Grid Workflow Scheduling In WOSE

Yash Patel [1], Andrew Stephen M^{c}Gough [2], John Darlington [3]
London e-Science Centre, Department of Computing, Imperial College
South Kensington Campus, London SW7 2AZ, United Kingdom
{yp03 [1], asm [2], jd [3]}@doc.ic.ac.uk

***Abstract*— The success of web services has infuenced the way in which grid applications are being written. Grid users seek to use combinations of web services to perform the overall task they need to achieve. In general this can be seen as a set of services with a workflow document describing how these services should be combined. The user may also have certain constraints on the workflow operations, such as execution time or cost to the user, specified in the form of a Quality of Service (QoS) document. These workflows need to be mapped to a subset of the Grid services taking the QoS and state of the Grid into account – service availability and performance. We propose in this paper an approach for generating constraint equations describing the workflow, the QoS requirements and the state of the Grid. This set of equations may be solved using Integer Linear Programming (ILP), which is the traditional method. We further develop a 2-stage stochastic ILP which is capable of dealing with the volatile nature of the Grid and adapting the selection of the services during the life of the workflow. We present experimental results comparing our approaches, showing that the 2-stage stochastic programming approach performs consistently better than other traditional approaches. This work forms the workflow scheduling service within WOSE (Workflow Optimisation Services for e-Science Applications), which is a collaborative work between Imperial College, Cardiff University and Daresbury Laborartory.**

I. Introduction

Grid Computing has been evolving over recent years towards the use of service orientated architectures [1]. Functionality within the Grid exposes itself through a service interface which may be a standard web service endpoint. This functionality may be exposing computational power, storage, software capable of being deployed, access to instruments or sensors, or potentially a combination of the above.

Grid workflows that users write and submit may be abstract in nature, in which case the final selection of web services has not been finalised. We refer to the abstract description of services as abstract services in this paper. Once the web services are discovered and selected, the workflow becomes concrete, meaning the web services matching the abstract description of services are selected.

The Grid is by nature volatile – services appear and disappear due to changes in owners policies, equipment crashing or network partitioning. Thus submitting an abstract workflow allows late binding of the workflow with web services currently available within the Grid. The workflow may also take advantage of new web services which were not available at the time of writing. Users who submit a workflow to the Grid will often have constraints on how they wish the workflow to perform. These may be described in the form of a QoS document which details the level of service they require from the Grid. This may include requirements on such things as the overall execution time for their workflow; the time at which certain parts of the workflow must be completed; and the cost of using services within the Grid to complete the workflow.

In order to determine if these QoS constraints can be satisfied it is necessary to store historic information and monitor performance of different web services within the Grid. Such information could be performance data related to execution and periodic information such as queue length, availability. Here we see that existing Grid middleware for performance repositories may be used for the storage and retrieval of this data. If the whole of the workflow is made concrete at the outset, it may lead to QoS violations. Therefore we have adopted an iterative approach. At each stage the workflow is divided into those abstract services which need to be deployed now and those that can be deployed later. Those abstract services which need to be deployed now are made concrete and deployed to the Grid. However, to maintain QoS constraints it is necessary to ensure that at each iteration the selected web services will still allow the whole workflow to achieve QoS.

This paper presents results of the workflow scheduling service within WOSE (Workflow Optimisation Services for e-Science Applications). WOSE is an EPSRC-funded project jointly conducted by researchers at Imperial College, Cardiff University and Daresbury Laboratory. We discuss how our work relates to others in the field in Section II. Section III describes the process of workflow aware performance guided scheduling, followed by a description of the 2-stage stochastic programming approach and an algorithm for stochastic scheduling in Section IV. In Section V we illustrate how our approach performs through simulation before concluding in Section VI.

II. Related Work

Business Process Execution Language (BPEL) [2] is beginning to become a standard for composing web-services and many projects such as Triana [3] and WOSE [4] have adopted it as a means to realise service-based Grid workflow technology. These projects provide tools to specify abstract workflows and workflow engines to enact workflows. Buyya et al [5] propose a Grid Architecture for Computational Economy

TABLE I

SCHEDULING PARAMETERS.

Symbol	**Name**
A_i	Abstract service i
a_{ir}, c_{ir}, x_{ir}	Expected time, cost and selection variable associated with r^{th} web service matching A_i
$time_{QoS}$	Maximum time in which the workflow should get executed
$deadline_i$	Time in which A_i is expected to complete
$\|A\|$	Number of abstract services
$\|a_i\|$	Number of web services matching A_i

(GRACE) considering a generic way to map economic models into a distributed system architecture. The Grid resource broker (Nimrod-G) supports deadline and budget based scheduling of Grid resources. However no QoS guarantee is provided by the Grid resource broker. Zeng et al [6] investigate QoS-aware composition of Web Services using integer programming method. The services are scheduled using local planning, global planning and integer programming approaches. The execution time prediction of web services is calculated using an arithmetic mean of the historical invocations. However Zeng et al assume that services provide upto date QoS and execution information based on which the scheduler can obtain a service level agreement with the web service. Brandic et al [7] extend the approach of Zeng et al to consider application-specific performance models. However their approach fails to guarantee QoS over entire life-time of a workflow. They also assume that web services are QoS-aware and therefore certain level of performance is guaranteed. However in an uncertain Grid environment, QoS may be violated. Brandic et al have no notion of global planning of a workflow. Thus there is a risk of QoS violation. Huang et al [8] have developed a framework for dynamic web service selection for the WOSE project. However it is limited only to best service selection and no QoS issues are considered. We see our work fitting in well within their optimisation service of the WOSE architecture. A full description of the architecture can be found in [8]. Our approach not only takes care of dynamically selecting the optimal web service but also makes sure that overall QoS requirements of a workflow is satisfied with sufficiently high probability. The main contribution of our paper is the novel QoS support approach and an algorithm for stochastic scheduling of workflows in a volatile Grid.

III. WORKFLOW AWARE PERFORMANCE GUIDED SCHEDULING

We provide Table: I as a quick reference to the parameters of the ILP.

A. Deterministic Integer Linear Program (ILP)

Before presenting our 2-stage stochastic integer linear program we first present the deterministic ILP program. The program is integer linear as it contains only integer variables (unknowns) and the constraints appearing in the program are all linear. The ILP consists of an objective which we wish to minimise along with several constraints which need to be satisfied. The objective here is to minimise the overall workflow cost:

$$Cost = minimize[O] \tag{1}$$

$$O = \sum_{i}^{|A|} \sum_{r}^{|a_i|} c_{ir} x_{ir} \tag{2}$$

O is the cost associated with web services. We have identified the following constraints.

- **Selection Constraint :**

$$\forall i, \sum_{r}^{|a_i|} x_{ir} = 1 \tag{3}$$

$$x_{ir} \in \{0, 1\} \tag{4}$$

  Equation 3 takes care of mapping A_i to one and only one web service. For each A_i, only one of the x_{ir} equals 1, while all the rest are 0.
- **Deadline Constraint :** Equation 5 ensures that A_i finishes within the assigned deadline.

$$\sum_{r}^{|a_i|} a_{ir} x_{ir} \leq deadline_i \tag{5}$$

- **Other workflow specific constraints :** These constraints are generated based on the workflow nature and other soft deadlines (execution constraints). This could be explicitly specified by the end-user. e.g. some abstract service or a subset of abstract services is required to be completed within t seconds. These could also be satisfying other QoS parameters such as reliability and availability. A full list of constraints is beyond the scope of this paper.

IV. TWO-STAGE STOCHASTIC ILP WITH RECOURSE

Stochastic programming, as the name implies, is mathematical (i.e. linear, integer, mixed-integer, nonlinear) programming but with a stochastic element present in the data. By this we mean that in deterministic mathematical programming the data (coefficients) are known numbers while in stochastic programming these numbers are unknown, instead we may have a probability distribution present. However these unknowns, having a known distribution could be used to generate a finite number of deterministic programs through techniques such as Sample Average Approximation (SAA) and an ϵ-optimal solution to the true problem could be obtained. A full discussion of SAA is beyond the scope of this paper and interested readers may refer [9].

Consider a set S of abstract services that can be scheduled currently and concurrently. Let $|S|$ be the number of such services. Similarly let P be the set of unscheduled abstract services and $|P|$ be its number. Equations (6) to (9) represent a 2-stage stochastic program with recourse, where stage-1 minimises current costs and stage-2 aims to minimise future costs. The recourse term is $Q(x_S, \omega)$, which is the future cost. The term $e^T z$ in the objective of the stage-2 program is the penalty incurred for failing to compute a feasible schedule. The vector e has values such that the incurred penalty is clearly apparent in the objective value. The z variables are also present in the constraints of stage-2 programs in order to keep the program feasible as certain realisations of random variables will make the program infeasible. The vector z consists of continuous variables whose size depends on the number of constraints appearing in the program.

$$Cost = minimise[O + E(Q(x_S, \omega))] \tag{6}$$

- **Stage-1**

$$O = \sum_{i}^{|S|} \sum_{r}^{|a_i|} c_{ir} x_{ir} \tag{7}$$

Subject to the following constraints: selection, scheduling along with other possible constraints.

- **Stage-2**

ω is a vector consisting of random variables of runtimes and costs of services. x_S is the vector denoting the solutions of stage-1. Q(x_S,ω) is the optimal solution of

$$Cost_\xi \quad = \quad minimise[\xi] + \mathbf{e^T z} \tag{8}$$

$$\xi \quad = \quad \sum_{i}^{|P|} \sum_{r}^{|a_i|} c_{ir} x_{ir} \tag{9}$$

Subject to the following constraints: selection, scheduling along with other possible constraints. ξ is a realisation of expected costs of using services. The function E is the expected objective value of stage-2, which is computed using the SAA problem listed in equation (10). The stage-2 solution can be used to recompute stage-1 solution, which in turn leads to better stage-2 solutions.

$$minimise[O + \frac{1}{N} \sum_{n=1}^{N} Q(x_S, \xi^n)] \tag{10}$$

$$N \geq \frac{3\sigma^2_{max}}{(\epsilon - \delta)^2} \frac{log|F|}{\alpha} \tag{11}$$

In equation (11), $|F|$ is the number of elements in the feasible set, which is the set of possible mappings of abstract services to real Grid services. 1 - α is the desired probability of accuracy, δ the tolerance, ϵ the distance of solution to true solution and σ^2_{max} is the maximum execution time variance of a particular service in the Grid. One could argue that it may not be trivial to calculate both σ^2_{max} and $|F|$. Maximum execution time variance of some Grid service could be a good approximation for σ^2_{max} and $|F|$ could be obtained with proper discretisation techniques. Equation (11) is derived in [10]. Our scheduling service provides a 95% guarantee. Hence 1 - α is taken as 0.95. ϵ - δ is taken as 2 for convenience, while $log|F|$ turns out to be approximately equal to 4. In our case in order to obtain 95% confidence level, N approximately turns out to be around 600. This means that one needs to solve nearly 600 deterministic ILP programs in stage-2 for each iteration of algorithm 1. The number of unknowns in the ILP being only about 500, negligible time is spent to solve these many scenarios.

A. Algorithm for stochastic scheduling of workflows

Algorithm 1 obtains scheduling solutions for abstract workflow services by solving 2-stage stochastic programs, where stage-1 minimises current costs and stage-2 minimises future costs. This algorithm guarantees an ϵ-optimal solution (i.e., a solution with an absolute optimality gap of ϵ to the true solution) with desired probability [9]. However to achieve the desired accuracy one needs to sample enough scenarios, which often get quite big in a large utility grid, and in a service rich environment with continuous execution time distributions associated with Grid services, the number of scenarios is theoretically infinite. However with proper discretisation techniques the number of scenarios or the sample size required to get the desired accuracy is at most linear in the number of Grid services. This is clearly evident from the value of N (equation (11)), which is the sample size, as $|F|$ being the size of feasible set, is exponential in the number of Grid services. Finally statistical confidence intervals are then derived on the quality of the approximate solutions.

Algorithm 1 initially obtains scheduling solutions for stage-1 abstract services, S in the workflow. This stage-1 result puts constraints on stage-2 programs, which aims at finding scheduling solutions for rest of the unscheduled workflow. The sampling size (equation (11)) for each iteration, guarantees an ϵ-optimal solution to the true scheduling problem with desired accuracy, 95% in our case. If the optimality gap or variance of the gap estimator are small, only then the scheduling operation is a success. If not, the iteration is repeated as mentioned in step 3.6 of the algorithm. This leads to computing new schedule for stage-1 abstract services with tighter QoS bounds. When the scheduled stage-1 abstract services finish execution, algorithm 1 is used to schedule abstract services that follow them in the workflow. Step 4 selects the stage-1 solution, which has a specified tolerance δ to the true problem with probability at least equal to specified confidence level 1 - α.

$$L \quad = \quad \frac{\sum_{m=1}^{M} O^m}{M} \tag{12}$$

$$Var^L \quad = \quad \frac{\sum_{m=1}^{M} (O^m - L)^2}{M(M-1)} \tag{13}$$

$$U \quad = \quad O_1 + \frac{1}{N'} \sum_{n=1}^{N'} Q(x_S, \xi^n) \tag{14}$$

$$Var^U \quad = \quad \frac{\sum_{n=1}^{N'} (Q(x_S, y_S, \xi^n) - U)^2}{N'(N'-1)} \tag{15}$$

Algorithm 1 initially obtains scheduling solutions for stage-1 abstract services, S in the workflow. This stage-1 result puts constraints on stage-2 programs, which aims at finding scheduling solutions for rest of the unscheduled workflow. The sampling size (eq. 11) for each iteration, guarantees an ϵ-optimal solution to the true scheduling problem with desired accuracy, 95% in our case. If the optimality gap or variance of the gap estimator are small, only then the scheduling operation is a success. If not, the iteration is repeated as mentioned in step 3.6 of the algorithm. This leads to computing new

Algorithm 1 Algorithm for stochastic scheduling

Step 1 : Choose sample sizes N and $N^{'} \geq N$, iteration count M, tolerance ϵ and rule to terminate iterations
Step 2 : Check if termination is required
for m = 1, . . .,M **do**
  Step 3.1 : Generate N samples, and solve the SAA problem, let the optimal objective be O^m for corresponding iteration
end for
Step 3.2 : Compute a lower bound estimate L (eq. 12) on the objective and its variance Var^L (eq. 13)
Step 3.3 : Generate $N^{'}$ samples, use one of the feasible stage-1 solution and solve the SAA problem to compute an upper bound estimate U (eq. 14) on the objective and its variance Var^U (eq. 15)
Step 3.4 : Estimate the optimality gap (Gap = $|L-U|$) and the variance of the gap estimator (Var^{Gap} = Var^L + Var^U)
Step 3.5 : If Gap and Var^{Gap} are small, choose stage-1 solution. Stop
Step 3.6 : If Gap and Var^{Gap} are large, tighten stage-1 QoS bounds, increase N and/or $N^{'}$, goto **step 2**

schedule for stage-1 abstract services with tighter QoS bounds. Step 3.5 selects the stage-1 solution, which has a specified tolerance δ to the true problem with probability at least equal to specified confidence level 1 - α.

V. Experimental Evaluation

In this section we present experimental results for the ILP techniques described in this paper.

A. Setup

Table II summarises the experimental setup. We have performed 3 simulations and for each different setup of a simulation we have performed 10 runs and averaged out the results. Initially 500 jobs allow the system to reach steady state, the next 1000 jobs are used for calculating statistics such as mean execution time, mean cost, mean failures, mean partial executions and mean utilisation. The last 500 jobs mark the ending period of the simulation. Mean of an abstract service is measured in millions of instructions (MI). In order to compute expected runtimes, we put no restriction on the nature of execution time distribution and apply Chebyshev inequality [11] to compute expected runtimes such that 95% of jobs would execute in time under a_{ir} (equation (16)). It should be noted that such bounds or confidence intervals on the execution times can also be computed using other techniques such as Monte Carlo approach [12] and Central Limit Theorem [11] or by performing finite integration, if the underlying execution time PDFs (Probability Density Functions) are available in analytical forms. The waiting time is also computed in such a way that in 95% of the cases, the waiting time encountered will be less than the computed one. The value 4.47 appearing in the equations below is due to applying Chebyshev inequality for including 95% of the execution or waiting time distribution area. In equation (16), μ_{ir} and σ_{ir} are the mean and standard deviation of the execution time distribution of a running software service. c_{ir} is a simple product function of a_{ir}.

$$a_{ir} = \mu_{ir} + 4.47\sigma_{ir} + waiting\ time \tag{16}$$

a_{ir} (equation (17)) for stage-2 programs is calculated in a slightly different fashion.

$$a_{ir} = \xi^e(\mu_{ir}, \sigma_{ir}^2) + \xi^w(\mu_r, \sigma_r^2) \tag{17}$$

Here ξ^e is the execution time distribution sample of an abstract service on a Grid service. ξ^w is the waiting time distribution sample associated with R_r. We have used Monte-Carlo [12] technique for sampling values out of the distributions. Other sampling techniques such as Latin Hypercube sampling could also be used in place. We provide an example for calculating initial deadlines, given by equation (18) for the first abstract service (generate matrix) of workflow type 1. Deadline calculation of an abstract service takes care of all possible paths in a workflow and scaling is performed with reference to the longest execution path in a workflow. Equation (18) is scaled with reference to $time_{QoS}$. It should be noted that initially implies calculation before performing the iterations of algorithm 1. Subsequent deadlines of abstract services in a workflow are calculated initially by scaling with reference to the remaining workflow deadline.

$$\begin{aligned} deadline_1 &= \frac{X_1}{X_1 + X_{234}} time_{QoS} && (18)\\ X_1 &= \mu_i^{max}(1 + 4.47CV_i^{max}) && (19)\\ X_{234} &= \sum_{j=2}^{4} \mu_j^{max}(1 + 4.47CV_j^{max}) && (20) \end{aligned}$$

Initial deadline calculation is done in order to reach an optimal solution faster. We are currently investigating cut techniques which can help to reach optimal solutions even faster. Here μ_i^{max} and CV_i^{max} are the mean and coefficient of variation of a Grid service that has the maximum expected runtime. If Gap and Var^{Gap} are large, bounds are tightened in such a way that in the next iteration they become smaller. e.g. minimum coefficient for time (a_{ir}) could be set as the deadline or recourse term variable values (z) in the stage-2 programs could be used to tighten deadline. The workflows experimented with are shown in figure 1. The workflows are simulation counterparts of the real world workflows. Their actual execution is a delay based on their execution time distribution, as specified in table II. In the first simulation, type 1 workflows are used, in the second simulation, type 2 workflows are used and in the third simulation workload is made heterogenous (HW). The type 1 workflow is quite simple compared to type 2, which is a real scientific workflow. All the workflows have different QoS requirements as specified in table II. The ILP solver used is CPLEX by ILOG [13], which is one of the best industrial quality optimisation software. The simulation is developed on top of simjava 2 [14], a discrete event simulation package. The Grid size is kept small in order to get an asymptotic

TABLE II

SIMULATION PARAMETERS.

Simulation	**1**	**2**	**3**
Services matching A_i	24	12	24
Service speed (kMIPS)	3-14	3-14	3-14
Unit cost (per sec)	5-29	5-29	5-29
Arrival Rate (λ) (per sec)	1.5-10	0.1-2.0	1.5-3.6
A_i Mean (μ) (kMI)	7.5-35	10-30	7.5-35
A_i CV = σ/μ	0.2-2.0	0.2-1.4	0.2-2.0
Workflows	Type 1	Type 2	HW
$time_{QoS}$ (sec)	40-60	80-100	40-60

behaviour of workflow failures as coefficient of variation (CV) of execution or arrival rates (λ) are increased.

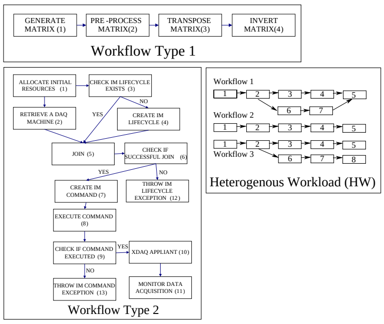

Fig. 1. Workflows

B. Results

We compare our scheme (DSLC) with two traditional schemes (DDLC and SDLC), all with a common objective of minimising cost and ensuring workflows execute within deadlines. The workflows don't have any slack period, meaning they are scheduled without any delay as soon as they are submitted. DDLC (dynamic, deterministic, least cost satisfying deadlines) and DSLC (dynamic, stochastic, least cost satisfying deadlines) job dispatching strategies calculate an initial deadline based on equation (18). Though DDLC calculates new deadlines each time it needs to schedule abstract services, the deadlines don't change once they are calculated. The deadlines get changed iteratively in case of DSLC due to the iterative nature of algorithm 1. Scheduling of abstract services continues until the lifetime of workflows in case of DDLC and DSLC. It is not the case with SDLC (static, deterministic, least cost satisfying deadlines) and as soon as the workflows are submitted, an ILP is solved and scheduling solutions for all abstract services within the workflows are obtained. In case of SDLC, once the scheduling solutions are obtained, they don't get changed during the entire lifetime of the workflows. The main comparison metrics here are mean cost, mean time, failures and mean utilisation as we increase λ and CV. However we will keep our discussion limited to failures as a workflow failure means failure in satisfying QoS requirements of workflows.

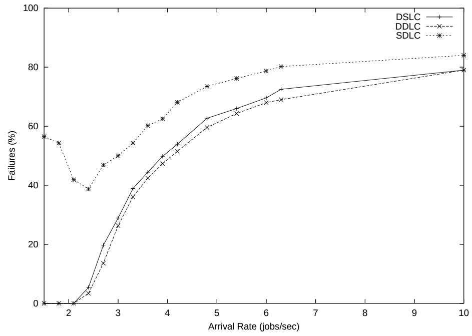

Fig. 2. Failures vs λ, CV = 0.2 (Simulation 1)

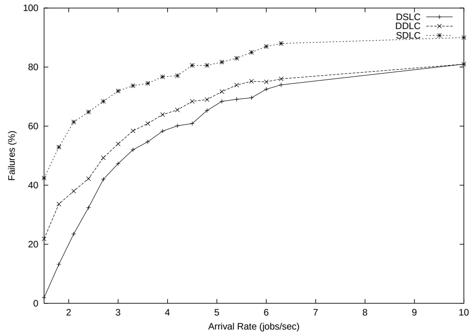

Fig. 3. Failures vs λ, CV = 1.8 (Simulation 1)

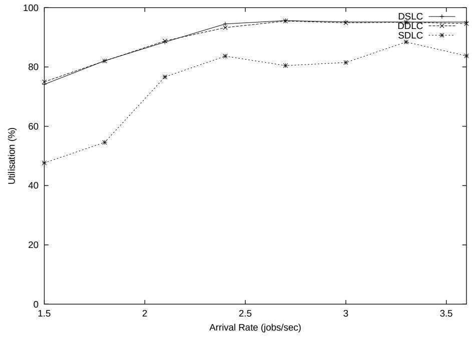

Fig. 4. Avg Utilisation vs λ, CV = 0.2 (Simulation 1)

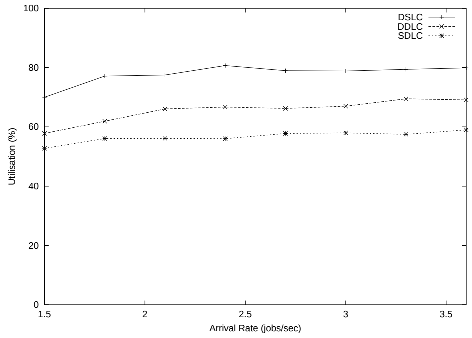

Fig. 5. Avg Utilisation vs λ, CV = 1.8 (Simulation 1)

C. Effect of arrival rate and workload

We see that in case of figures 2 and 3, as λ increases, DSLC continues to outperform other schemes. This trends continues however but with a reduced advantage. This can be explained as follows. This trends continues however but the advantage keeps on reducing as arrival rates increase. This can be explained as follows. When arrival rates increase, more work needs to be scheduled in the same amount of time, as previously available. Moreover it is safe to assume that response time of services is an increasing function of arrival rate. Hence failures increase. Moreover this behaviour not being linear and failures themselves reaching a limiting value, this advantage is reduced. SDLC obtains a joint solution and therefore is a sub-optimal solution or is optimal only at the time of scheduling. Hence more failures are registered in case of SDLC. Referring to figures 4 and 5, it is apparent that when CV is low, utilisation in case of DSLC and DDLC turns out to be the same. However SDLC also registers reasonable utilisation. Overall utilisation is maximum in case of DSLC due to its capability of obtaining optimal solutions. When CV is high, DSLC still outperforms other schemes. Due to high unpredicatability, DDLC and SDLC register moderate utilisations. In case of workflow type 2, for low and high CVs, as λ is increased, DSLC outperforms all other schemes. In case of utilisation, for low CV, all schemes register high utilisations. However in case of high CV, DSLC registers far higher utilisation than other schemes. Referring to figures 12 and 13, again DSLC registers lowest failures for both low and high CVs. This is because workload is quite heterogenous and environment therefore becomes quite unpredictable. In this case DSLC obtains better scheduling solutions than other schemes. In case of utilisation (figures 14 and 15), again due to less failures in case of DSLC, utilisation is registered higher than other schemes.

D. Effect of CV

We see that in case of workflow type 1, which is quite predictable and sequential, as arrival rates increase, for low CV (predictable behaviour), DDLC performs slightly better than DSLC. This is because even if DSLC iteratively tightens deadlines, it doesn't help to get a better schedule due to highly predictable environment and as a result failures increase slightly as it tries to schedule workflows which would have failed in case of DDLC. As CV is increased, we see that DSLC outperforms other schemes. This is because the environment becomes less predictable and algorithm 1 obtains better deadline solutions solutions that help to reduce failures. DDLC closes the gap asymptotically as λ increases. This is because failures increase as λ increases and theoretically the workflows themselves cannot be scheduled as they would fail to meet their deadlines. In case of workflow type 2, for both low and high CVs, DSLC performs significantly better than other schemes. Referring to figures 10 and 11, we see that as CV is increased for low arrival rates, utilisation drops which indicates that failures increase, which in turn indicates that environment becomes more and more unpredictable. With high workloads, as CV is increased, utilisation drops, but this time DSLC registers highest utilisation. SDLC and DDLC register lower utilisation as they fail to cope with the increasing uncertainty. However for low CV, they all start off from about the same utilisation mark. When workload is made heterogenous, for both low and high CVs, DSLC outperforms other schemes. For high CV, the environment becomes highly uncertain and hence SDLC registers a spiky behaviour in utilisation. This is in agreement considering its static nature of job assignment.

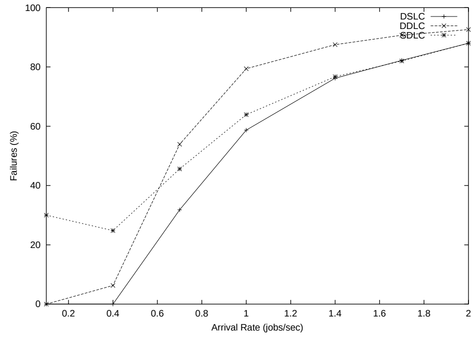

Fig. 6. Failures vs λ, CV = 0.2 (Simulation 2)

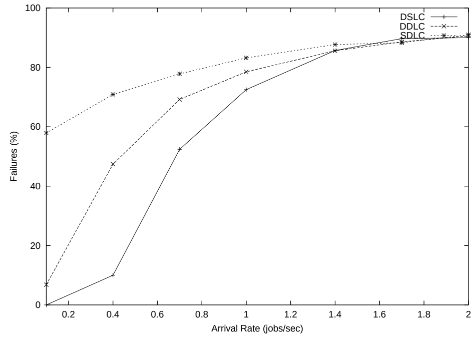

Fig. 7. Failures vs λ, CV = 1.4 (Simulation 2)

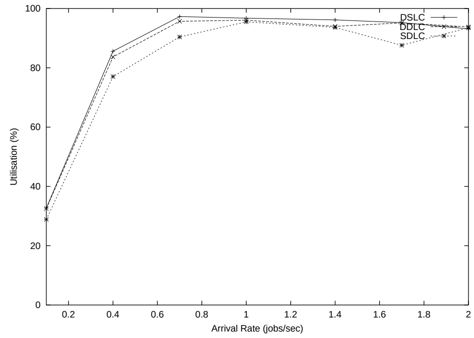

Fig. 8. Avg Utilisation vs λ, CV = 0.2 (Simulation 2)

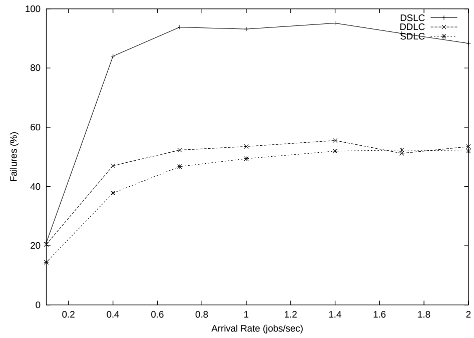

Fig. 9. Avg Utilisation vs λ, CV = 1.4 (Simulation 2)

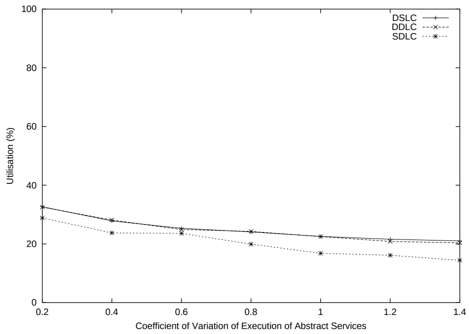

Fig. 10. Avg Utilisation vs CV, λ = 0.1 (Simulation 2)

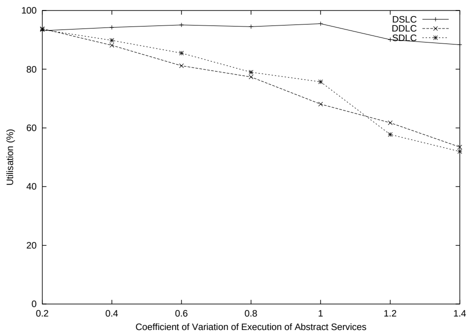

Fig. 11. Avg Utilisation vs CV, λ = 2.0 (Simulation 2)

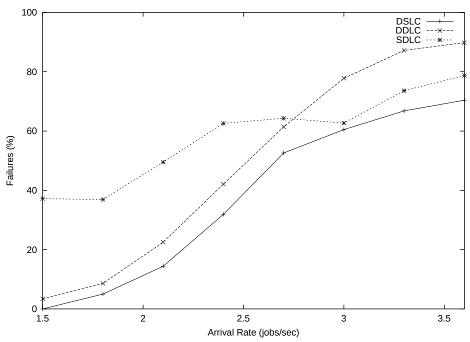

Fig. 12. Failures vs λ, CV = 0.2 (Simulation 3)

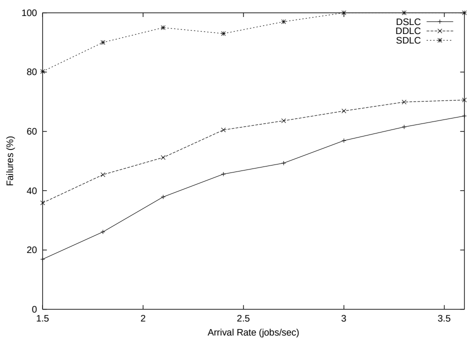

Fig. 13. Failures vs λ, CV = 1.8 (Simulation 3)

E. Effect of workflow nature

Workflow type 2 is more complex and far less predictable than workflow type 1. Hence in such case we see that DSLC outperforms other schemes for low and high CVs. This is to say that DSLC algorithm obtains better deadline solutions by solving the SAA problem than other schemes, as a result of

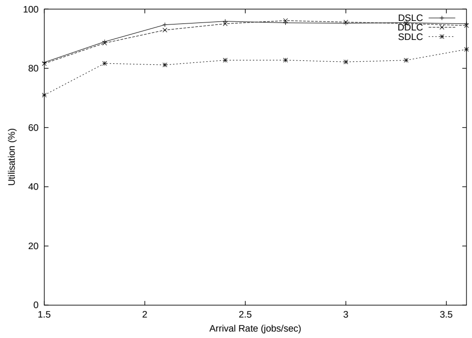

Fig. 14. Avg Utilisation vs λ, CV = 0.2 (Simulation 3)

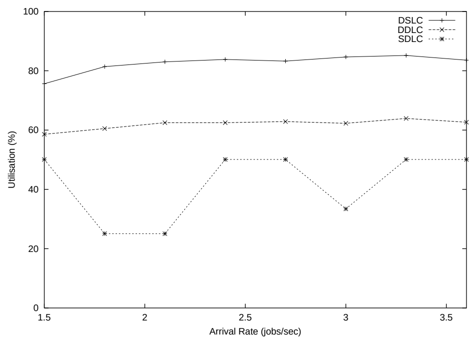

Fig. 15. Avg Utilisation vs λ, CV = 1.8 (Simulation 3)

which less failures are experienced. The other schemes, since they obtain static deadlines, fail to outperform DSLC. However when λ increases, all the curves merge to values closer to 100%. In case of heterogenous workload, the environment again becomes less predictable and as a result DSLC continues to outperform other schemes.

VI. Conclusion and Future Work

We have developed a 2-stage stochastic programming approach to workflow scheduling using an ILP formulation of QoS constraints, workflow structure, performance models of Grid services and the state of the Grid. The approach gives a considerable improvement over other traditional schemes. This is because SAA approach obtains ϵ-optimal solutions minimised and approximated over uncertain conditions while providing QoS guarantee over the workflow time period. The developed approach performs considerably better particularly when the CV of execution times and the workflow complexity are high. At both low and high arrival rates, the developed approach comfortably outperforms the traditional schemes.

As future work we seek to extend our model of Grid services and the constraints on these. This will enable us to more accurately schedule workflows onto the Grid. As the number of constraints increase along with a greater number of Grid services we see that the solution time of the ILP may become significant. A parallel approach may be used to improve on this situation. We would like to perform experiments with workflows having a slack period, meaning workflows can wait for sometime before getting serviced. We would also like to develop pre-optimisation techniques that would decrease the unknowns requiring to be solved in the ILP. i.e. prune certain Grid services from the ILP that cannot improve the expectation of its objective.

REFERENCES

[1] N. Furmento, J. Hau, W. Lee, S. Newhouse, and J. Darlington, "Implementations of a Service-Oriented Architecture on top of Jini, JXTA and OGSI," in *Grid Computing: Second European AcrossGrids Conference, AxGrids 2004*, ser. Lecture Notes in Computer Science, vol. 3165, Nicosia, Cyprus, Jan. 2004, pp. 90–99.

[2] *BPEL Specification*, Std. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/2003/

[3] S. Majithiaa, M. S. Shields, I. J. Taylor, and I. Wang, "Triana: A Graphical Web Service Composition and Execution Toolkit," *International Conference on Web Services*, 2004.

[4] "WOSE (Workflow Optimisation Services for e-Science Applications)." [Online]. Available: http://www.wesc.ac.uk/projects/wose/

[5] R. Buyya et al., "Economic Models for Resource Management and Scheduling in Grid Computing," *Concurrency and Computation*, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[6] L. Zeng et al., "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, May 2004.

[7] I. Brandic and S. Benkner and G. Engelbrecht and R. Schmidt, "QoS Support for Time-Critical Grid Workflow Applications," Melbourne, Australia, 2005.

[8] Lican Huang and David W. Walker and Yan Huang and Omer F. Rana, "Dynamic Web Service Selection for Workflow Optimisation ," in *UK e-Science All Hands Meeting*, Nottingham, UK, Sept. 2005.

[9] Kleywegt and A. Shapiro and H. De-Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal of Optimization*, pp. 479–502, 2001.

[10] T. Homem-de-Mello, "Monte Carlo methods for discrete stochastic optimization," *Stochastic Optimization: Algorithms and Applications*, pp. 95–117, 2000.

[11] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 1972.

[12] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, 1949.

[13] "ILOG." [Online]. Available: http://www.ilog.com/

[14] "SimJava." [Online]. Available: http://www.dcs.ed.ac.uk/home/hase