QoS-based Selection of Multi-Granularity Web Services for the Composition

Bo Zhou, Keting Yin, Honghong Jiang and Shuai Zhang
College of Computer Science and Technology, Zhejiang University, Hangzhou, China
Email: {bzhou, yinketing, zhejianghonghong, zhangshuai}@zju.edu.cn

Aleksander J. Kavs
State Street Corporation, Boston, MA, USA
Email: ajkavs@statestreet.com

***Abstract*—Existing methods for QoS-aware services composition only consider web services whose service class is specified in the process definition as the candidate. However, there may exist some services that could also accomplish partial goal of the web service composition but their service classes do not appear in the process definition. In this paper, we propose a new QoS-aware service composition approach, which expands the choice for services selection by allowing web services of various granularities to be available for selection. A method based on Mixed Integer Linear Programming (MILP) is proposed to solve the QoS-based Multi-Granularity Service Selection Problem (QMGSSP), which optimizes the user-defined objective and meets the end-to-end QoS constraints as well. Experiments show the effectiveness of our approach.**

***Index Terms*—web service, service composition, QoS, multi-granularity**

I. Introduction and Related Work

In Service-Oriented Computing (SOC) paradigm, WS[1] can be composed to form value-added services through the process of web service composition. Web service composition can be staged in two phases. The first phase is functionality-oriented, in which the process definition is generated. The process definition is a specification that consists of service classes aggregated by composition patterns, and the service class is an abstraction of web services, which specifies the interface and functionality. Fig. 1 shows major composition patterns for WS.

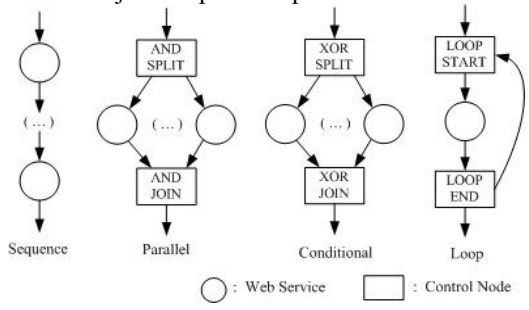

Figure 1. Web Service Composition Patterns

[1] WS is used as shorthand for "web service" or "web services" in the paper.

Since many existing WS could provide equivalent functionality, although with different QoS values, a selection should be executed in the second phase to choose a WS for each service class, which optimizes the user-defined objective and meets the end-to-end QoS constraints as well. The process is called QoS-aware services composition, which is a hot research topic. QoS-based services selection methods can be classified into local and global approaches. The former approach is done at the task level, where component WS are selected for each task individually [1-3]. Although simple and efficient, this approach cannot guarantee the end user's QoS constraints. On the other hand, the latter approach aims at achieving global optimum solution for the composition, which is NP-hard in strong sense [3]. A lot of effort has been devoted to enhance this key technology in recent years. Reference [2, 4] use linear integer programming method to dynamically find the best solution for the composition. Similarly to this approach, Reference [3, 5] extend previous work by introducing advanced techniques, such as loop peeling, negotiation and web service dependencies constraints allowing the execution of stateful WS. Reference [6] presents the VRESCo runtime that supports an end-to-end approach for QoS-aware service composition, where constraint programming and integer programming methods are used to search the best solution. Other techniques are also applied to optimize the QoS-aware services composition, such as genetic programming [7, 8], negotiation [9], workflow partition strategy [10].

However, current methods for QoS-aware services composition is lack of flexibility in that only WS whose service class is specified in the process definition will be available for selection, while there may exist other WS that could also accomplish partial goal of the web service composition but their service classes do not appear in the process definition. For example, consider a simple process definition composed by two service classes in sequence. In traditional service selection process, service classes are fixed once the process definition is given and each service class will be instantiated with its candidate WS. Assume service classes sc_1 and sc_2 are bound with WS ws_1, ws_2 respectively, meanwhile there exists another

doi:10.4304/jsw.6.3.366-373

WS ws_{12}, which could complete the task that is performed by ws_1 and ws_2 in sequence. Current service selection process will not consider ws_{12} even if its QoS is better than the aggregated QoS of ws_1 and ws_2, since the process definition does not contain a service class that can accommodate ws_{12}.

To overcome this shortcoming, our work aims at expanding the choice space for services selection to achieve better solution. We propose an approach for QoS-aware services composition, which is capable of selecting from multi-granularity web services. In our context, granularity denotes the extent to which the web service composition is broken down into small parts. In above example, ws_{12} is more large-grained compared with ws_1, ws_2. Our approach breaks through the restriction imposed by the structure of the given process definition by allowing WS of various granularities (e.g. ws_1, ws_2 and ws_{12}) to be considered for selection.

Our contributions are threefold:

- Introduce the concept of "granularity" to WS and define related terminology.
- Propose and formally define the QoS-based Multi-Granularity Service Selection Problem (QMGSSP).
- Present the MILP-based approach to solve QMGSSP.

The rest of this paper is arranged as follows. Section II briefly introduces WS QoS modeling as preliminary knowledge. In Section III, we present related concepts and the problem statement. Our approach for QMGSSP is proposed in Section IV and experiments are discussed in Section V. The final section concludes our research and discusses the future work.

II. Web Service QoS Modeling

The QoS modeling for WS includes four primary QoS dimensions but is extensible:

Response Time: the elapsed time between the moment the user requests a service and the moment that a response is returned.

Cost: the fee has to be paid to the service provider in order to fulfill a service request.

Reputation: a measure of trustworthiness of the WS, which is generally calculated based on the consumers' feedbacks, with a value range of [0, 10].

Reliability: the probability that a response will be returned within a reasonable duration after invoking the WS, with a value range of [0, 1].

We use a 4-tuple to denote the QoS values of WS:

$$QoS(ws) = [RT, Cost, Rep, Rel]$$

where RT, Cost, Rep, Rel are the values of response time, cost, reputation and reliability respectively.

QoS aggregation methods for web service composition have been researched extensively, which are generally based on workflow patterns. Due to space limitation, we refer interested readers to existing literature [11-13] about QoS aggregation rules for the composition.

III. Definitions and Problem Statement

In this section, we formally define relevant concepts and propose the QoS-based Multi-Granularity Service Selection Problem (QMGSSP) at the end. The comprehensive example of process definition in Fig. 2 will be used for illustration.

Definition 1: Service Class Region (SCR)

Service Class Region is a well-formed sub-structure of a process definition, which contains one or more service classes. A sub-structure is well-formed if

(1) it is connected;

(2) for any SPLIT/MERGE node contained by the sub-structure, its corresponding MERGE/SPLIT node and the nodes between them should also be contained;

(3) for any LOOPSTART/LOOPEND node contained by the sub-structure, its corresponding LOOPEND/LOOPSTART node and the nodes between them should also be contained.

Formally, Service Class Region with regard to a process definition can be notated using the following grammar:

$$scr := sc \mid (op, \langle \text{ordered list of } scr \rangle)$$

where:

- *sc* represents the original service class in the process definition;
- op = { $\triangleright$, +, $\otimes$, μ }, which represent the composition patterns for sequence, parallel, conditional and loop respectively.

Following the definition, rules to identify SCR for a process definition can be listed straightforwardly (Table I).

TABLE I. Service Class Region Identification Rules

ID	Identification Rule	Examples
R1	An existing service class can be regarded as SCR.	sc_1, sc_2, sc_6
R2	A well-formed split/merge structure (parallel or conditional pattern) is identified as a SCR.	(+, sc_7, sc_8)
R3	A well-formed loop structure is identified as a SCR.	(μ , sc_9)
R4	A SCR can be merged from any two adjacent SCR. Using this rule, SCR can be recursively identified.	($\triangleright$, sc_6, (+, sc_7, sc_8))

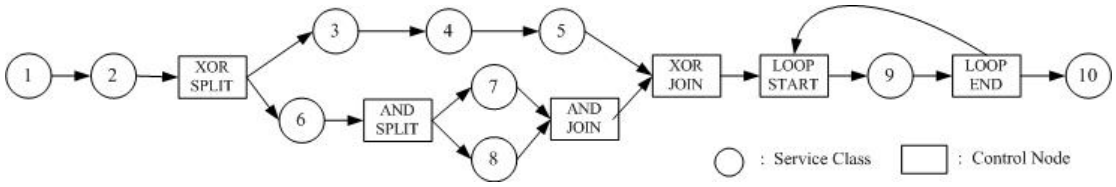

Figure 2. Comprehensive Example of Process Definition

If a SC *sc* is in the SCR *scr*, then we call *scr* **contains** *sc*, and the relation is expressed as $sc \in scr$.

For two SCR scr_1, scr_2, if $\forall\ sc \in scr_2 \rightarrow sc \in scr_1$, then we also call scr_1 **contains** scr_2, and the relation is expressed as $scr_1 \succ scr_2$ or $scr_2 \prec scr_1$.

Definition 2: Virtual Service Class (VSC)

Any Service Class satisfying the conditions below is called Virtual Service Class of the process definition:

(1) can achieve partial or overall goal of the web service composition;

(2) is or can be derived from existing Service Classes in the process definition.

Compared with the service classes in the given process definition, Virtual Service Class can be more fine-grained, which is decomposed from an existing service class; or be more large-grained, which is merged from existing service classes. In this paper, we focus on the more large-grained VSC, and this kind of Virtual Service Class can be defined based on the concept of SCR:

$$vsc = IF(scr) \quad (1)$$

where the operation IF(*scr*) is used to get the interface and functional specification for the Service Class Region *scr*, which can be deduced from the service classes and their composition structures contained by *scr*. Examples of VSC in Fig. 2: $vsc_1 = IF(\triangleright, sc_1, sc_2)$, $vsc_2 = IF(\triangleright, sc_6, (+, sc_7, sc_8))$, $vsc_3 = IF(\otimes, (\triangleright, sc_3, sc_4, sc_5), (\triangleright, sc_6, (+, sc_7, sc_8)))$.

We also introduce SCR(*vsc*) to represent the Service Class Region for the Virtual Service Class *vsc*. For example, $SCR(vsc_1) = (\triangleright, sc_1, sc_2)$.

VSC_SET(*pd*) is used to denote the complete set of VSC for the process definition *pd*.

Corollary 1: The process definition for web service composition can be identified as VSC, which is the most large-grained service class.

Definition 3: Trivial Virtual Service Class (TVSC)

A Virtual Service Class *vsc* is trivial if SCR(*vsc*) is an existing service class or contains service classes aggregated only by sequence pattern. SCR for TVSC can be identified by using R1 and R4 only.

Definition 4: Non-Trivial Virtual Service Class (NTVSC)

A Virtual Service Class *vsc* is non-trivial if SCR(*vsc*) contains any parallel/conditional/loop pattern.

With the notion of Virtual Service Class, process definition can be instantiated by WS of various granularities. The process definition can be denoted using a similar notation as SCR. For instance, the process definition in Fig. 2 can be represented as:

$pd_1 = (\triangleright, sc_1, sc_2, (\otimes, (\triangleright, sc_3, sc_4, sc_5), (\triangleright, sc_6, (+, sc_7, sc_8))), (\mu, sc_9), sc_{10})$

Other possible process definitions for the same web service composition are listed as follows:

$pd_2 = (\triangleright, vsc_1, (\otimes, (\triangleright, sc_3, sc_4, sc_5), (\triangleright, sc_6, (+, sc_7, sc_8))), (\mu, sc_9), sc_{10})$

$pd_3 = (\triangleright, vsc_1, (\otimes, (\triangleright, sc_3, sc_4, sc_5), vsc_2), (\mu, sc_9), sc_{10})$

$pd_4 = (\triangleright, sc_1, sc_2, vsc_3, (\mu, sc_9), sc_{10})$

Definition 5: VSC Granularity

The granularity of VSC is defined as the number of service classes in its SCR. We use Gra(*vsc*) to represent the granularity for the VSC *vsc*. For example, $Gra(vsc_1) = 2$, $Gra(vsc_2) = 3$, $Gra(vsc_3) = 6$.

Definition 6: VSC Nesting Level

The nesting level of VSC is defined as the maximum nesting level in its SCR. We use NL(*vsc*) to represent nesting level of VSC *vsc*. For example, $NL(vsc_1) = 0$, $NL(vsc_2) = 1$, $NL(vsc_3) = 2$.

Definition 7: VSC Search Space (VSS)

Assume *pd*, *gra* and *nl* are used to represent the process definition, the threshold of VSC Granularity and the threshold of VSC Nesting Level respectively, then VSC Search Space can be formally defined as:

VSS(*pd*, *gra*, *nl*)

where:

$VSS(pd, gra, nl) \subseteq VSC\_SET(pd)$,

$\forall\ vsc \in VSS(pd, gra, nl) \rightarrow Gra(vsc) <= gra$,

$\forall\ vsc \in VSS(pd, gra, nl) \rightarrow NL(vsc) <= nl$,

By the definition, VSC_SET(*pd*) is actually the largest VSS for the process definition *pd*.

With the above definitions, we can now define the problem:

Definition 8: QoS-based Multi-Granularity Service Selection Problem (QMGSSP)

Given a process definition *pd* and the VSC Search Space (*pd*, *gra*, *nl*), QMGSSP aims to select WS for each VSC in the restructured process definition, which optimizes the user-defined objective and meets the end-to-end QoS constraints at the same time. VSC in the restructured process definition should be the member of the given VSC Search Space.

Conceptually, the solution can be divided into two sub-processes:

SP1: Process Definition Restructuring

In this sub-process, some services classes will be merged as VSC in order to allow more large-grained WS to be available for selection.

SP2: QoS-based Service Selection

In this sub-process, WS is selected for each SC based on the objective and QoS constraints given the process definition. This is exactly the traditional service selection process.

The two sub-processes could be executed interleavedly to achieve the optimum solution.

Corollary 2 Traditional service selection problem is a special kind of QMGSSP with VSS(*pd*, 1, 0).

Definition 9: TVSC-capable QMGSSP

A QMGSSP is TVSC-capable if the VSS is in the form of (*pd*, *gra*, 0). In other words, only TVSC is considered during SP1.

Definition 10: NTVSC-capable QMGSSP

A QMGSSP is NTVSC-capable if there is no constraint on the form of the VSS. Both TVSC and NTVSC are considered during SP1.

In this paper, we present the solutions for TVSC-capable QMGSSP in Section IV.

IV. TVSC-capable Multi-Granularity Services Selection

Linear Programming is an optimization method, which aims to maximize or minimize a linear objective function, subject to a series of linear constraints. The constraints can be equality or inequality, and variables should be continuous. Frequently, for many problems, some variables can take only integer values. These problems are called mixed integer linear programming (MILP) problem, which are NP-complete [14].

As far as we are concerned, Ref. [2, 4] firstly introduces the integer programming(IP) method to the problem of QoS-aware services composition. In his approach, the composite web service is modeled as the statecharts and splits into multiple execution paths. Execution path contains a set of tasks (or service classes) $\{sc_1, sc_2, \ldots, sc_n\}$ such that sc_1 is the initial service class, sc_n is the final service class, and no service classes belong to alternative branches. Then, IP problem is formulated for each execution path, and these "partial" solutions will be merged to get an overall solutions for the web service composition. For service classes belong to more than one execution path, the selection result in the "hot path" will be adopted, where hot path could be defined as the most frequently executed execution path.

Our MILP-based approach for TVSC-capable QMGSSP is developed based on Ref. [2, 4]'s approach, which consists of the following five steps:

1. Reduce the VSC search space.
2. Choose one execution path *ep* as the hot path.
3. Formulate the MILP problem for TVSC-capable QMGSSP with regard to the hot path *ep*.
4. Formulate the MILP problem for TVSC-capable QMGSSP with regard to other execution paths
5. Merge the "partial" solution from (3) and (4) to get the overall solution.

Step 1 aims to reduce the VSC search space in order to accelerate the services selection process, which is discussed in Section A. In Section B, we propose the MILP problem formulation for TVSC-capable QMGSSP with regard to the hot path *ep* (Step 3) and discuss related issues in detail. Similar MILP problem can be formulated for other execution paths (Step 4), and we mention the difference at the end of Section B. Other steps have been discussed in existing literatures [2, 4].

A. VSC Search Space Reduction

Definition 11: Sequence Segment

Any SCR *scr* that meets the following conditions is called Sequence Segment of the process definition *pd*:

$scr \in \mathrm{VSS}(pd, g, 0)$,

$\forall\, scr_i \in \mathrm{VSS}(pd, g, 0) \rightarrow scr_i \prec scr$, or scr_i and *scr* contain no common Service Class.

Intuitively, Sequence Segment is a "longest" sub-structure of the process definition that is aggregated only by sequence pattern. For example, ($\triangleright$, sc_1, sc_2), ($\triangleright$, sc_3, sc_4, sc_5) and (sc_6) are all identified as Sequence Segments.

Assume there are *k* Sequence Segments in the process definition *pd*, and m_i is the number of SC contained by the *i*-th Sequence Segment, then the size of VSC search space for TVSC-capable QMGSSP can be calculated as:

$$\#(\mathrm{VSS}(pd, g, 0)) = \sum_{i=1}^{k} S_k \tag{2}$$

$$S_i = (m_i^2 + m_i) / 2 \tag{3}$$

where $\#(x)$ is used to denote the number of elements in the set *x*, and S_k is the number of VSC contained by the *k*-th Sequence Segment. For example, six VSC can be identified for the Sequence Segment ($\triangleright$, sc_3, sc_4, sc_5), namely sc_3, sc_4, sc_5, ($\triangleright$, sc_3, sc_4), ($\triangleright$, sc_4, sc_5), ($\triangleright$, sc_3, sc_4, sc_5).

Our approach for reducing VSC search space is based on the local QoS constraints and scoring for each VSC. The top k VSC will be chosen for the next step.

The scoring function for the VSC is designed as follows:

$$\mathrm{score}(vsc) = \frac{Gra(vsc) * h_Q(vsc)}{h_Q(scr)} * \frac{\max U(vsc)}{\max U(scr)} * \mathrm{p} \tag{4}$$

where:

$h_Q(vsc)$ is the number of candidate WS for *vsc* that meets local QoS constraints;

$h_Q(scr) = \sum_{sc \in scr} h_Q(sc)$, where $h_Q(sc)$ is the number of candidate WS for service class *sc* that meets local QoS constraints and $scr = \mathrm{SCR}(vsc)$;

$\max U(vsc)/U(scr)$ is the highest utility score that can be achieved for *vsc*/*scr* respectively. The way to calculate utility score will be introduced in the following section.

p is the execution probability of the path where *vsc* locates. If *vsc* does not belong to a conditional branch, then $p = 1$.

Here the local QoS constraints are used to filter out low-quality WS during the scoring process. They can be specified by the domain expert or estimated from history records. We use VSSR(*pd*, *g*, *nl*) to represent the reduced VSC Search Space.

B. MILP Problem Formulation for QMGSSP

In this section, we present MILP problem formulation for QMGSSP. To differentiate from traditional approach, the proposed one is referred as "MGC-aware approach".

Variables

The decision variables are defined as:

$$y_{ij} = \begin{cases} 1, & sc_i \text{ is bounded with } ws_{ij} \\ 0, & \text{otherwise} \end{cases}, \quad \forall\, i \in A$$

where $A = \{ i \mid \forall sc_i \in ep\}$

Assume n is the number of service classes and m is the number of candidate WS for the service class.

Objective

$$\max \sum_{i=1}^{4} w_i * \mathrm{Norm}(Q_i) \quad (5)$$

For i = 1, 2 (decreasing dimensions, i.e. response time, price)

$$\mathrm{Norm}(Q_i) = \begin{cases} \dfrac{\max Q_i - Q_i(ep)}{\max Q_i - \min Q_i} & if\ \max Q_i \neq \min Q_i \\ 1 & otherwise \end{cases} \quad (6)$$

For i = 3, 4 (increasing dimensions, i.e. reputation, reliability)

$$\mathrm{Norm}(Q_i) = \begin{cases} \dfrac{Q_i(ep) - \min Q_i}{\max Q_i - \min Q_i} & if\ \max Q_i \neq \min Q \\ 1 & otherwise \end{cases} \quad (7)$$

w_i is weight facotr for i-th QoS, and $\sum_{i=1}^{4} w_i = 1$.

Assignment Constraint

$$\sum_{j=1}^{m} y_{ij} = 1 \qquad \forall i \in A \quad (8)$$

QoS Constraint

(1) Delay Constraint (D)

$$Q_1(ep) = \sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} * \mathrm{RT}(ws_{ij}) * z_i <= D \quad (9)$$

where: $z_i := \begin{cases} 1, & sc_i \text{ is on the critical path} \\ 0, & \text{otherwise} \end{cases}$

$$d_i = \sum_{j=1}^{m} \mathrm{RT}(ws_{ij}) * y_{ij} \quad (10)$$

$$t_k >= d_i + t_i, \ \forall sc_i \rightarrow sc_k \quad (11)$$

$$Q_1(ep) >= t_i + d_i, \ \forall sc_i \in (ep) \quad (12)$$

where: Critical Path is a path from the initial WS to the final WS that has the longest total sum of response time [2, 4].

$sc_i \rightarrow sc_k$ means sc_i is executed before sc_k,

t_i, t_k is the starting time to execute sc_i and sc_k, which are continuous variables.

(2) Budget Constraint (B)

$$Q_2(ep) = \sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} * \mathrm{Cost}(ws_{ij}) <= B \quad (13)$$

(3) Reputation Constraint (R)

$$Q_3(ep) = \frac{1}{n} * \sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} * \mathrm{Rep}(ws_{ij}) >= R \quad (14)$$

(4) Reliability Constraint (S)

The aggregated reliability should be firstly linearized:

$$\ln(\prod_{i=1}^{n}\prod_{j=1}^{m} \mathrm{Rel}(ws_{ij})^{z_{ij}}) = \sum_{i=1}^{n} \ln(\prod_{j=1}^{m} \mathrm{Rel}(ws_{ij})^{z_{ij}})$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij} * \ln(\mathrm{Rel}(ws_{ij}))$$

let $\mathrm{Rel}'(ws_{ij}) = \ln(\mathrm{Rel}(ws_{ij}))$, then

$$Q_4(ep) = \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij} * \mathrm{Rel}'(ws_{ij}) >= S \quad (15)$$

Above is MILP problem formulation for traditional QoS aware service composition, which has been mentioned in Ref. [2-5]. To enable QoS-based selection from multi-granularity candidate WS for the composition, we propose a novel mechanism that maps VSC candidate WS to SC candidate WS and introduce additional linear constraints to form MILP problem for TVSC-capable QMGSSP. We use a simple example to illustrate our idea:

pd: ($\triangleright$, sc_1, sc_2, sc_3)

VSSR(pd, 2, 0) = {sc_1, sc_2, sc_3, vsc_1, vsc_2}

vsc_1 = ($\triangleright$, sc_1, sc_2),

vsc_1 = ($\triangleright$, sc_2, sc_3)

We use CDT(x) to represent the set of candidate WS for SC/VSC, assume:

$CDT(sc_1) = \{ws_{11}\}$,

$CDT(sc_2) = \{ws_{21}, ws_{22}\}$

$CDT(sc_3) = \{ws_{31}\}$,

$CDT(vsc_1) = \{ws_{011}\}$,

$CDT(vsc_2) = \{ws_{021}\}$

The 4-tuple (i, j, s, t) is used to indicate the mapping relationship, which means t-th candidate WS of vsc_s is mapped to j-th candidate WS of sc_i. Then four mapping relations exist in this example:

(1, 12, 1, 011),

(2, 23, 1, 011),

(2, 24, 2, 021),

(3, 32, 2, 021)

For example, for (1, 12, 1, 011), a "virtual" WS ws_{12} will be added as candidate WS for sc_1, which is mapped from ws_{011}. We use ECDT(sc_i) to represent the set of candidate WS with the "virtual" WS added. For example

$ECDT(sc_1) = \{ws_{11}, ws_{12}\}$,

$ECDT(sc_2) = \{ws_{21}, ws_{22}, ws_{23}, ws_{24}\}$

Note a candidate WS of VSC can be mapped to multiple "virtual" WS, which are correlated. For example, ws_{011} is mapped to ws_{12} and ws_{23}. During the selection, when ws_{12} is selected, ws_{23} should also be selected, and vice versa. To ensure the requirement, we introduce the Multi-Granularity Constraints (MGC). Two MGC are needed for the above example,

$$y_{12} = y_{23},\ y_{24} = y_{32}$$

where y_{12}, y_{23}, y_{24} and y_{32} are decision variables for ws12, ws_{23}, ws_{24} and ws_{32} respectively.

Based on the idea, we propose the MILP-based solution for TVSC-capable QMGSSP as follows:

Step 1. Map VSC candidate WS to SC candidate WS and store the mapping relations for later use. Also, MGC will be generated in the process. (Algo. 1)

Step 2. Add MGC to the MILP formulation

Step 3. Solve the MILP problem using any MILP solver such as lp_solve [15].

Step 4. Interpret the MILP solution. Since the selected WS may be "virtual" (mapped from VSC candidate WS), they have to be restored to the actual WS. (Algo. 2)

Algo 1 Multi-Granularity Web Service Pre-processing

Input:

VSSR(*ep*, *g*, *nl*), where *ep* is an execution path and VSSR(*ep*, *g*, *nl*) is the subset of VSSR(*pd*, *g*, *nl*) contains VSC that belongs to *ep*.

CDT(*vsc/sc*): set of candidate WS for each VSC/SC.

Output:

mgcs: set of linear equations that represents MGC.

ws_map: set of (*i*, *j*, *s*, *t*), which means *j*-th candidate WS of sc_i is mapped to *t*-th candidate WS of vsc_s.

```
01 for each vsc_s ∈ VSSR(ep, g, nl)
02   if (Gra(vsc_s) > 1)
03     set mi = min {i | sc_i ∈ SCR(vsc_s)}
04     for each ws_t ∈ CDT(vsc_s)
05       for each sc_i ∈ SCR(vsc_s)
06         set j = #ECDT(sc_i) + 1
07         add the 4-tuple (i, j, s, t) to ws_map
08         if (i == mi)
09           QoS(ws_ij) = QoS(ws_st)
10         else
11           QoS(ws_ij) = [0, 0, Rep(ws_st), 1],
12         end if
14       end for
15     end for
16     for each ws_t ∈ CDT(vsc_s)
17       for all sc_p, sc_q ∈ SCR(vsc_s) and p ≠ q
18         add the equation "y_pa = y_qb" to mgcs,
19         where (p, a, s, t), (q, b, s, t) ∈ ws_map
20       end for
21     end for
22   end if
23 end for
```

The main steps of the algorithm are as follows:

- **Line 3:** Get the minimum index of SC contained by the *vsc*.
- **Line 4–15:** Each candidate WS of *vsc* is mapped to a "virtual" WS of every SC that belongs to SCR(*vsc*) and the mapping is stored in *ws_map* (Line 5-7). The QoS of *vsc* candidate WS is inherited by the candidate WS of SC which has the minimum index (Line 8–12).
- **Line 16–21:** Generate the MGC.

At this point, MGC output by Algo. 1 can be added to formulate MILP problem for TVSC-capable QMGSSP:

Multi-Granularity Constraint

$$y_{ij} = y_{pq}, \tag{16}$$

where:

$i \neq p$;

$\forall$ (*i*, *j*, *s*, *t*), (*p*, *q*, *s*, *t*) $\in$ *ws_map*, $\forall$ $vsc_s \in$ VSSR(*ep*, *gra*, 0), $\forall$ $ws_{st} \in vsc_s$, $\forall$ sc_i, $sc_p \in$ SCR(vsc_s).

Algo 2 Multi-Granularity Web Service Post-processing

Input:

ws_map: output of Algo 1.

ws_selected: set of (*i*, *j*), which represents *i*-th SC is bounded with *j*-th candidate WS. This is output of Step 3.

Output:

ws_set: set of (*sc*, *j*) or (*vsc*, *j*), where *sc*/*vsc* is SC/VSC in the restructured process definition and its *j*-th candidate WS is selected.

```
01 set n as the number of SC in ep
02 for i = 1 to n
03   if (sc_i is not instantiated)
04     get (i, j) from ws_selected
05     if ( ∃ (a, b, c, d) ∈ ws_map,
                 where a == i and b == j)
06       set s = c, t = d
07       add (vsc_s, t) to ws_set
08       for each (a, b, c, d) ∈ ws_map,
                 where c == s and d == t
09         mark sc_a as instantiated
10       end for
11     else
12       add (sc_i, j) to ws_set
13       mark sc_i as instantiated
14     end if
15   end if
16 end for
```

The main steps of the algorithm are as follows:

- **Line 5-10:** If a new VSC *vsc* replaces existing SC in the original process definition, add the entry (*vsc*, index of selected WS) to *ws_set* (Line 7) and any *sc* $\in$ SCR(*vsc*) will be marked as instantiated (Line 9).
- **Line 11-14:** If original SC is still in the restructured process definition, simply add (*sc*, index of selected WS) to *ws_set* (Line 12) and marked *sc* as instantiated.

We have discussed MILP-based solution for TVSC-capable QMGSSP with regard to the hot path. For other execution paths, they can be divided into two parts:

- common path: SC on the common path also belong to hot path
- unique path: SC on the unique path does not belong to hot path.

For common path, its structure will adopt the one in the hot path, i.e. only WS needs to be selected for the specified structure. However, for the unique path, respective MILP problem can be formulated to choose WS of various granularities to optimize the objective.

V. EXPERIMENTATION

The purpose of our experiments is to evaluate the performance cost and effectiveness of MGC-aware approach compared to the traditional MILP-based QoS-aware services composition method. All experiments were conducted under Windows XP SP3, running on a LENOVO machine with 2 Intel Duo 2.33GHz processors and 2 GB RAM. We developed a program to simulate the services selection process, which is able to: (1) generate the process definition and its VSC; (2) generate the candidate WS with reasonable QoS values for SC/VSC; (3) output the MILP problem definition file for QMGSSP, which can be solved using the open source integer linear programming system lpsolve 5.5 [15]; (4) analyze the result file from lpsolve. The program is developed using Java 2 Standard Edition v1.6.0.

QoS values of WS is simulated based on the current use in the literature [5, 16], where response time and price are randomly generated from the normal distributions (μ =100, σ=40), (μ =100, σ=30) respectively, while reputation and reliability obey the uniform distributions [7, 10], [0.98, 0.99999] respectively. By default, the number of Sequence Segments in a process definition is set to 15 and length of each Sequence Segment is randomly generated from the uniform distribution [2, 15]. We also set QoS weights to [0.3, 0.3, 0.2, 0.2]. Other parameters are described in individual experiment.

A. Computation Cost Evaluation

In this series of experiments, the impact on the performance of services selection process is evaluated when multi-granularity WS are considered. We executed extensive test cases of different scales, where both the number of service classes in the process definition and candidate WS for each SC are varied from 10 to 100 with a step of 10. We give a description for a typical case that SC number and candidate WS number are both set to 50. Similar results can be obtained for other test cases.

For this test case, traditional MILP-based method cost 0.231 second in average to achieve the optimum solution. In contrast, the computation time needed for MGC-aware approach is illustrated using contour in Fig. 3. The x-axis represents the product of VSC number and candidate WS number of the VSC, e.g. 600 can correspond to 20 VSC and 30 candidate WS for each VSC, or 30 VSC and 20 candidate WS for each VSC (the experiment result reveals the computation cost is almost the same as long as their product is the same); the y-axis is the granularity of VSC search space. For each pair, MGC-aware method was run 10 times to get the average computation cost.

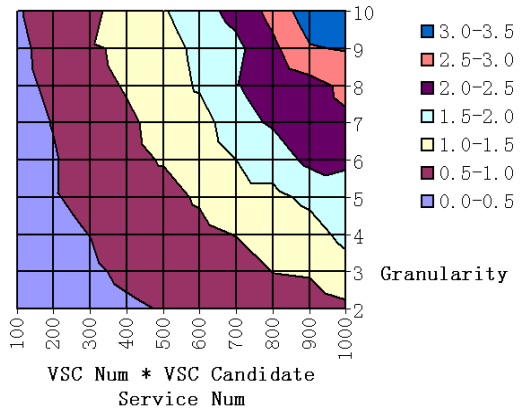

Figure 3. Computation Cost Evaluation

Fig. 3 shows that the computation cost increases with both the product and granularity. More specifically, the increase is slow when the value of product and granularity is small and becomes faster as the value of these two dimensions get larger. It is reasonable to expect that most VSC will have a small granularity (<=5) since the functionality of VSC will become more specialized as the granularity get larger and less providers would provide such services. For pairs (600, 5), (1000, 5), the computation cost is 1.059 second and 1.836 second respectively, which should be acceptable.

B. Effectiveness Evaluation

The second series of experiments aims at evaluating the effectiveness of our approach compared with traditional method, which is measured quantitatively using the improved percentage of the score for the optimum solution.

Fig. 4 demonstrates the result where service class number was set to 30 and QoS constraints are set to [2700, 2700, 9, 0.9]. Candidate WS number is varied from 10 to 50 with a step of 5. The score of solution for each case obtained using traditional method is presented in the datatable under x-axis. There are 5 lines in the figure and each line uniquely corresponds with specific values for VSC-related parameters. For example, [5*20, 3] represents the setting that the number of VSC is 5, the number of candidate WS for VSC is 20 and the granularity of VSC search space is 3. For each setting, the selection process is repeated 10 times to get the average score.

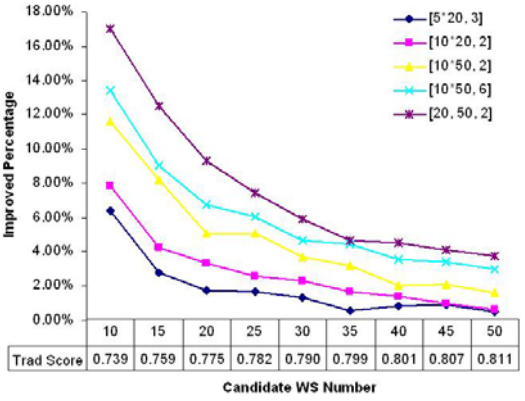

Figure 4. Effectiveness Evaluation

Fig. 4 indicates general advantages of our approach. To be more specifically, our approach is more effective when:

1) candidate WS number of SC is relatively small;
2) VSC number and its candidate WS number is larger;
3) the granularity of VSC search space is larger.

We also investigate the influence of QoS constraints on the effectiveness of our approach. Table II presents the results. Again, each score is the average value from results of 10 executions.

The result shows that as QoS constraints get stricter, our approach become more effective in that it can sustain high success rate and achieve better solutions.

TABLE II. INFLUENCE OF QOS CONSTRAINTS ON THE SOLUTION

QoS Constraint				Trad Approach		MGC Approach		Improved Percentage
RT	Price	Rep	Rel	Success Rate	Score	Success Rate	Score	
4000	4000	8.0	0.70	100%	0.802	100%	0.832	3.81%
3500	3500	8.5	0.80	100%	0.800	100%	0.839	4.86%
3000	3000	9.5	0.95	55.55%	0.717	100%	0.782	9.07%
2800	2800	9.5	0.95	38.46%	0.737	100%	0.797	8.14%
2700	2700	9.5	0.95	11.70%	0.758	100%	0.802	5.82%

V. CONCLUSION & FUTURE WORK

In this paper, we introduce the concept of "granularity" to WS and propose the QoS-based Multi-Granularity Service Selection Problem (QMGSSP). QMGSSP is a generalization of traditional QoS-aware services composition problem, in that the latter is a special kind of QMGSSP. By allowing WS of various granularities to be considered as the candidate for selection, we can obtain a better solution. Furthermore, we formulate MILP problem for QMGSSP by introducing Multi-Granularity Constraints (MGC). Experiments show the effectiveness of our approach and the performance cost is acceptable.

Future work will consider the more general form of QMGSSP, which is NTVSC-capable. In addition, we will research on applying data mining techniques to identify correlations in service classes so that VSC can be discovered more accurately.

REFERENCES

[1] Y. T. Liu, A. H. H. Ngu, L. Z. Zeng, "QoS computation and policing in dynamic web service selection," WWW (Alternate Track Papers & Posters) 2004, pp. 66-73.

[2] L. Z. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, "QoS-Aware Middleware for Web Services Composition," IEEE Trans. Software Eng, 30(5), pp. 311-327, 2004.

[3] D. Ardagna and B. Pernici, "Global and Local QoS Guarantee in Web Service Selection," Business Process Management Workshops, 2006, pp. 32-46.

[4] L. Z. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Z. Sheng, "Quality driven web services composition," WWW 2003, pp. 411-421.

[5] D. Ardagna B. Pernici, "Adaptive Service Composition in Flexible Processes," IEEE Trans. Software Eng, 33(6), pp. 369-384, 2007.

[6] F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, S. Dustdar, „An End-to-End Approach for QoS-Aware Service Composition," IEEE EDOC, 2009, pp. 151-160.

[7] G. Canfora, M.D. Penta, R. Esposito, M.L. Villani, "A lightweight approach for QoS-aware service composition," ICSOC 2004.

[8] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," GECCO, 2005, pp. 1069-1075.

[9] K. J. Ren, N. Xiao, J. Q. Song, C. Yang, M. Zhu, and J. J. Chen, "Gradual Removal of QoS Constraint Violations by Employing Recursive Bargaining Strategy for Optimizing Service Composition Execution Path," ICWS 2009, pp. 485-492.

[10] J. H. Jang, D. H. Shin, K. H. Lee, "Fast Quality Driven Selection of Composite Web Services," ECOWS 2006, pp. 87-98.

[11] M. C. Jaeger, G. Rojec-goldmann, G. Muhl, "QoS aggregation for Web service composition using workflow patterns," IEEE EDOC Workshop, 2004, pp. 149-159.

[12] M. C. Jaeger, G. Rojec-Goldmann, G. Mühl, "QoS Aggregation in Web Service Compositions," EEE 2005, pp. 181-185.

[13] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, K. Kochut, "Quality of service for workflows and web service processes," Journal of Web Semantics, 1(3), 2004, pp. 281-308.

[14] Wolsey. L, Integer Programming. New York, USA: John Wiley and Sons, 1998.

[15] Berkelaar, et al. 2009. lpSolve: Open source (mixed-integer) linear programming system, Sourceforge. Available at: http://lpsolve.sourceforge.net/ [2010-05-25]

[16] E. Al-Masri, Q. H. Mahmoud, "Investigating web services on the world wide web," WWW 2008, pp. 795-804.