Robust Execution of Service Workflows using Redundancy and Advance Reservations (Online Appendix)

Sebastian Stein, Terry R. Payne and Nicholas R. Jennings

1 Introduction

This online appendix accompanies the paper "Robust Execution of Service Workflows using Redundancy and Advance Reservations" [1]. It illustrates how the algorithm proposed in that paper is applied to a real workflow from the bioinformatics domain. In doing so, it highlights the applicability of the algorithm to a real application, and it complements the high-level description in the original paper by giving a concrete example.

The appendix is organised as follows. In Section 2, we briefly describe the bioinformatics workflow that serves as the example throughout the appendix, and in Section 3, we detail the performance characteristics of the available services. In Section 4, we provide some examples of the high-level strategies that are available to the consumer agent. Then, in Section 5, we detail the process of reserving offers for the workflow when the consumer associates a low utility and long deadline with it, and in Section 6, we discuss how this changes when the workflow is more urgent. We conclude in Section 7.

2 Bioinformatics Workflow

Throughout this appendix, we consider a typical workflow from the bioinformatics domain[1] — an area that relies heavily on computationally intensive services and that has increasingly seen the establishment of large distributed Grid systems for sharing resources, as exemplified by the GriPhyN [3], myGrid [4] and Comb*e*Chem [5] projects. Bioinformatics, and, more generally, data processing applications in e-science, are also prime examples of a domain where execution uncertainty and failures are rife. More specifically, computational tasks in this domain often take a long time to complete, sometimes running for hours or days. This directly increases the probability that hardware failures or software problems disrupt the workflow, especially when this contains many such tasks. As a concrete example of this uncertainty, [6] describes a real scientific workflow, which routinely encounters multiple task failures during a single run.

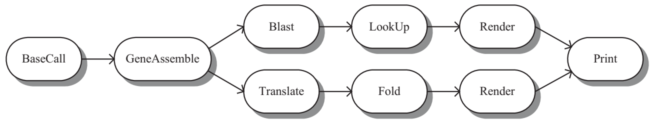

Fig. 1. Example bioinformatics workflow, based on workflows described by [7], [8] and [9].

Now, although providers may be able to mitigate the uncertainty by using replication, as we argued in Section 2, this requires a considerable effort on the provider's side, which may simply not be economical. As an example, it may often be more profitable for a provider to also sell processing time on a newly acquired mainframe (and thereby potentially doubling its revenue), rather than to use it for replication, in order to increase reliability by a few percent. Clearly, some providers *may* choose to increase reliability in this way and charge a premium for this, but our approach will then make the appropriate choice for the consumer, and, critically, it also addresses systems where all providers are unreliable.

Against this background, for our example we assume that a scientist has just sequenced a previously unknown gene of a bacterium, and is now interested in visualising the shape of the associated protein. For this, she has to carry out a number of tasks, which are shown in Figure 1.

Her initial data comprises a large set of overlapping DNA fragments in the form of chromatograms, as is common in shotgun DNA sequencing [10]. These show characteristic light traces at different wavelengths, corresponding to the four bases found in a DNA sequence. As these traces typically contain some noise and errors,

• *Stein and Jennings are with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK. E-mail: {ss2,nrj}@ecs.soton.ac.uk.*
• *Payne is with the School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Liverpool, L69 3BX, UK. E-mail: T.R.Payne@liverpool.ac.uk.*

1. This workflow was first described in [2].

the scientist first needs to run a base-calling service (*BaseCall*). This translates the chromatograms to the corresponding base sequences, attaching a quality value to each base in the process that denotes how accurate the assignment of the base is. The resulting base sequences are then assembled to a single continuous DNA sequence by identifying and merging overlapping fragments, using the quality values to find and repair errors. This task is performed by a sequence-assembling service, which also identifies and isolates the coding region of the gene (*GeneAssemble*).

When the coding region of the gene has been assembled, it is then translated to the corresponding amino acid sequence using a simple translation service (*Translate*). As the primary structure of the protein, this forms the input to the computationally-intensive folding service (*Fold*), which predicts the 3-dimensional shape of the protein based on a search for the conformation with the lowest free energy. The output of this — a file containing the tertiary structural data — is then rendered in high resolution using an appropriate graphics service (*Render*). In parallel with the folding simulation, the scientist is also interested in comparing the new gene to previously discovered sequences. To this end, she searches through public collections of known proteins to find the closest match using a specialised service (*Blast*), and then accesses commercial database services to retrieve structural information about the protein (*LookUp*). This is rendered again, and both images are printed as part of a report on a local printer (*Print*).

In the following, we describe how offers for these services are generated at run-time.

3 Service Market

For the purpose of illustrating our algorithm, we detail the operation of a dynamic market for the services described in the previous section in a similar manner as done in Section 4 of [1]. In more detail, we keep a list of currently available offers associated with each time step (representing a minute in real time), from the current step t to $t+60$. Hence, the consumer may reserve services up to one hour in advance. During the simulation, at the beginning of each time step, we first generate new offers that become available in the market by drawing the number of new offers and their parameters from random distributions. These distributions are detailed in Table 1 ($\mathcal{U}_c(a, b)$ is a continuous uniform distribution over the interval $[a, b]$) and depend on the number of time steps the offer is generated in advance. This time dependency allows us to include performance differences in offers when they are reserved with varying advance notice periods. It is expressed here by including two rows of distributions for each service type — the first indicates the performance of offers when reserved at short notice (as given by the *advance time* column), and the second gives the performance when offers are reserved with a long advance notice period.

To elaborate this, for each service type and for each possible time step from t to $t + 60$, we first generate the number of new offers by drawing a sample from a Poisson distribution with a mean given by the respective *birth* rate[2]. Then, for each such generated offer, we draw its failure probability, reservation cost, execution cost and service duration from the relevant distributions (depending on how far the offer is generated in advance). When the offer time lies between the two extremes corresponding to the two rows for each service type, we interpolate linearly between the distribution parameters. For example, when generating an offer for the *BaseCall* service type for time step $t + 2$, we draw its failure probability from $\mathcal{U}_c(0.18, 0.44)$. If the service type is marked as repaying in the table, we set the failure penalty to the combined execution and reservation costs (this means that providers for these services always refund a consumer in case of failures). Otherwise, it is set to 0. Finally, at the end of each time step, we remove offers in a similar way as above by drawing a random sample from a Poisson distribution with its mean given by the *death* rate. This models the demand for such services and we randomly remove the generated number of offers from that time step (or all offers if the number exceeds the current supply).

The service parameters in Table 1 were chosen to represent a range of heterogeneous services with some, such as *Fold*, being expensive and time-consuming, while others, such as BaseCall, are fast and cheap. Also, the services display significant performance differences depending on the time of reserving them. For example, some services are generally more reliable and faster when reserved in advance (*BaseCall* and *GeneAssemble*), some offer a far better service overall but are also significantly more expensive and need a long advance notice period (*Fold*), and a few display no difference in quality over time (*Look Up* and *Print*).

Given this service market, we describe a number of example high-level task strategies in the following section.

4 High-Level Task Strategies

As described in Section 3.2 of the main paper, we assume that the service consumer has already obtained a library of atomic high-level strategies by observing the market over some time. To illustrate this, Figure 2 shows a number of example strategies and their statistics for the *Fold* service type. In the centre, we use the notation $\omega = \langle 1, 20, 10, b\rangle$ to represents a strategy ω that reserves a single offer ($n(\omega) = 1$) 20 time steps before it is required ($t_a(\omega) = 20$), with the consumer considering 10 consecutive time steps ($t_w(\omega) = 10$) and selecting the offer that best balances all performance characteristics

2. As in the main paper [1], we chose the Poisson distribution here because it is a common distribution for modelling random arrival events [11].

Service Type	Advance Time (minutes)	Failure Probability	Reservation Cost ($)	Execution Cost ($)	Time (minutes)	Birth/ Death	Repay?
BaseCall	≤ 0	$\mathcal{U}_c(0.2, 0.5)$	$\mathcal{U}_c(0.5, 1.0)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(6, 10)$	1/3	no
	≥ 10	$\mathcal{U}_c(0.1, 0.2)$	$\mathcal{U}_c(0.5, 1.0)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(1, 2)$	0.1/0	
GeneAseemble	≤ 5	$\mathcal{U}_c(0.1, 0.2)$	$\mathcal{U}_c(2, 5)$	$\mathcal{U}_c(2, 5)$	$\mathcal{U}_c(10, 30)$	1/0.5	yes
	≥ 10	$\mathcal{U}_c(0, 0.1)$	$\mathcal{U}_c(1, 2)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(5, 10)$	0.5/1	
Blast	≤ 5	$\mathcal{U}_c(0.5, 1)$	$\mathcal{U}_c(2, 3)$	$\mathcal{U}_c(1, 5)$	$\mathcal{U}_c(20, 40)$	1/2	no
	≥ 15	$\mathcal{U}_c(0, 0.1)$	$\mathcal{U}_c(1, 5)$	$\mathcal{U}_c(2, 3)$	$\mathcal{U}_c(5, 10)$	0.5/0.5	
LookUp	≤ 0	$\mathcal{U}_c(0.5, 0.7)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(4, 10)$	$\mathcal{U}_c(2, 8)$	0.5/0.25	yes
	≥ 1	$\mathcal{U}_c(0.5, 0.7)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(4, 10)$	$\mathcal{U}_c(2, 8)$	0.5/0.25	
Render	≤ 15	$\mathcal{U}_c(0.2, 0.3)$	$\mathcal{U}_c(5, 10)$	$\mathcal{U}_c(10, 15)$	$\mathcal{U}_c(150, 240)$	0.5/1	no
	≥ 30	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(5, 10)$	$\mathcal{U}_c(5, 10)$	$\mathcal{U}_c(80, 120)$	1/1	
Translate	≤ 10	$\mathcal{U}_c(0.7, 1.0)$	$\mathcal{U}_c(2, 5)$	$\mathcal{U}_c(1, 2)$	$\mathcal{U}_c(10, 40)$	0.5/1	no
	≥ 30	$\mathcal{U}_c(0.3, 0.4)$	$\mathcal{U}_c(0.1, 0.5)$	$\mathcal{U}_c(0.25, 0.5)$	$\mathcal{U}_c(5, 10)$	0.5/0.5	
Fold	≤ 15	$\mathcal{U}_c(0.25, 0.75)$	$\mathcal{U}_c(5, 20)$	$\mathcal{U}_c(5, 20)$	$\mathcal{U}_c(80, 400)$	3/5	yes
	≥ 45	$\mathcal{U}_c(0, 0.05)$	$\mathcal{U}_c(10, 20)$	$\mathcal{U}_c(20, 30)$	$\mathcal{U}_c(20, 30)$	1/1	
Print	≤ 0	$\mathcal{U}_c(0.4, 0.8)$	$\mathcal{U}_c(1, 2)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(8, 12)$	2/2	yes
	≥ 1	$\mathcal{U}_c(0.4, 0.8)$	$\mathcal{U}_c(1, 2)$	$\mathcal{U}_c(0, 0)$	$\mathcal{U}_c(8, 12)$	2/2	

TABLE 1
Distributions used to generate random offers for bioinformatics services.

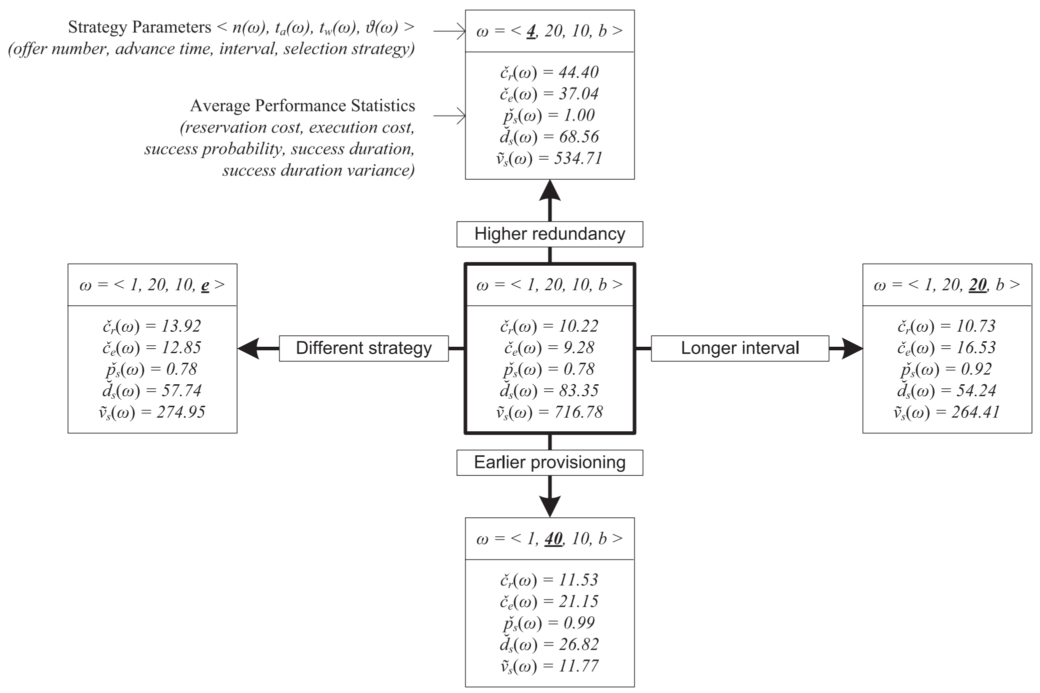

Fig. 2. Example high-level strategies for the *Fold* task.

($\vartheta(\omega) = \texttt{balanced}$)[3]. We note that this strategy is relatively cheap, has a success probability of only 78% and takes a long and highly uncertain time to complete. The remaining strategies shown in Figure 2 demonstrate the impact of slightly altering this reservation strategy — for example, when increasing the advance notice period from 20 to 40 time steps, the cost rises, but the success probability also increases to 99%, while the duration and its variance drop significantly (these trends all emerge from the distributions given in Table 1).

Now, these high-level strategies form the basic decisions available to our algorithm. In the following, we demonstrate how they are used in the context of the entire workflow.

3. Here, and in the remainder of the section, we abbreviate each selection strategy in {`cost`, `unreliability`, `end_time`, `balanced`} with its first letter.

5 Non-Urgent Scenario

First, we consider a scenario where the workflow has a relatively low value and a long deadline. Hence, it may represent a routine task for a scientist, which is not particularly critical to her work. More specifically, we assume the workflow has a maximum utility $u_{\max} = 150$, deadline $t_{\max} = 240$ (4 hours) and penalty $\delta = 1$ (i.e., no more utility is gained after 6.5 hours).

Figure 3 now shows the initial high-level decisions that our agent takes for the workflow. Here, the agent generally attempts to spend as little on services as possible, preferring to wait longer for completion. Thus, the

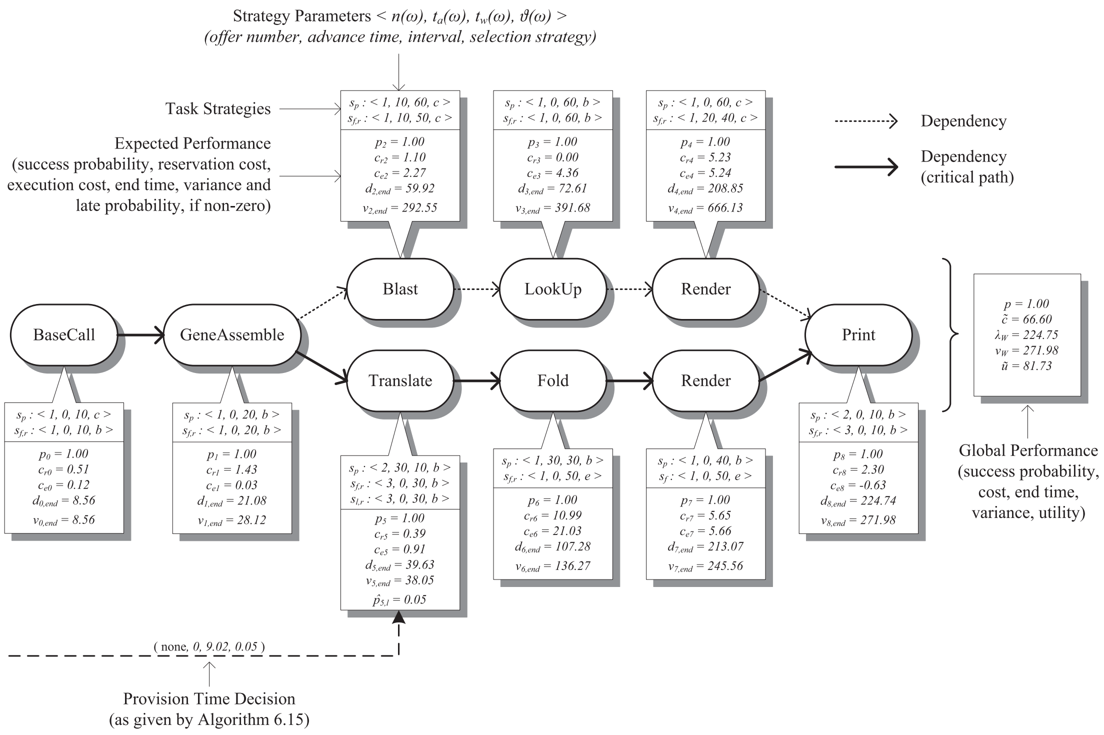
Strategy Parameters < n(ω), t_a(ω), t_w(ω), ϑ(ω) >
(offer number, advance time, interval, selection strategy)
Task Strategies
Expected Performance (success probability, reservation cost, execution cost, end time, variance and late probability, if non-zero)
s_p : < 1, 10, 60, c > s_f,r : < 1, 10, 50, c > p_2 = 1.00 c_r2 = 1.10 c_e2 = 2.27 d_2,end = 59.92 v_2,end = 292.55
s_p : < 1, 0, 60, b > s_f,r : < 1, 0, 60, b > p_3 = 1.00 c_r3 = 0.00 c_e3 = 4.36 d_3,end = 72.61 v_3,end = 391.68
s_p : < 1, 0, 60, c > s_f,r : < 1, 20, 40, c > p_4 = 1.00 c_r4 = 5.23 c_e4 = 5.24 d_4,end = 208.85 v_4,end = 666.13
Dependency
Dependency (critical path)
BaseCall
GeneAssemble
Blast
LookUp
Render
Translate
Fold
Render
Print
p = 1.00 ĉ = 66.60 λ_W = 224.75 v_W = 271.98 ũ = 81.73
s_p : < 1, 0, 10, c > s_f,r : < 1, 0, 10, b > p_0 = 1.00 c_r0 = 0.51 c_e0 = 0.12 d_0,end = 8.56 v_0,end = 8.56
s_p : < 1, 0, 20, b > s_f,r : < 1, 0, 20, b > p_1 = 1.00 c_r1 = 1.43 c_e1 = 0.03 d_1,end = 21.08 v_1,end = 28.12
s_p : < 2, 30, 10, b > s_f,r : < 3, 0, 30, b > s_l,r : < 3, 0, 30, b > p_5 = 1.00 c_r5 = 0.39 c_e5 = 0.91 d_5,end = 39.63 v_5,end = 38.05 p̂_5,l = 0.05
s_p : < 1, 30, 30, b > s_f,r : < 1, 0, 50, e > p_6 = 1.00 c_r6 = 10.99 c_e6 = 21.03 d_6,end = 107.28 v_6,end = 136.27
s_p : < 1, 0, 40, b > s_f : < 1, 0, 50, e > p_7 = 1.00 c_r7 = 5.65 c_e7 = 5.66 d_7,end = 213.07 v_7,end = 245.56
s_p : < 2, 0, 10, b > s_f,r : < 3, 0, 10, b > p_8 = 1.00 c_r8 = 2.30 c_e8 = -0.63 d_8,end = 224.74 v_8,end = 271.98
Global Performance (success probability, cost, end time, variance, utility)
(none, 0, 9.02, 0.05)
Provision Time Decision (as given by Algorithm 6.15)

Fig. 3. Workflow with initial high-level decisions.

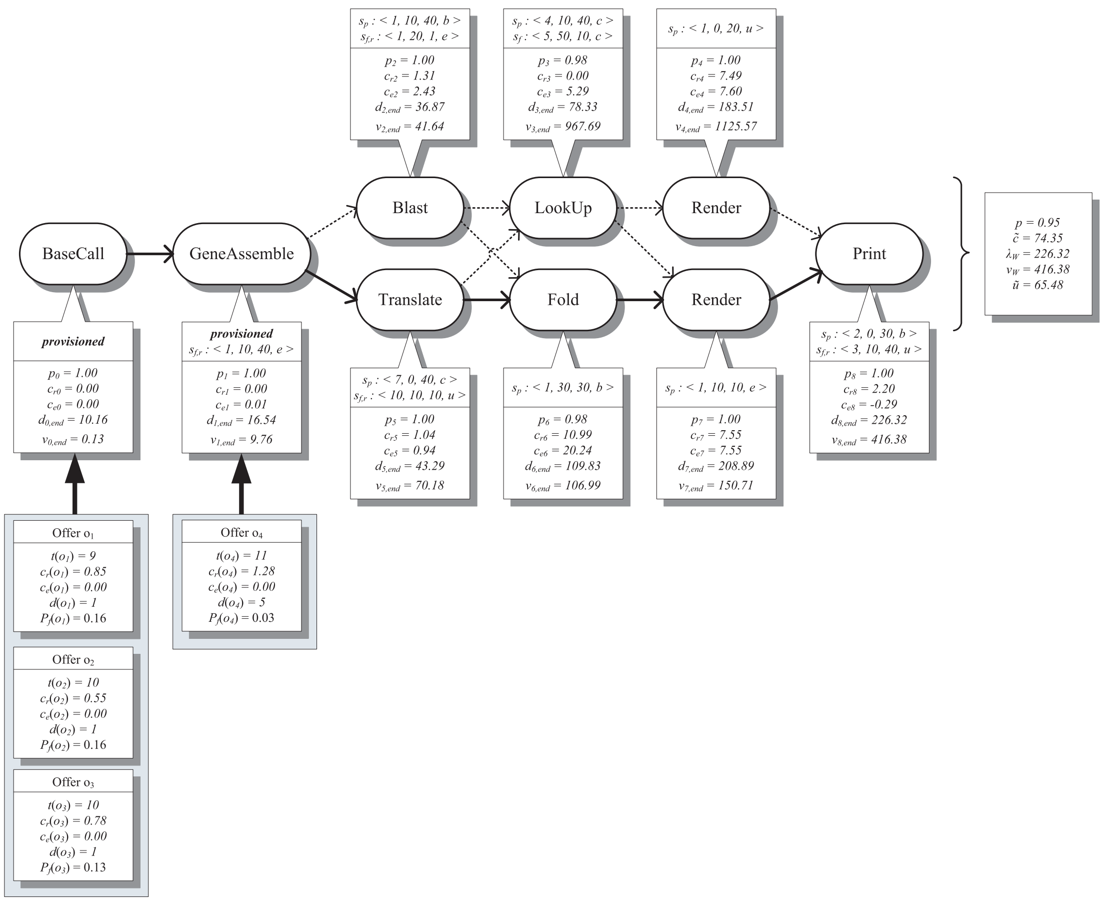
s_p : < 1, 10, 40, b > s_f,r : < 1, 20, 1, e > p_2 = 1.00 c_r2 = 1.31 c_e2 = 2.43 d_2,end = 36.87 v_2,end = 41.64
s_p : < 4, 10, 40, c > s_f : < 5, 50, 10, c > p_3 = 0.98 c_r3 = 0.00 c_e3 = 5.29 d_3,end = 78.33 v_3,end = 967.69
s_p : < 1, 0, 20, u > p_4 = 1.00 c_r4 = 7.49 c_e4 = 7.60 d_4,end = 183.51 v_4,end = 1125.57
BaseCall
GeneAssemble
Blast
LookUp
Render
Translate
Fold
Render
Print
p = 0.95 ĉ = 74.35 λ_W = 226.32 v_W = 416.38 ũ = 65.48
provisioned p_0 = 1.00 c_r0 = 0.00 c_e0 = 0.00 d_0,end = 10.16 v_0,end = 0.13
provisioned s_f,r : < 1, 10, 40, e > p_1 = 1.00 c_r1 = 0.00 c_e1 = 0.01 d_1,end = 16.54 v_1,end = 9.76
s_p : < 7, 0, 40, c > s_f,r : < 10, 10, 10, u > p_5 = 1.00 c_r5 = 1.04 c_e5 = 0.94 d_5,end = 43.29 v_5,end = 70.18
s_p : < 1, 30, 30, b > p_6 = 0.98 c_r6 = 10.99 c_e6 = 20.24 d_6,end = 109.83 v_6,end = 106.99
s_p : < 1, 10, 10, e > p_7 = 1.00 c_r7 = 7.55 c_e7 = 7.55 d_7,end = 208.89 v_7,end = 150.71
s_p : < 2, 0, 30, b > s_f,r : < 3, 10, 40, u > p_8 = 1.00 c_r8 = 2.20 c_e8 = -0.29 d_8,end = 226.32 v_8,end = 416.38
Offer o_1 t(o_1) = 9 c_r(o_1) = 0.85 c_e(o_1) = 0.00 d(o_1) = 1 P_f(o_1) = 0.16
Offer o_2 t(o_2) = 10 c_r(o_2) = 0.55 c_e(o_2) = 0.00 d(o_2) = 1 P_f(o_2) = 0.16
Offer o_3 t(o_3) = 10 c_r(o_3) = 0.78 c_e(o_3) = 0.00 d(o_3) = 1 P_f(o_3) = 0.13
Offer o_4 t(o_4) = 11 c_r(o_4) = 1.28 c_e(o_4) = 0.00 d(o_4) = 5 P_f(o_4) = 0.03

Fig. 4. Workflow after reserving initial offers.

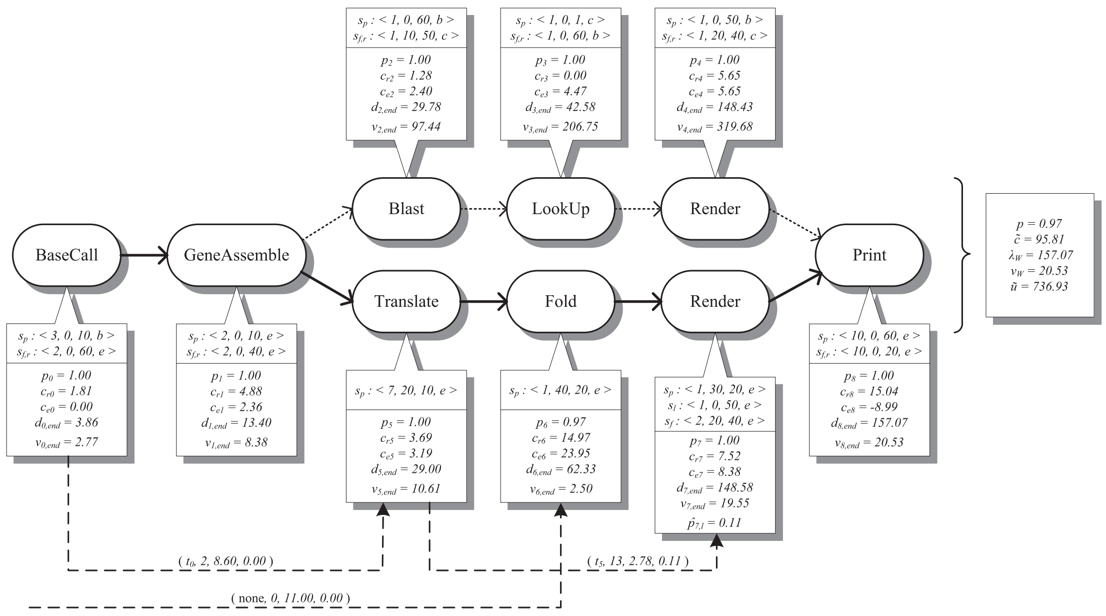

Fig. 5. Urgent workflow with initial high-level decisions.

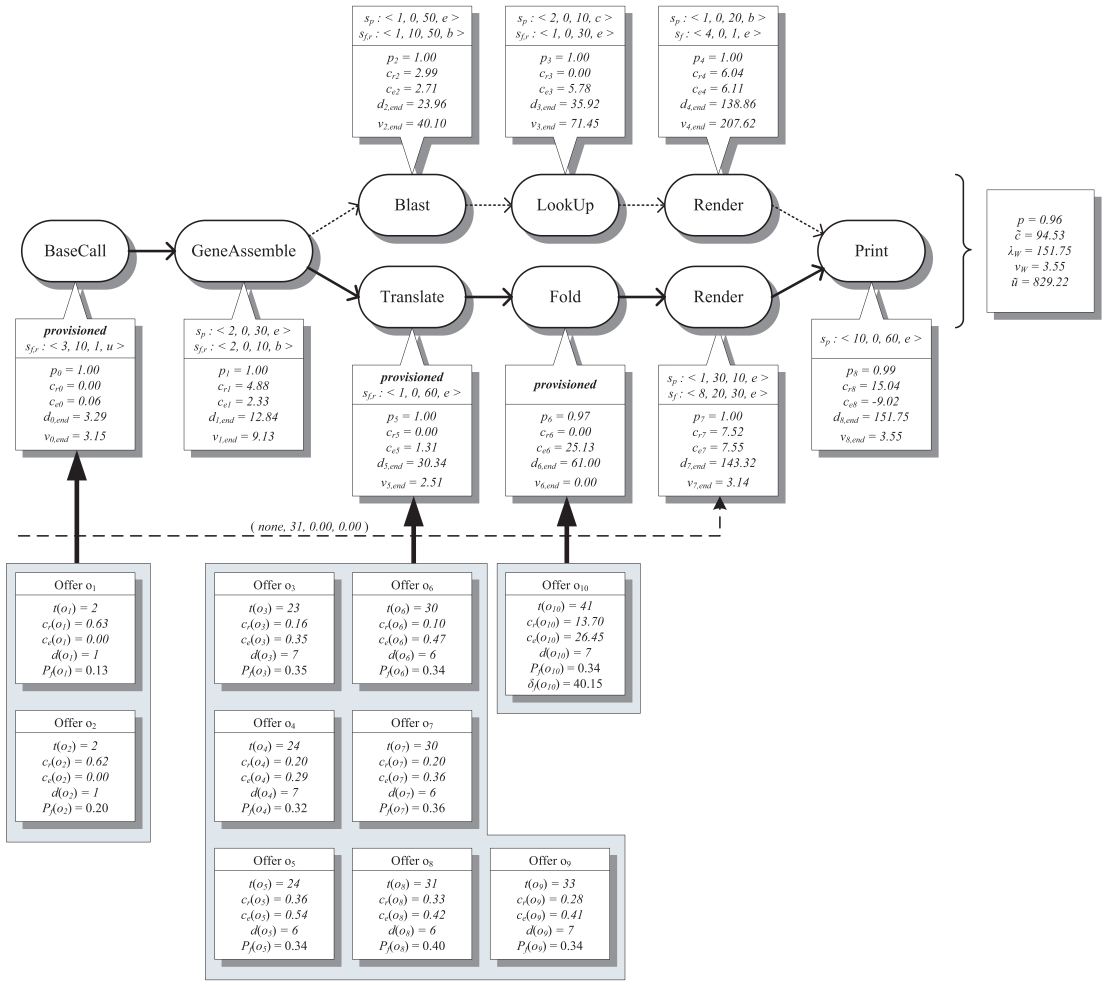

Fig. 6. Urgent workflow reserving initial offers.

algorithm decides to reserve only single offers for most tasks and relies on cautious contingency plans, where more single offers are reserved gradually (and repeatedly) in case of failure. The only parallel redundancy in the plan is used for tasks t_5 (*Translate*) and t_8 (*Print*), which are relatively cheap. Due to the longer deadline in this case, the consumer also decides to include few task overlaps in the workflow and instead prefers to leave the reservation of each task until all predecessors are complete. The only exception to this is t_5 (*Translate*), which the consumer chooses to reserve while its predecessors are still executing. Using Algorithm 1 in the main paper, the strategy here determines that the task should be reserved immediately when the workflow is started (for time step 30) and that this will result in a 5% probability of losing the reserved offers later on and an additional delay of 9.02 time steps. These figures are based on the uncertain duration of its predecessors (*BaseCall* and *GeneAssemble*), which are expected to complete by $t = 21.08$. Overall, the consumer expects the workflow to finish just before the deadline, after 224.75 time steps (but with considerable variance) and expects to spend \$66.60, thus achieving an expected utility of $\tilde{u} = 81.73$.

Next, Figure 4 shows the same workflow after the first offers have been reserved (during the inital time step). Here, the agent has consulted the market and followed its high-level strategies in reserving offers for some of the workflow tasks. In particular, the consumer has now reserved three offers for t_0 (*BaseCall*). In this case, it is different from the initial decision of reserving a single offer, as the agent immediately revises and improves its decisions as it observes the actual offers available on the market. Given these three offers, the task parameters are updated to reflect their terms (hence, the task end time is now almost certain). As is evident in the remainder of the workflow, the consumer has also now adapted its high-level strategies based on the new information. In particular, knowing that t_0 is almost certain to complete by time step $t = 11$, it has decided to reserve offers for task t_1 (*GeneAssemble*) earlier than originally planned. On the other hand, it also delayed reserving offers for task t_5 (*Translate*) to a later time. Finally, the consumer has introduced a number of additional edges into the workflow. Some of these have no impact on the estimated workflow utility, but the additional edge between t_3 (*LookUp*) and t_7 (*Render*) ensures that the former, slightly uncertain task completes before the expensive *Render* task is started.

In the next section, we contrast these decisions with a case where the workflow has a higher priority.

6 Urgent Scenario

In order to investigate how the agent's reservation decisions change as the workflow becomes more urgent and valuable, we now consider a second scenario. For this, we assume a maximum utility $u_{\max} = 1000$, a deadline $t_{\max} = 150$ and penalty $\delta = 20$. This could occur in the case where the results are critical to the scientist and when they are needed within a short amount of time (for example, for an impending project meeting or a publication deadline). Figure 5 illustrates the initial high-level reservation decisions, clearly highlighting an increasing reliance on redundant services and also on advance reservations, which here allows the agent to obtain better services and decrease the overall execution time (as services are reserved before their predecessors are completed). In more detail, the agent here decides to reserve some tasks immediately (such as *Fold*), but leaves the reservation of others until later (such as the lower *Render* task), according to the advance notice periods required and the expected completion times of their predecessors.

Next, Figure 6 shows the workflow after the first offers have been reserved. Here, the strategy has mostly followed its initial decisions. However, based on the offers reserved for task t_0 (*BaseCall*), it also immediately reserves offers for t_5 (*Translate*), rather than waiting an additional two time steps. As the offers were generally as expected, most remaining high-level decisions are unchanged and the overall expected utility has risen slightly, due to an earlier estimated completion time.

7 Conclusions

In this online appendix to [1], we have applied our algorithm to a real bioinformatics workflow. This work demonstrates how our approach dynamically decides when to reserve services in advance and how its reservation decisions are driven by the overall utility function, implicitly balancing the criticality and uncertainty of completing workflow tasks with the additional cost of introducing redundancy or selecting better service offers. Thus, our approach allows service consumers to execute complex workflows in a dependable manner, even when service providers are highly unreliable.

Acknowledgments

This work was funded by BAE Systems and the EPSRC.

References

[1] S. Stein, T. R. Payne, and N. R. Jennings, "Robust execution of service workflows using redundancy and advance reservations," *IEEE Transactions on Services Computing*, 2010.

[2] ——, "Flexible provisioning of web service workflows," *ACM Transactions on Internet Technology*, vol. 9, no. 1, pp. 2:1–2:45, 2009.

[3] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, S. Koranda, A. Lazzarini, G. Mehta, M. A. Papa, and K. Vahi, "Pegasus and the Pulsar Search: From Metadata to Execution on the Grid," in *Parallel Processing and Applied Mathematics: 5th International Conference, PPAM 2003, Czestochowa, Poland, September 7-10, 2003*, vol. 3019 / 2004, September 2003.

[4] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1067–1100, 2006.

[5] S. Coles, J. G. Frey, M. B. Hursthouse, M. E. Light, K. E. Meacham, D. J. Marvin, and M. Surridge, "ECSES - examining crystal structures using 'e-science': a demonstrator employing web and grid services to enhance user participation in crystallographic experiments," *Journal of Applied Crystallography*, vol. 38, no. 5, pp. 819–826, 2005.

[6] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim, "Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows," in *Proc. 22nd Conf. on AI, Canada*, 2007, pp. 1767–1774.

[7] T. M. Smith, C. Abajian, and L. Hood, "Hopper: software for automating data tracking and flow in DNA sequencing," *Comput. Appl. Biosci.*, vol. 13, no. 2, pp. 175–182, 1997.

[8] K. Kochut, J. Arnold, A. Sheth, J. Miller, E. Kraemer, B. Arpinar, and J. Cardoso, "IntelliGEN: A distributed workflow system for discovering protein-protein interactions," *Distributed and Parallel Databases*, vol. 13, no. 1, pp. 43–72, 2003.

[9] A. O'Brien, S. Newhouse, and J. Darlington, "Mapping of scientific workflow within the e-protein project to distributed resources," in *Proceedings of UK e-science All Hands Meeting (AHM 2004), Nottingham, UK*. EPSRC, 2004, pp. 404–409.

[10] B. Ewing, L. Hillier, M. C. Wendl, and P. Green, "Base-calling of automated sequencer traces using phred. i. accuracy assessment," *Genome Research*, vol. 8, no. 3, pp. 175–185, 1998.

[11] M. H. DeGroot and M. J. Shervish, *Probability and Statistics*, third edition ed. Addison-Wesley, 2002.