if the associated upper bound is less than the best lower bound found so far. This pruning and updating of the lower bound is performed in lines 12–16.

We now describe LOWER$(\rho'_s)$ and UPPER$(\rho'_s)$. To find the lower bound, we simply restrict ourselves to the providers in $\rho'_s$, and find the optimal times $\rho'_t$ for this ordering and return the associated utility, i.e., $U_A(\text{FINDTIMES}(\rho'_s))$, where FINDTIMES returns the optimal procurement strategy using the Equations from Section 3.1. Calculating an upper bound is less obvious, because we may be able to derive significantly higher utility by invoking further services. To this end, we let $M'$ be the remaining service providers that are not in $\rho'_s$. If $M' = \emptyset$, then the upper bound is equal to the lower bound discussed above. Otherwise, we create a virtual service provider $s_\rho$ with $c_{s_\rho} = \min_{i \in M'} c_i$ and $\lambda_{s_\rho} = \sum_{i \in M'} \lambda_i$. This is based on the rationale that if any providers from $M'$ are invoked in any order, their cost is bound to be at least $c_{s_\rho}$ and their combined probability of success within any given time interval after invocation will never be higher than when immediately invoking all in parallel. With this reasoning, we obtain a new ordering $\rho''_s$ by appending $s_\rho$ to $\rho'_s$ and then calculate the upper bound as $U(\text{FINDTIMES}(\rho''_s))$. If that is less than the lower bound, this indicates that it is not possible to achieve a higher utility by invoking further providers, and we can set the upper bound equal to the lower bound.

At the end of each iteration, only unexpanded orderings with an upper bound that is higher than the currently highest lower bound are retained (line 17). This limits the size of $Q$ (which we implemented using a priority queue), and also ensures that it is empty when all necessary orderings have been searched. When this happens, the best ordering and associated optimal times are returned (line 18). This final procurement strategy is optimal, because the algorithm searches all orderings, except for those that are known to have a lower expected utility than those already considered. Hence, the optimal ordering will never be discarded from the search.

However, while significantly reducing the search space in most realistic settings, this algorithm still searches for the optimal solution and may sometimes consider a large proportion of the entire search space. This may be the case, for example, when there are large numbers of highly similar providers and when the value of the task is very large in relation to the service costs. To address such scenarios, we introduce a fast heuristic approach in the following section.

### 3.3 The Heuristic Algorithm

Although we argued in Section 3.1 that a greedy approach does not generally result in an optimal strategy, it can still achieve good results in practice and is more scalable than exhaustive approaches. Hence, we present such an algorithm that starts with an empty ordering and then greedily adds, removes or switches providers until a local optimum is reached. Intuitively, this algorithm benefits from selecting providers that offer a good trade-off between performance and cost.

In more detail, given a current ordering $\rho_s$ and a set of providers $M'$ that are currently not in $\rho_s$, the greedy approach picks one of the following three actions, in order to maximise the expected utility of its next ordering: (1) it selects a provider $x \in M'$ and adds it to $\rho_s$ at position $i \in \{1, 2, \ldots, n+1\}$ (shifting other providers as necessary), (2) it selects a provider $s_i$ in $\rho_s$ and removes it, or (3) it se-

Stochastic Knapsack problems, but we leave a formal proof of this conjecture for future work.

[2]We remove the ordering that promises the highest lower bound on the expected utility. This allows us to quickly increase the best lower bound, thereby pruning the search space more effectively.

[3]A provider $i$ dominates $j$ if and only if $(c_i \leq c_j \wedge \lambda_i > \lambda_j) \vee (c_i < c_j \wedge \lambda_i \geq \lambda_j)$. Clearly, it is suboptimal to invoke $j$ before $i$.

lects two providers $s_i$ and $s_j$ in $\rho_s$ and swaps their positions. This continues until the algorithm cannot find another better ordering. In this case, the current best is returned.

# 4 Mechanisms for Eliciting Costs

Whereas so far we have assumed that the consumer, $A$, has complete information about both the costs and the duration distributions of the providers, here we consider a setting where the cost information is private and unknown to the consumer. Instead, the consumer has to provide *incentives* so as to induce the providers to reveal this information truthfully. Note that we still assume that the providers' duration distributions are known by the consumer, as this information may be obtained from past and shared experiences, e.g using a trust or reputation system, or simply given by the provider.[4]

## 4.1 $(k+1)^{th}$ Price Mechanism

Typically, when mechanism design is applied to task allocation problems, the well-known Vickrey-Clarke-Groves (VCG) mechanism is used. Providers are asked to reveal their private information (called their *type*) and in exchange are paid a transfer equal to their marginal contribution to the system. This payment structure provides the correct incentives so that each provider willingly reveals their type truthfully [Mas-Colell *et al.*, 1995]. Unfortunately, however, the VCG mechanism is not applicable in our setting, since it is only suited for situations where the types (i.e. the costs) of the providers are independent. In our domain, this property does not hold, since the *expected* cost incurred by a provider depends significantly on which other providers are selected in the procurement strategy. Thus, our problem falls into the class of *interdependent valuations*, and it is well known from the literature that providers no longer have an incentive to reveal their private information truthfully if the VCG mechanism is used [Mas-Colell *et al.*, 1995]. Moreover, in such a setting there exists no general mechanism which is both truthful and efficient [Jehiel and Moldovanu, 2001].

Our first mechanism is the $(k+1)^{th}$ mechanism which works as follows. First, the consumer, $A$, announces $k$, $1 \leq k < m$. Then, each of the $m$ providers reports a cost, $\hat{c}_i$, which may differ from their true cost $c_i$. We assume that providers are ordered so that $\hat{c}_i \leq \hat{c}_{i+1}$, and we define $K = \{i \mid i \in M \text{ and } \hat{c}_i < \hat{c}_{k+1}\}$. That is, $K$ is the set of $k$ providers with the lowest reported costs. These form the *candidate providers* for the procurement strategy. After finding the set $K$, the payment value $\tau_i$ of each candidate provider is set to $\tau_i = \hat{c}_{k+1}$. Now, the procurement strategy used by $A$ is calculated by finding $\rho' = \text{argmax}_{\rho|s_i \in K} U_A(\rho)$ where only providers in $K$ are (potentially) selected to be part of the strategy. It is important to notice that the payment $\tau_i$ for $i \in K$ is *conditional*. That is, a payment or transfer to agent $i \in K$ only occurs if the candidate provider is both selected as part of the procurement strategy $\rho'$, and is subsequently invoked. Otherwise, it receives no payment (and incurs no cost).

[4]To verify these distributions in the latter case, a payment scheme based on *scoring rules* could be used in conjunction with our mechanism, see [Zohar and Rosenschein, 2008], but we leave a more detailed investigation of this issue for future work.

**Theorem 1.** *Let $M$ be the set of service providers, $|M| = m$. For any $k$ such that $1 \leq k < m$, the $(k+1)^{th}$ mechanism is incentive compatible in dominant strategies (i.e., truthtelling is optimal, irrespective of what other agents do) and (ex-post) individually rational (i.e., the providers always receive zero or positive utility).*

*Proof.* Individual rationality holds when $c_i = \hat{c}_i$ since $\tau_i = \hat{c}_{k+1} \geq c_i$ for $i \in K$, and thus $\tau_i - c_i \geq 0$, and furthermore $\tau_i = 0$ for $i \notin K$. As a result, the provider's expected utility (Equation 2) when $c_i = \hat{c}_i$ is always positive, irrespective of the reports of others. To show incentive compatibility, we first note that the selection of providers in K as part of the procurement strategy and their invocation time (if selected) are indendependent of their reported costs. Therefore, if $i \in K$, the probability of being invoked and the payment, and thus the expected utility, are independent of a provider's report. A provider's report thus only affects whether or not it will be selected as one of the candidate providers in $K$. Now, by overreporting, a provider can only reduce the likelihood of being selected and thus potentially foregoes a positive expected utility. By underreporting, a provider can increase this likelihood, but this will only result in a change (compared to reporting truthfully) when $\hat{c}_i < \hat{c}_{k+1}$ and $c_i > \hat{c}_{k+1}$. In this case, however, if the provider is invoked its expected utility is strictly negative since $\tau_i - c_i < 0$. □

While this mechanism has desirable properties, it also suffers from some key limitations. First, it selects providers based solely on their cost information, ignoring the duration distributions, leading to the possibility that expensive providers with fast completion times are excluded. Second, the parameter $k$ must be announced before providers reveal their costs. To set $k$ optimally requires *a priori* information about the distribution of the costs, and expensive calculations and/or simulations (as done in Section 5). To address this last problem, we now introduce two variations of our mechanism.

## 4.2 Grouping Mechanisms

We introduce two new mechanisms: *Pairing* and *Halving*. These mechanisms differ from the $(k+1)^{th}$ mechanism in both the provider-selection process and the calculation of the (conditional) payment, $\tau$.

In the *Pairing mechanism*, every provider $i \in M$ reports a cost, $\hat{c}_i$. Then, all providers are *randomly* paired with another provider (if $|M| = m$ is odd, then a single triplet is formed). For each pair, the provider with the lower announced cost is placed in the set $K$ and the conditional payment is set equal to the announced cost of the other provider (in the case of a triplet, the provider with the lowest announced cost is placed in $K$ and the conditional payment is equal to the second lowest announced cost in the triplet). All providers not in $K$ are not selected and therefore receive no payment. This results in $|K| = \lfloor m/2 \rfloor$ and $\rho' = \text{argmax}_{\rho|s_i \in K} U_A(\rho)$ as before.

In the *Halving mechanism* all providers in $M$ announce costs as is done in the Pairing mechanism. Then, $\lfloor m/2 \rfloor$ providers are randomly selected and placed into a set $G$. All other providers are randomly paired and are then treated identically to those in the Pairing mechanism. For members of $G$, the provider with the lowest announced cost is placed in $K$

and its conditional payment is equal to the cost of the second lowest announced cost from $G$, while all other providers are discarded. The procurement strategy is computed as before.

**Theorem 2.** *The Pairing and Halving mechanisms are incentive compatible and (ex post) individually rational.*

*Proof.* Since the pairs and $G$ are formed independently of the providers' reports, the proof follows from Theorem 1. □

We note that there are many possible variations of these mechanisms, but all would share some key features. First, the size of $K$ is solely determined by the *number of providers*, and thus does not rely on the consumer choosing an appropriate value. Second, the mechanisms require no *a priori* information about the cost distributions. Finally, they implement *discriminatory pricing* (i.e., different providers receive different payments), information which is then used to form the optimal procurement strategy (given $K$). On the other hand, the payments are always based on the higher costs (except in the Halving mechanism), and it is therefore not clear whether these variations offer any real benefits in practice. To this end, we experimentally evaluate them in the next section.

## 5 Evaluation

We now evaluate our proposed approaches in a variety of simulated environments, to determine if they provide benefits over existing techniques, and to investigate the cost of incentivising providers to reveal their private information. Throughout this section, we randomly generate each provider $i$ by drawing its cost $c_i$ and duration rate $\lambda_i$ independently and uniformly at random from $[0, 1]$. To consider a range of settings, tasks have either a low ($V_{\text{low}} = 2$) or a high value ($V_{\text{high}} = 8$) and their deadline is either normal ($D_{\text{normal}} = 2$) or urgent ($D_{\text{urgent}} = 0.5$). Furthermore, we repeat experiments 1000 times and use ANOVA and t-tests to ensure statistical significance at the $p < 0.05$ level. As the associated confidence intervals are small, we omit these here for clarity.

### 5.1 Optimal Service Procurement

First, we consider environments where providers charge their true costs, i.e., $\tau_k = c_k$, and compare the average utility obtained by the optimal procurement strategy[5] described in Section 3 (*Optimal*) to a strategy that always selects the single provider that individually maximises the consumer's expected utility (*Single*). This latter strategy represents current task allocation approaches that do not include redundancy.

The results are shown in Figure 1. Here, we vary the number of providers in the system and plot the average expected consumer's utility, which is equal to the overall efficiency in this setting, as a proportion of $V$. Observing these trends, it is obvious that using redundancy can significantly improve the consumer's utility and does so in almost all settings considered. In fact, when averaging over all cases considered, the Optimal approach achieves more than a 35% improvement over the Single approach. In particular, when the deadline

[5]This is found using our branch-and-bound algorithm when there are up to ten providers. We then use the heuristic algorithm to obtain a lower bound for the optimal when there are more providers. However, as we show later, the heuristic obtains near-optimal results.

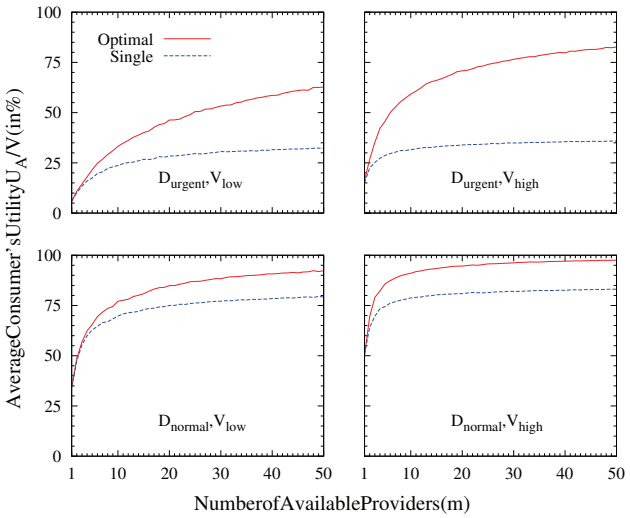


Figure 1: Performance in full information setting.

is small ($D_{\text{urgent}}$) and the task value high ($V_{\text{high}}$), the Optimal strategy is able to employ high redundancy to ensure the task is completed within the deadline, while the higher task utility justifies the additional investment. For example, when there are 50 providers, the Single approach achieves 35.87% of $V$, while the Optimal achieves 82.65% — a 130% improvement.

Next, we note that when solving the above problems, our branch-and-bound approach significantly reduces the computational time required when compared to a brute-force algorithm. For example, when there are 12 providers and we consider $V_{\text{high}}$ and $D_{\text{urgent}}$, a brute force approach searches over 1.3 billion provider orderings, which takes an average 3.3 hours (using a Java implementation on a Windows-based Intel 2.2 GHz laptop with 4 GB RAM). By contrast, the branch-and-bound algorithm searches an average 42000 orderings (0.003% of the total search space), finding the optimal in half a second. While the latter still finds solutions for 18–23 providers in minutes (where the brute-force would take over $2 \cdot 10^{10}$ years — longer than the age of the universe), our heuristic approach is better suited for larger settings with hundreds or thousands of providers. To investigate how its performance compares to the optimal, we have applied both to all settings described above with ten or less providers. Here, the heuristic achieved 99.88% of the optimal on average.

### 5.2 Incentive Compatible Mechanisms

Now we consider the mechanisms described in Section 4 and investigate how close the resulting procurement decisions are to the optimal (both in terms of the consumer's utility and the overall efficiency). To this end, Figure 2 compares the performance of a range of $(k+1)^{th}$ price mechanisms with varying $k$, and our Halving and Pairing mechanisms to the optimal (for brevity, we consider only two representative scenarios here, but similar results are obtained in other settings).

It is immediately obvious here that all mechanisms suffer from a loss in utility for the consumer — a cost that results from incentivising providers to report truthfully. More

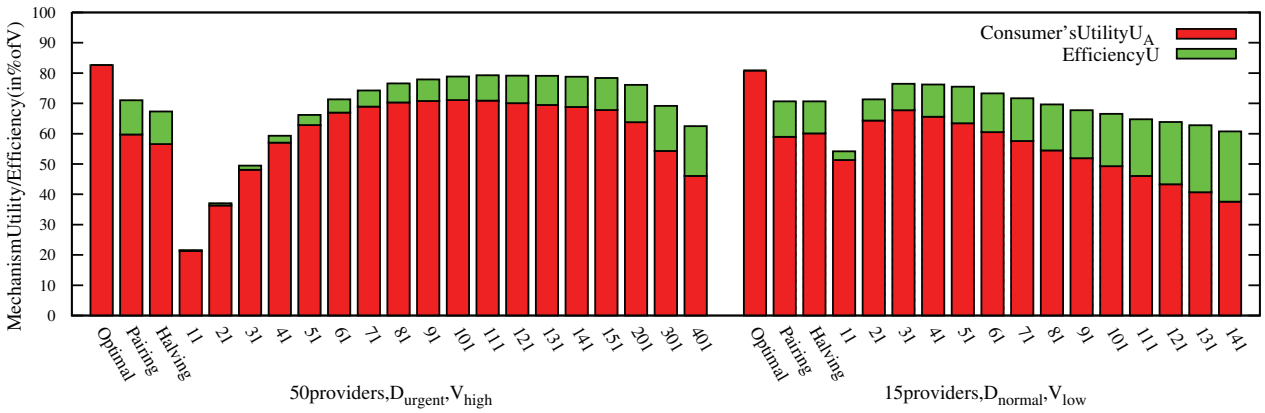


Figure 2: Performance of incentive-compatible mechanisms.

specifically, the best $(k+1)^{th}$ price mechanism in each case achieves an average $85\%$ of the optimal, while Pairing and Halving both achieve over $70\%$. We also note that the performance of the $(k+1)^{th}$ depends heavily on the choice of $k$ and can be as low as $25\%$ of the optimal if the wrong parameter is chosen. Furthermore, the best parameter depends on the scenario. For example, for the task with $V_{\text{low}}$, $k=3$ is the best choice, achieving over $83\%$ of the optimal. However, for $V_{\text{high}}$, it is one of the worst, achieving only $58\%$. Hence, these results indicate that a consumer can achieve a good utility by using appropriate $k$ parameters. However, when insufficient information is available to set $k$, it can obtain good results by using a Pairing or Halving mechanism.

Next, we consider the overall efficiency, or social welfare. This ignores utility transfers between the consumer and the providers, and therefore gives a better indication of how effectively the available services are used to complete the task at hand. Here, we note that the mechanisms consistently achieve a good overall efficiency. The best $(k+1)^{th}$ mechanism now reaches, on average, over $95\%$ of the optimal efficiency while the Pairing and Halving mechanisms achieve $86\%$ and $84\%$, respectively.

## 6 Conclusions

In this paper, we considered a setting where multiple providers can be redundantly procured to perform a single task which has to be completed within a given deadline. The providers have uncertain execution times, which are given by probability distributions, and incur different costs for executing the task. We first considered the setting with known costs and introduced an algorithm for calculating the optimal procurement strategy, as well as a near-optimal heuristic algorithm for settings with a large number of providers. We then introduced several incentive-compatible mechanisms for eliciting the costs when these are unknown, and we evaluated our approaches empirically. The results showed that redundancy significantly outperforms the standard approach where only a single provider is selected for each task, and it continues to perform well in the incomplete information setting.

In future work, we are interested in investigating a setting where the execution duration distributions are also privately known, and need to be elicited by a consumer. Furthermore, we intend to apply this approach to larger settings with multiple, interdependent tasks.

## Acknowledgments

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) 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

[Chalasani *et al.*, 1998] Prasad Chalasani, Somesh Jha, Onn Shehory, and Katia Sycara. Query restart strategies for web agents. In *Proc. AGENTS '98*, pages 124–131, 1998.

[Dean and Ghemawat, 2008] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[Jehiel and Moldovanu, 2001] P. Jehiel and B. Moldovanu. Efficient design with interdependent valuations. *Econometrica*, 69(5):1237–1259, 2001.

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

[Porter *et al.*, 2008] R. Porter, A. Ronen, Y. Shoham, and M. Tennenholtz. Fault tolerant mechanism design. *Artificial Intelligence*, 172(15):1783–1799, 2008.

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

[Stein *et al.*, 2008] S. Stein, N. R. Jennings, and T. R. Payne. Flexible service provisioning with advance agreements. In *Proc. AAMAS08*, pages 249–256, 2008.

[Trivedi, 2001] K.S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley & Sons, Inc., USA, 2nd edition, 2001.

[Weitzman, 1979] M. L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.

[Zohar and Rosenschein, 2008] A. Zohar and J. S. Rosenschein. Mechanisms for information elicitation. *Artificial Intelligence*, 172(16–17):1917–1939, 2008.