RC22505 (W0207-011) July 2, 2002
Computer Science

IBM Research Report

OptFlow - Optimal Service Selection for Workflow Execution

Jayant R. Kalagnanam, Liangzhao Zeng
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

OptFlow– Optimal Service Selection for Workflow Execution

Liangzhao Zeng , Jayant Kalagnanam

IBM T.J. Watson Research Center, USA

zlzhao@cse.unsw.edu.au, jayant@us.ibm.com

Abstract. With the proliferation of the Internet and the wide acceptance of e-commerce, increasing numbers of business processes and services are offered by distributed and heterogeneous service providers. This has created the need to explicitly employ workflow management systems (WFMS) to compose, coordinate and control the flows of services. Since each service offering differs from another along multiple QoS attributed such cost, process time, and reliability, the selection of these services to execute workflow poses a challenge. Currently, most of service selection tools regard tasks in a workflow are independent, without considering constraints (e.g. precedence constraints) among them. In this project, we proposal *OptFlow*, a framework that optimizes the service selection for entire workflow instance execution. Based on user configurable linear models, it considers constraints among the tasks and selects the optimal combination of services to execute the workflow. We adopt linear programming as a means to select services. It can guarantee that the selection result is optimal for current available services and computation cost is polynomial time.

Keywords: Optimal Web service composition, QoS, linear programming, static versus dynamic workflows.

Contact Author: Liangzhao Zeng

E-mail: zlzhao@cse.unsw.edu.au

1 Introduction

The automation of Web services composition using workflow technologies gaining a considerable momentum as a paradigm for Business-to-Business (B2B) collaboration. Leading organizations are embracing Web services today, realizing the enormous competitive advantages since this model provides faster time to market reducing cycle times and increasing customer service. Widely available and standardized Web services make it possible to realize just-in-time Business-to-Business Integration (B2Bi) which focuses on integrating Web services into a composite web service to create added value. For example, a composite web service can provide a set of high level financial management services which uses payroll, tax preparation, and cash management service as components. Using WFMSs, virtual enterprises might be formed by integrating Web services from various service providers to manufacture and sell the product.

By Web service(also called *e-service*) [1], we mean a semantically well defined abstraction that allows user to access functionalities offered by Web applications (i.e. service providers). Each service is provided by multiple providers and differ from another along multiple QoS attributes such as cost, processing time, and reliability. The selection of these services to execute workflow poses a challenge. Currently, most of the tools focus on selecting service for single task, without considering the constraints among the tasks in a workflow instance. Such solutions are easy to implement, but they cannot guarantee that the execution of whole workflow instance is optimal. In fact, an important aspect of executing a workflow instance is the selection of service providers that optimizes the utility measure (such as time to product introduction in market) of the whole workflow instance, subject to considerations such as precedence constraints, processing time, budget, and reliability.

In this project, we outline a suite of decision models and mathematical programming based solution approaches for optimal composition of services to execute workflow instances. We adopt linear programming as a means to select services. It can guarantee that the selection result is optimal for current available services and computation cost is polynomial time [3]. We have implemented this decision support solution within a Java platform that provides a standard API to work with any workflow related platform. Experimental results show that in static environments, *OptFlow* provides optimal execution plan while the computation cost is similar to one-by-one approach. For dynamic environments, *OptFlow* provides better execution plans compared to one-by-one approach in most cases, while the computation cost is still small enough to allow real-time replanning.

[1] In the remainder, we will use the terms e-service, Web service and service interchangeably

The remainder of this paper is organized as follows. Section 2 presents background about workflow management systems. Workflow modeling and web service quality are discussed. Section 3 provides a formal setting for workflow execution planning and the service provider selection problem. Section 4 presents the details about *OptFlow* - a linear programming based decision making solution for service selection. Section 5 gives details on implementation and some experimentation results. Section 6 discussion how to integrate *OptFlow* with WFMSs. Finally, section 7 gives a brief overview of related work and provides some concluding remarks.

2 Fundamental Concepts in Workflow Management Systems

This section provides the basic taxonomy and concepts that are used in workflow management systems. We first provide an overview of workflow modeling and then introduce a set of Quality-of-Service (QoS) measures relevant for workflow management and optimal Web services composition.

2.1 Workflow Model

In *OptFlow*, statechart diagram is adopted to define workflow schemas. Statechart is based upon finite automata and Event-Condition-Action (ECA). The statechart formalism has be integrated into the Unified Modeling Language (UML)[8], as the foundation of many intra and inter-object process modeling constructs. A workflow can be graphically defined using a statechart diagram. A statechart is made up of states connected by transitions. Transitions are labelled by ECA rules. There are two kinds of states in statechart: actual states and pseudo states. An actual state represents a task that involves a Web service to execute it, and is labelled by the task's name. Pseudo states correspond to two specific states: initial state and final state. They do not involve any task execution. They contribute to control structures in workflow schemes. Initial state indicates the starting of a workflow while final state indicates the completion of a workflow. The vertical bars in statechart diagram represent pseudo states that also contribute to control structure. It is noted that statechart diagram is a task oriented workflow model, it can be map to other workflow model such as WPDL from Workflow Management Coalition (WfMC) [9], WSFL[2] from IBM.

Figure 1 provides an example workflow schema that defined by a statechart, named `Create a New Part`. There are five tasks in this workflow. After task of `Make a New Part`, the task `Cost Analysis` is performed parallel to `Clash Testing` and `Clash Analysis`. When both of these threads are completed, a task of `Verify Testing` is performed. Notice that an important aspect of modeling workflow is to incorporate precedence constraints among the various tasks. These constraints specify when some tasks need to be completed before other tasks can begin or whether they can be performed in parallel.

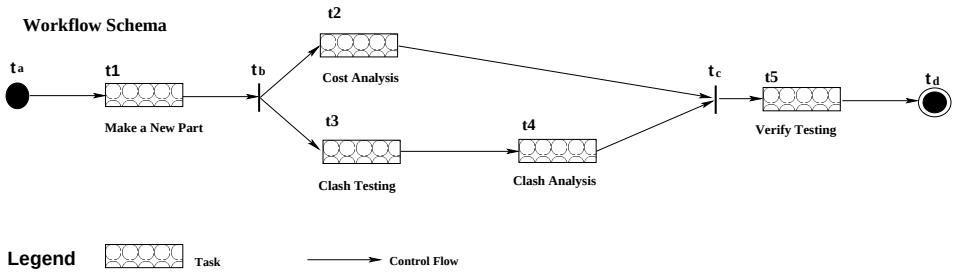

Fig. 1. An Example of Workflow

2.2 Web Service Quality

Currently, there are no established standards to measure the quality of Web services. We have identified a set of quality criteria [10] (see Table 1) to evaluate QoS of Web services. Notice that the criteria given here is generic criteria, which can be used to evaluate all services. In different application domain, some domain specific criteria may also be used to evaluate the QoS.

Table 1. Web Service Quality Criteria

Criteria	**Expression**	**Brief Explaination**
Execution Cost	q_{cost}	Monetary cost of handling a service request in US Dollars.
Execution Duration	q_{du}	Process time, measured in seconds.
Reputation	$q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$	R_i $(R_i \in [0, 10])$ is end user's ranking on a Web service's reputation, n is the times of the Web service being graded.
Reliability	$q_{rel} = \frac{N_s}{N_r}$	N_s is the total times of successful response the requests; N_r is total times of receiving requests.
Availability	$q_{av} = \frac{T_r}{T_r + T_d}$	T_r is a Web service's available time in hour after the Web service being created; T_d is the web service's unavailable time in hour after the Web service being created.

3 Planning Workflow Execution

In this section, we give a detail scenario on planning workflow execution. The assumption for planning workflow execution within the context of workflow instance is as follows:

- A workflow contains a set of takes. $WF = \{ t_1, t_2, ...t_N \}$ [2].

[2] A list of notations using in this paper is given in appendix A

- We assume that for each task t_j there is a set of potential services S_j that are available to which task t_j can be assigned.
- Each service s_{ij} provides an offer o_{ij} for task t_j. Associated with each offer o_{ij} is a quality vector $\mathcal{Q}$:

$$\mathcal{Q}(s_{ij}) = (q_{cost}(s_{ij}), q_{du}(s_{ij}), q_{rep}(s_{ij}), q_{rel}(s_{ij}), q_{av}(s_{ij})) = (q_1\ q_2, ..., q_5) \quad (1)$$

Definition 1 (*Execution Plan*).
$p = \{< t_1, s_{i1}, \mathcal{Q}_1 >, < t_2, s_{i2}, \mathcal{Q}_2 >, ..., < t_N, s_{iN}, \mathcal{Q}_N >\}$ is an **execution plan** of the workflow schema WF if:

- t_1, t_2, ... t_N are all the tasks in workflow schema.
- For each 3-tuple $< t_j, s_{ij}, \mathcal{Q}_i >$ in p, web service s_{ij} execute task t_j with service quality of $\mathcal{Q}_i$, where $\mathcal{Q}_i$ is a service quality vector.

□

In fact, an execution plan indicates which Web services are selected to execute the workflow instance. Based on each offer's quality vector and objective functions in table 2, the workflow execution plan's quality vector can be given:

$$Q(p) = (Q_{cost}(p), Q_{du}(p), Q_{rep}(p), Q_{rel}(p), Q_{av}(p)) \quad (2)$$

Table 2. Objective functions for workflow execution plan's quality

Criteria	**Objective function**
Execution Cost	$Q_{cost}(p) = \sum_{i=1}^{N} q_{cost}(t_i)$
Execution Duration	$Q_{du}(p) = CPT(q_{du}(t_1), q_{du}(t_2), ..., q_{du}(t_N))$
Reputation	$Q_{rep}(p) = \sum_{i=1}^{N} q_{rep}(t_i)$
Reliability	$Q_{rel}(p) = \Pi_{i=1}^{N}(q_{rel}(t_i) * z_i)$
Availability	$Q_{av}(p) = \Pi_{i=1}^{N}(q_{av}(t_i) * z_i)$

Note that t_j in table 2 is the task in workflow. The function CPT uses the Critical Path algorithm[3] to calculate the execution plan's total execution duration. The z_j is 0/1 variable: value 1 indicates the

[3] The Critical Path algorithm [6] is a graph algorithm which is used in project scheduling application. The description of this algorithm is outside the scope of this paper.

task t_j is a critical task that contributes to makespan; value 0 indicates the task t_j is not a critical task.

Assuming based on available offers, we can create M independent workflow execution plans. Applying above objective functions on each execution plan, we arrive at the following matrix $\mathbf{Q}$. Each row in $\mathbf{Q}$ represents a workflow execution plan, and the index for execution plan is k, the index for criteria is l.

$$\mathbf{Q} = \begin{pmatrix} Q_{1,1} & Q_{1,2} & \dots & Q_{1,5} \\ Q_{2,1} & Q_{2,2} & \dots & Q_{2,5} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{M,1} & Q_{M,2} & \dots & Q_{M,5} \end{pmatrix}$$

In *OptFlow*, we adopt the Simple Additive Weighting method (SAW) to select workflow execution plan. The SAW method has two phases:

1. **Scaling Phase**

   Among above criteria, some of them are negative criteria i.e. the higher of the score, the worse of the quality, such as execution time and execution cost. The rest of them are positive criteria i.e. the higher of the score, the better of the quality. For negative criteria, scores are scaled according to Equation 3. For positive criteria, scores are scaled according to Equation 4. This normalize scores to be in [0, 1] for each criterion.

$$v_{k,l} = \begin{cases} \frac{Q_l^{max} - Q_{k,l}}{Q_l^{max} - Q_l^{min}} & \text{if } Q_l^{max} - Q_l^{min} \neq 0 \\ 1 & \text{if } Q_l^{max} - Q_l^{min} = 0 \end{cases} \tag{3}$$

$$v_{k,l} = \begin{cases} \frac{Q_{k,l} - Q_l^{min}}{Q_l^{max} - Q_l^{min}} & \text{if } Q_l^{max} - Q_l^{min} \neq 0 \\ 1 & \text{if } Q_l^{max} - Q_l^{min} = 0 \end{cases} \tag{4}$$

   where $Q_l^{max} = Max(Q_{k,l}), Q_l^{min} = Min(Q_{k,l}), 1 \leq k \leq M$. In section 4.1, we show that the computation of Q_l^{max} and Q_l^{min} is straight forward and does not require enumerating all the plans. Applying these two equations on $\mathbf{Q}$, we can have $\mathbf{Q}^{'}$ as follows:

$$\mathbf{Q}^{'} = \begin{pmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,5} \\ v_{2,1} & v_{2,2} & \dots & v_{2,5} \\ \vdots & \vdots & \vdots & \vdots \\ v_{M,1} & v_{M,2} & \dots & v_{M,5} \end{pmatrix}$$

2. **Weighting Phase**

   Following formula will be used to compute overall quality score for each web service:

$$V(p_k) = \sum_{l=1}^{5} (v_{k,l} * W_l), W_l \in [0,1] \tag{5}$$

   where W_j represents the weight of each criteria. In (5), workflow designers can balance different criteria to select a desired workflow execution plan by adjusting the value of W_j. The system will choose the workflow execution plan that has the maximal value of $V(p)$. If there are more than one workflow execution plan which have same maximal value of $V(p)$, then a workflow execution plan will be selected randomly.

4 Linear programming for Service Selection

Section 3 suggests an approach to select the workflow execution plan by generating all possible workflow execution plans. Assuming there are N tasks in workflow; there are M offers for each task. So, the total workflow execution plans is M^N. The selection computation cost is $O(M^N)$ time. Such approach is impractical for most of workflow instances. In this section, we present an approach based on a linear programming formulation [3], which can select optimal workflow execution plans in polynomial time.

4.1 Compute Q_l^{max} and Q_l^{min}

As we discussed in section 3, we need to compute Q_l^{max} and Q_l^{min} for each criterion. In fact we can compute Q_l^{max} and Q_l^{min} without generating all possible execution plans. For example, in order to compute the maximum execution cost of all the workflow execution plans, we select the most expensive offer for each task and sum up all the execution cost in these offers, we can get Q_{cost}^{max}. For another example, in order to compute the minimum execution duration of all the workflow execution plans, we select the shortest execution time offer for each task and use CPT function to compute minimum

execution duration (i.e. Q_{du}^{min}). It is noted that computation cost of Q_l^{max} and Q_l^{min} is polynomial time. Using 3 and 4, 5 can be rewritten as:

$$V(p_k) = \sum_{l=1}^{5}(v_{k,l} * W_l) = \sum_{l=1}^{2}\left(\frac{Q_l^{max} - Q_{k,l}}{Q_l^{max} - Q_l^{min}} * W_l\right) + \sum_{l=3}^{5}\left(\frac{Q_{k,l} - Q_l^{min}}{Q_l^{max} - Q_l^{min}} * W_l\right), W_l \in [0,1] \quad (6)$$

In the following subsection, we present different decision models that allow for different objective functions and constraints.

4.2 Makespan and Cost Considerations

In this section, we first simplify the problem by only considering makespan and cost into the decision process. Here the objective function is a weighted sum of the makespan and costs. Here we assume the makespan and costs have been normalized to a comparable scale using 3 and 4.

$$Max\left(\frac{Q_{cost}^{max} - Q_{cost}}{Q_{cost}^{max} - Q_{cost}^{min}} * W_1 + \frac{Q_{du}^{max} - Q_{du}}{Q_{du}^{max} - Q_{du}^{min}} * W_2\right) \quad (7)$$

Assuming A is the set of all tasks to be executed and the index for tasks is j. For each task j, there is a set of services S_j, and the index for services is i; y_{ij} (i.e. 0/1 variable) indicates whether service i is chosen for task j. For each task, we only selected one service:

$$\sum_{i \in S_j} y_{ij} = 1, \ \forall j \in A \quad (8)$$

Assuming x_j is the earliest start time of task j, p_j is the execution time for task j, p_{ij} is execution time for task j by service i and x_k is task j's next task (i.e. $j \to k$), we have constraints as follows:

$$\sum_{i \in S_j} p_{ij}\ y_{ij} = p_j, \forall j \in A \quad (9)$$

$$x_k - (p_j + x_j) \geq 0, \forall j \to k, j, k \in A \quad (10)$$

$$Q_{du} - (x_j + p_j) \geq 0, \forall j \in A \quad (11)$$

For execution cost, c_{ij} is execution cost for task j by service i, we have a constraint as follows:

$$Q_{cost} = \sum_{j \in A}\sum_{j \in S_j} c_{ij}\ y_{ij} \quad (12)$$

After re-arranging, we have a $MakeSpanCost$ decision model as follows:

$$\begin{array}{lll}
\text{Max} & \left(\frac{Q_{cost}^{max}-Q_{cost}}{Q_{cost}^{max}-Q_{cost}^{min}} * W_1 + \frac{Q_{du}^{max}-Q_{du}}{Q_{du}^{max}-Q_{du}^{min}} * W_2\right) & MakespanCost \\
\text{subject to} & & \\
 & \sum_{i \in S_j} y_{ij} & = 1, \quad \forall j \in A \\
 & \sum_{i \in S_j} p_{ij}\, y_{ij} & = p_j, \quad \forall j \in A \\
 & x_k - (p_j + x_j) & \geq 0, \quad \forall j \rightarrow k, j, k \in A \\
 & Q_{du} - (x_j + p_j) & \geq 0, \quad \forall j \in A \\
 & x_j & \geq 0 \quad \forall j \in A \\
 & y_j & \in 0, 1 \quad \forall j \in A \\
 & \sum_{j \in A} \sum_{j \in S_j} c_{ij}\, y_{ij} & = Q_{cost}
\end{array}$$

Notice that this formulation requires 0/1 integer variables and suggests the need for a IP Solver. However, it can be shown that the LP relaxations provide integer solutions.[4] An alternative is to examine the makespan that can be achieved within a given budget constraint B. The decision model is as follows:

$$\begin{array}{lll}
\text{Min} & (Q_{du}) & Budget - Constraint \\
\text{subject to} & & \\
 & \sum_{i \in S_j} y_{ij} & = 1, \quad \forall j \in A \\
 & \sum_{i \in S_j} p_{ij}\, y_{ij} & = p_j, \quad \forall j \in A \\
 & x_k - (p_j + x_j) & \geq 0, \quad \forall j \rightarrow k, j, k \in A \\
 & Q_{du} - (x_j + p_j) & \geq 0, \quad \forall j \in A \\
 & \sum_{j \in A} \sum_{j \in S_j} c_{ij}\, y_{ij} & \leq B \\
 & x_j & \geq 0 \quad \forall j \in A \\
 & y_j & \in 0, 1 \quad \forall j \in A
\end{array}$$

By introducing a budget constraint the above problem can no longer be solved as an LP but needs to be explicitly solved as an IP. This problem is a special case of the knapsack problem and hence NP-hard [4]. Notice that other criteria can be easily incorporated into this model by adding criteria into the objective function if the objective function is a linear function. For example, assuming r_{ij} is service reputation for task j by service i, we have a linear model that includes the consideration of

[4] It can be shown that any fractional solution can be improved by reassigning the task to one of the services and hence the LP provides integer solution.

service reputation as follows:

$$Max\left(\frac{Q_{cost}^{max}-Q_{cost}}{Q_{cost}^{max}-Q_{cost}^{min}}*W_1+\frac{Q_{du}^{max}-Q_{du}}{Q_{du}^{max}-Q_{du}^{min}}*W_2+\frac{Q_{rep}-Q_{rep}^{min}}{Q_{rep}^{max}-Q_{rep}^{min}}*W_3\right) \quad (13)$$
$$where \;\; Q_{rep}=\sum_{j\in A}\sum_{j\in S_j} r_{ij}y_{ij}$$

It is noted that the constraints are the same as $MakespanCost$ model.

4.3 Reliability and Availability

In this section, we consider criteria that the objective function is not a linear function. Among criteria that are used to select services, the availability and reliability's objective function is nonlinear (See table 2). We can linearize them using log function as shown below. Assuming a_{ij} is reliability for task j by service i, z_{ij} is a 0/1 variable that indicates for task j whether service i contributed to the makespan, we have the object function as follows:

$$Max\left(\frac{Q_{cost}^{max}-Q_{cost}}{Q_{cost}^{max}-Q_{cost}^{min}}*W_1+\frac{Q_{du}^{max}-Q_{du}}{Q_{du}^{max}-Q_{du}^{min}}*W_2+\frac{Q_{rep}-Q_{rep}^{min}}{Q_{rep}^{max}-Q_{rep}^{min}}*W_3+\frac{Q_{rel}-Q_{rel}^{min}}{Q_{rel}^{max}-Q_{rel}^{min}}*W_4\right)$$

As we know, the reliability of workflow execution plan is

$$Q_{rel}=\Pi_{j\in A}\left(\sum_{j\in S_j} a_{ij}z_{ij}\right)$$

By applying the function ln, we arrive:

$$ln(Q_{rel})=\sum_{j\in A} ln\left(\sum_{j\in S_j} a_{ij}z_{ij}\right)$$

Since $\sum_{j\in A} z_{ij}=1$ and $z_{ij}=0/1$, we can write this as

$$ln(Q_{rel})=\sum_{j\in A}\left(\sum_{j\in S_j} ln(a_{ij})z_{ij}\right)$$

Let $Q'_{rel} = ln(Q_{rel})$ and $a'_{ij} = ln(a_{ij})$, we have

$$Q'_{rel} = \sum_{j \in A} \left(\sum_{j \in S_j} a'_{ij} z_{ij} \right) \tag{14}$$

Using Q'_{rel} to replace Q_{rel} and a'_{ij} to replace a_{ij}, we have a linear model as follows:

$$Max \left(\frac{Q^{max}_{cost} - Q_{cost}}{Q^{max}_{cost} - Q^{min}_{cost}} * W_1 + \frac{Q^{max}_{du} - Q_{du}}{Q^{max}_{du} - Q^{min}_{du}} * W_2 + \frac{Q_{rep} - Q^{min}_{rep}}{Q^{max}_{rep} - Q^{min}_{rep}} * W_3 + \frac{Q'_{rel} - Q^{min}_{rel}}{Q^{max}_{rel} - Q^{min}_{rel}} * W_4 \right)$$

subject to

$$\begin{array}{lll}
\sum_{i \in S_j} y_{ij} & = 1, & \forall j \in A \\
\sum_{i \in S_j} p_{ij}\, y_{ij} & = p_j, & \forall j \in A \\
x_k - (p_j + x_j) & \geq 0, & \forall j \to k, j, k \in A \\
Q_{du} - (x_j + p_j) & \geq 0, & \forall j \in A \\
x_j & \geq 0 & \forall j \in A \\
y_j & \in 0, 1 & \forall j \in A \\
\sum_{j \in A} \sum_{i \in S_j} p_{ij} z_{ij} & = Q_{du} & \\
\sum_{j \in A} \sum_{j \in S_j} r_{ij} y_{ij} & = Q_{rep} & \\
\sum_{j \in A} \sum_{j \in S_j} c_{ij}\, y_{ij} & = Q_{cost} & \\
z_{ij} & \leq y_{ij} & \forall j \in A, i \in S_j \\
\sum_{j \in A} \sum_{i \in S_j} a'_{ij} z_{ij} & = Q'_{rel} &
\end{array}$$

Assuming b_{ij} is availability for task j by service i, we have $TotalConstraints$ model that considers all the criteria listed in table 1 as follows:

$$Max(\frac{Q^{max}_{cost} - Q_{cost}}{Q^{max}_{cost} - Q^{min}_{cost}} * W_1 + \frac{Q^{max}_{du} - Q_{du}}{Q^{max}_{du} - Q^{min}_{du}} * W_2 + \frac{Q_{rep} - Q^{min}_{rep}}{Q^{max}_{rep} - Q^{min}_{rep}} * W_3 + \frac{Q'_{rel} - Q^{min}_{rel}}{Q^{max}_{rel} - Q^{min}_{rel}} * W_4 + \frac{Q'_{av} - Q^{min}_{av}}{Q^{max}_{av} - Q^{min}_{av}} * W_5) \tag{15}$$

subject to

$$
\begin{array}{lll}
\sum_{i \in S_j} y_{ij} & = 1, & \forall j \in A \\
\sum_{i \in S_j} p_{ij}\, y_{ij} & = p_j, & \forall j \in A \\
x_k - (p_j + x_j) & \geq 0, & \forall j \rightarrow k, j, k \in A \\
Q_{du} - (x_j + p_j) & \geq 0, & \forall j \in A \\
x_j & \geq 0 & \forall j \in A \\
y_j & \in 0, 1 & \forall j \in A \\
\sum_{j \in A} \sum_{i \in S_j} p_{ij} z_{ij} & = Q_{du} & \\
\sum_{j \in A} \sum_{j \in S_j} r_{ij} y_{ij} & = Q_{rep} & \\
\sum_{j \in A} \sum_{j \in S_j} c_{ij}\, y_{ij} & = Q_{cost} & \\
z_{ij} & \leq y_{ij} & \forall j \in A, i \in S_j \\
\sum_{j \in A} \sum_{i \in S_j} a'_{ij} z_{ij} & = Q'_{rel} & \\
\sum_{j \in A} \sum_{i \in S_j} b'_{ij} z_{ij} & = Q'_{av} &
\end{array}
$$

Where $Q'_{av} = ln(Q'_{av}) = \sum_{j \in A} \left(\sum_{j \in S_j} b'_{ij} z_{ij} \right)$ and $b'_{ij} = ln(b_{ij})$.

It is noted that criteria that can be added into our decision model are not limited in Table 1. Other domain specific criteria can be also easily added into our decision model once the objective functions are given.

4.4 Uncertainty Considerations

The models presented in the previous sections have assumed that the processing time reported by services are completely deterministic. In fact, the processing time p_{ij} might be uncertain. Assuming p_{ij} is normal distribution. The normal distribution has a probability function given as follows:

$$f_X(s) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{1}{2} \left(\frac{x - \mu}{\sigma} \right)^2 \right], -\infty < x < \infty \tag{16}$$

It is a 2 parameter distribution with a mean (i.e. μ_{ij}) and standard deviation (i.e. σ), where

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{17}$$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{18}$$

So, the σ_{ij} for each service's p_{ij} can be computed based on its pass execution results. Since $\sum_{i\in A}\sum_{i\in S_j} p_{ij}z_{ij} = Q_{du}$, the makespan Q_{du} must be normal distribution and its deviation σ_{du} is

$$\sigma_{du}^2 = \sum_{i\in A}\sum_{i\in S_j}\sigma_{ij}^2 z_{ij} \tag{19}$$

In the following $TotalUncertainty$ model, we consider the makespan's deviation and given a deviation constraint α.

$$Max(\frac{Q_{cost}^{max} - Q_{cost}}{Q_{cost}^{max} - Q_{cost}^{min}} * W_1 + \frac{Q_{du}^{max} - Q_{du}}{Q_{du}^{max} - Q_{du}^{min}} * W_2 + \frac{Q_{rep} - Q_{rep}^{min}}{Q_{rep}^{max} - Q_{rep}^{min}} * W_3 + \frac{Q_{rel}^{'} - Q_{rel}^{min}}{Q_{rel}^{max} - Q_{rel}^{min}} * W_4 + \frac{Q_{av}^{'} - Q_{av}^{min}}{Q_{av}^{max} - Q_{av}^{min}} * W_5) \tag{20}$$

subject to

$$\begin{array}{lll}
\sum_{i\in S_j} y_{ij} & = 1, & \forall j \in A \\
\sum_{i\in S_j} p_{ij}\ y_{ij} & = p_j, & \forall j \in A \\
x_k - (p_j + x_j) & \geq 0, & \forall j \to k, j, k \in A \\
Q_{du} - (x_j + p_j) & \geq 0, & \forall j \in A \\
x_j & \geq 0 & \forall j \in A \\
y_j & \in 0, 1 & \forall j \in A \\
\sum_{j\in A}\sum_{i\in S_j} p_{ij}z_{ij} & = Q_{du} & \\
\sum_{j\in A}\sum_{j\in S_j} r_{ij}y_{ij} & = Q_{rep} & \\
\sum_{j\in A}\sum_{j\in S_j} c_{ij}\ y_{ij} & = Q_{cost} & \\
z_{ij} & \leq y_{ij} & \forall j \in A, i \in S_j \\
\sum_{j\in A}\sum_{i\in S_j} a_{ij}^{'} z_{ij} & = Q_{rel}^{'} & \\
\sum_{j\in A}\sum_{i\in S_j} b_{ij}^{'} z_{ij} & = Q_{av}^{'} & \\
\sum_{i\in A}\sum_{i\in S_j} \sigma_{ij}^2 z_{ij} & \leq \alpha &
\end{array}$$

5 Implementation of *OptFlow* and Performance Evaluation

In this section, we first present the Java API provided by *OptFlow*, then provide some experimental results.

5.1 *OptFlow*'s Java API

OptFlow has been implemented as a Java Class based on IBM's OSL, an optimization package. For each decision model, a java method is provided. Detail description on the Java Class is given as follows:-

- **Field Detail**
  - `public float[ ]` **weights**
    The weight of each criterion.
  - `public` WorkflowInstance **workflowInstance**
    The class object representing the workflow instance, which contains control flows and service offers for each task in workflow instance.
  - `public` WorkflowExecutionPlan **workflowExecutionPlan**
    The class object representing the workflow execution plan, i.e. which services are selected to execute the workflow instance.
  - `public float` **budget**
    The budget constraints for workflow execution plan.
  - `public float` **deviation**
    The deviation constraints for workflow execution plan's makespan.
- **Constructor Detail**
  - OptFlow(`float[ ]` **weights**, WorkflowInstance **workflowInstance**)
    Constructs a OptFlow object that is initialized with weight of each Web service criterion and WorkflowInstance.
  - OptFlow(`float[ ]` **weights**, WorkflowInstance **workflowInstance** , `float` **budget**)
    Constricts a OptFlow object that is initialized with weight of each Web service criterion, WorkflowInstance and budget constraint on workflow execution plan.
  - OptFlow(`float[ ]` **weights**, WorkflowInstance **workflowInstance** , `float` **deviation**)
    Constricts a OptFlow object that is initialized with weight of each Web service criterion, WorkflowInstance and deviation constraint on workflow execution plan's makespan.
- **Method Detail**

- `public` WorkflowExecutionPlan MakespanCostSolver()
  Returns a workflow execution plan based on *MakespanCost* decision model.
- `public` WorkflowExecutionPlan BudgetConstraintSolver()
  Returns a workflow execution plan based on *BudgetConstraint* decision model.
- `public` WorkflowExecutionPlan TotalConstraintsSolver()
  Returns a workflow execution plan based on *TotalConstraints* decision mode.
- `public` WorkflowExecutionPlan MakespanUncertaintySolver()
  Returns a workflow execution plan based on *MakespanUncertainty* decision mode.

5.2 *OptFlow* Performance Evaluation

We conducted experiments using the implemented Java Class to evaluate the solution. The Java Class run on a PC with configuration of Pentium III 700 MHz and 512M RAM. The software platform is Windows 2000, Java 2 Standard Edition V1.3.0 and IBM OSL Version 3. We use the Simplex algorithm for solving LPs, which can be exponential. In practice, the performance of Simplex method is very efficient. Actually one can use the interior-point method which is polynomial.

We vary the configuration parameters such as size of takes (ST), number of offers per task (NO) and number of branches (NB) of workflow instances to evaluate the computation cost (CC, in second) and planning results. It is noted that number of branches of workflow presents how many concurrent execution flows in a workflow. For example, The workflow given in figure 1's branch number is two. This subsection presents two experimental results. The first experiment is conducted in static environments, i.e. during the workflow execution, quality of each task's execution results are the same as offers and the offers do not change during the execution of workflow instances. The second experiment is conducted in a dynamic environments, i.e. during workflow execution, quality of each task's execution results may different from the offers, existing offers may become invalid, new offers may be available, etc.

5.2.1 Experiments in Static Environments

In static environments, once a workflow execution plan is created, we do not need to replan the workflow execution. So, the planning method is involved only once for a workflow instance. In order to compare with other non-optimal selection approaches, we also use *OptFlow* to select services for each task in workflow instance one by one , without considering constraints among the tasks. So, for a workflow instance, assuming the size of tasks is N, then a planning method in *OptFlow* needs to be involved N times, each time selects a service. In the following tables 3 and 4, we give details experimental results on `MakespanBudgetSolver()` and `TotalConstraintsSolver()`.

Table 3. `MakespanCostSolver()` (MCS) *Vs* One-By-One (OBO) in Static Environments

Workflow Instance	ST	NO	NB	CC		$Q_{du}(p)$		$Q_{cost}(p)$	
				MCS	OBO	MCS	OBO	MCS	OBO
1	5	3	2	0.0141	0.015	90.0	100.0	6.0	7.0
2	10	3	2	0.0172	0.030	280.0	280.0	14.0	16.0
3	10	10	3	0.0251	0.055	230.0	270.0	13.5	15.5
4	25	15	3	0.0832	0.082	420.5	440.5	80.5	89.5
5	40	15	5	0.1481	0.141	1500.5	1620.5	120.5	140.5
6	40	20	5	0.1832	0.148	1420.2	1550.3	112.5	123.5

Table 4. Experiments of `TotalConstraintsSolver()` (TCS) and One-by-One (OBO) in Staitc Environments

Workflow Instance	ST	NO	NB	CC		$Q_{du}(p)$		$Q_{cost}(p)$		$Q_{rep}(p)$		$Q_{rel}(p)$		$Q_{av}(p)$	
				TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO
1	5	3	2	0.017	0.016	90.4	110.2	6.3	6.8	49	47	0.53	0.52	0.84	0.82
2	10	3	2	0.023	0.032	300.5	330.5	27.4	28.4	75	70	0.79	0.75	0.45	0.42
3	10	10	3	0.040	0.059	260.5	280.5	24.3	25.2	78	75	0.81	0.78	0.48	0.43
4	25	15	3	0.112	0.085	450.5	470.5	85.5	88.3	220	210	0.21	0.18	0.085	0.072
5	40	15	5	0.151	0.152	1930.4	1990.4	160.3	165.3	381	361	0.015	0.11	0.0161	0.0152
6	40	20	5	0.238	0.158	1890.3	1920.4	152	159.2	392	382	0.0163	0.0152	0.0189	0.0158

Tables 3 and 4 show that in static environments, *OptFlow* creates better workflow execution plans than non-optimal approach. At the same time, the computation cost is very limited.

5.2.2 Experiments in Dynamic Environments

We also conduct experiments in dynamic environments. In dynamic environments, execution of workflow instances need to be replanned whenever a task is completed. Assuming the number of tasks in workflow instance is N, in order to optimize the service selection, planning methods in *OptFlow* need to be involved N times during the execution of workflow instance. For non-optimal approach, it select services the same as in static environment, not any add-on overhead required. In the following table 5 and 6, we give details experimental results on `MakespanBudgetSolver()` and `TotalConstraintsSolver()`. It is noted that the number of offers in the table is in a given range. It changes from time to time.

Table 5 and 6 show that *OptFlow* creates better workflow execution results than that of non-optimal approach in most of the cases. The computation cost is still acceptable. Of course, in some case (see table 5), *OptFlow* may create worse workflow execution result. The reason is that *OptFlow* make the decision based on current available services. Some expected service may become unavailable

Table 5. `MakespanCostSolver()` (MCS) *Vs* One-By-One (OBO) in Dynamic Environments

Workflow Instance	ST	NO	NB	CC		$Q_{du}(p)$		$Q_{cost}(p)$	
				MCS	OBO	MCS	OBO	MCS	OBO
1	5	3-5	2	0.062	0.017	90.1	100.1	6.2	7.1
2	10	3-5	2	0.162	0.035	280.2	280.4	13.5	14.2
3	10	10-15	3	0.241	0.065	240.0	**230.2**	13.5	**13.2**
4	25	15-20	3	1.432	0.083	440.6	430.1	82.5	89.5
5	40	15-20	5	3.214	0.151	1501.5	1504.2	122.5	125.5
6	40	20-25	5	3.8832	0.178	1402.2	1460.3	112.5	115.5

Table 6. Experiments of `TotalConstraintsSolver()` (TCS) and One-by-One (OBO) in Dynamic Environments

Workflow Instance	ST	NO	NB	CC		$Q_{du}(p)$		$Q_{cost}(p)$		$Q_{rep}(p)$		$Q_{rel}(p)$		$Q_{av}(p)$	
				TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO	TCS	OBO
1	5	3-5	2	0.08	0.016	90.70	100.2	6.3	6.4	46	**48**	0.51	**0.52**	0.83	0.83
2	10	3-5	2	0.21	0.032	301.5	302.5	26.6	27.2	72	71	0.76	0.76	0.42	0.42
3	10	10-15	3	0.35	0.059	270.5	270.8	25.3	2.52	76	75	0.79	0.78	0.49	0.45
4	25	15-20	3	1.672	0.087	406.5	406.8	86.5	87.3	210	203	0.201	0.189	0.084	0.079
5	40	15-20	5	3.351	0.155	1990.4	2000.4	163.3	166.3	372	351	0.0149	0.11	0.0172	0.0161
6	40	20-25	5	5.438	0.162	1900.3	1940.4	153.1	159.5	381	367	0.0153	0.0142	0.0180	0.0168

when the task need to be executed, which make the created workflow execution become non-optimal. But overall of *OptFlow*'s performance still better than that of non-optimal approach.

6 Integrate *OptFlow* with WFMSs

In this section, we use PLM_{flow} [11] as an example to illustrate how to integrate *OptFlow* with WFMSs.

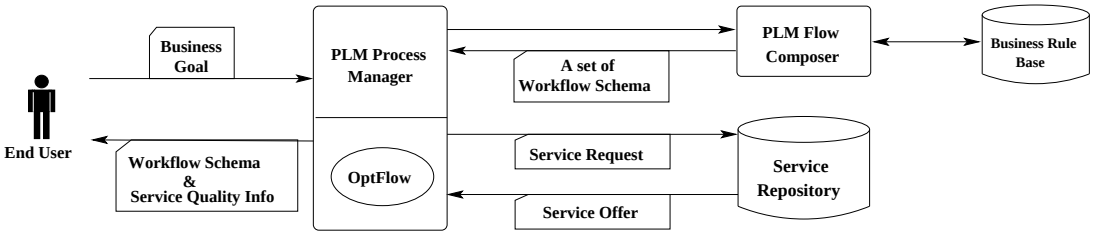

Fig. 2. PLM_{flow} System Architecture

PLM_{flow} is a workflow management system that dynamical composes and executes business process by rule reference. It is a dynamic workflow system that is capable of supporting non-deterministic

processes such as those found in collaborative product design scenarios, where decisions made by collaborative partners necessitate the dynamic composition and modification of running workflows. Instead of building complex static workflows to accommodate an explosive number of possibilities, PLM_{flow} advocates a business rule inference based system to dynamically generate and execute workflows. As a result, end users can focus on the business goals to be achieved, instead of having to create detailed control and data flows for the work at hand.

Figure 2 provide an overall picture of PLM_{flow} system architecture. PLM_{flow} consists of two basic components: PLM Process Manager and PLM Process Composer. *OptFlow* has been implemented as a component of PLM Process Manger that supports PLM Flow schema selection and PLM Flow execution. Business process composition and execution in PLM_{flow} involves three major steps (see Figure 3):

1. **PLM Flow schema composition.** Initially, when the system receives an end user's request, there is only one task (i.e. business goal) in the PLM Flow schema. In this phase, a set of PLM Flow schemas are composed using backward-chain inference and forward-chain inference.
2. **PLM Flow schema selection.** In this phase, *OptFlow* is used to create an optimized workflow execution plan for each PLM Flow schema. At the same time, it also predicate the service quality of each execution plan. Such service quality information are provided to end users to facilitate the PLM Flow schema selection.
3. **PLM Flow instance execution.** During the execution of PLM Flow instance, the execution plan that created by *OptFlow* is used to assign tasks to services to execute them. It is noted that PLM Flow structure may change based on rule inference, new service may add into service repository, or task actual execution result may different from service offer. In these case, *OptFlow* need to re-plan the PLM Flow execution.

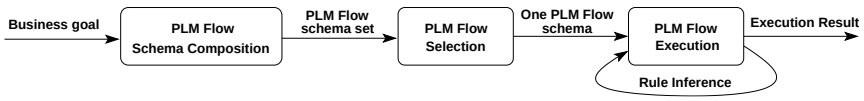

Fig. 3. Business Process Composition and Execution in PLM_{flow}

7 Related Work and Conclusion

There are many on going research efforts in service quality and service selection area. In [1] the authors introduce a market-based service selection mechanism. They focus on selecting service based

on deadline and budget constraints. They select a service for each task individually, then aggregate services to determine whether and to which extent the workflow execution plan has exceeded time or cost constraints. In [7], the authors propose a set of criteria to select service, but do not provide a detail approach on how to select services. In [5], linear programming is used to select a data source that has optimal information quality. In this project, we proposal *OptFlow*, a framework that optimizes the service selections for entire workflow instance execution. Based on user configurable linear models, it considers precedence constraints among the tasks and select service to create workflow execution plan. We adopt linear programming as a means to select services. It can guarantee the selection result is optimized and computation cost is polynomial time. We had implemented *OptFlow* and our experimental results demonstrate that *OptFlow* can be used both static and dynamic environments.

References

1. Andreas Geppert, Markus Kradolfer, and Dimitrios Tombros. Market-based workflow management. *Lecture Notes in Computer Science*, 1402, 1998.
2. Wsfl: Web service flow language, 2001. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.
3. Howard Karloff. *Linear Programming.* Birkhauser, 1991.
4. Silvano Martello and Paolo Toth. *Knapsack Problems : Algorithms and Computer Implementations.* John Wiley and Sons, 2001.
5. Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 447–458, Edinburgh, UK, 1999.
6. Michael Pinedof. *Scheduling: Theory, Algorithms, and Systems (2nd Edition).* Prentice Hall, 2001.
7. Claude Stricker, Stefano Riboni, Markus Kradolfer, and John Taylor. Market-based workflow management for supply chains of services. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 1998.
8. UML Resource page, 2000. At http://www.omg.org/uml/.
9. The Workflow Management Coalition home page, 2000. At http://www.aiim.org/wfmc/mainframe.htm.
10. Liangzhao Zeng, Boualem Benatallah, and Anne Hee Hiong Ngu. Dynamic web service integration and collaboration (submit for publication). Technical report, School of Computer Science and Engineering, University of New South Wales, 2002.
11. Liangzhao Zeng, David Flaxer, Henry Chang, and Jun-Jang Jeng. PLM_{flow}–dynamical business process composition and execution by rule inference. In *Proceedings 3rd VLDB Workshop on Technologies for E-Services (TES'02)*, HongKong, China, 2002.

Appendix A: List of Notations

A: set of all tasks to be performed

N: number of task in A, i.e. $|A|$

j: index for tasks, $j \in A$

S_j: set of services for task j

i: index for services that can execute task t_j, $j \in S_j$

s_{ij}: service i for task j, $s_ij \in S_j$

o_{ij}: offer provided by service s_{ij}

p_{ij}: processing time for task j by service i

c_{ij}: cost of using service s_{ij} for task j

y_{ij}: 0/1 variable incates if service s_{ij} is chosen for task j

B:total budget for project

x_j: earliest start time of task j

z_{ij}: 0/1 variable to indicate whether service s_{ij} contributes to critical path

a_{ij}: reliability of service ij for task j

b_{ij}: availability of service ij for task j