Utility-Based Optimal Service Selection for Business Processes in SOA

Vinod K. Dubey, Daniel Menascé

Summarized by
Uma Tadakamalla
Outline

I. Introduction

II. Problem Statement

III. QoS Composition

IV. Utility Functions

V. Optimal Service Selection

VI. Experimental Evaluation

VII. Conclusions
I. Introduction

- Service Oriented Architecture
  - Services
  - Service providers
- Problem: Optimal service selection
  - Support business processes
  - Maximizes a utility function under multiple QoS and cost constraints
II. Problem Statement

- **Optimization problem**

- Find a service selection that maximizes a utility function of multiple QoS objectives - average end-to-end execution time, availability and throughput and subject to constraints.

\[
\text{Maximize } U(\mathbb{E}[R(\mathbb{Z})], A(\mathbb{Z}), X(\mathbb{Z}))
\]

subject to

\[
\begin{align*}
\mathbb{E}[R(\mathbb{Z})] & \leq R_{\text{max}} \\
A_{\text{min}} & \leq A(\mathbb{Z}) \leq 1 \\
X(\mathbb{Z}) & \geq X_{\text{min}} \\
C(\mathbb{Z}) & \leq C_{\text{max}}
\end{align*}
\]

\[z \in \mathbb{Z}\]
III. QoS Composition

• **Expected End-to-End Execution time**

➢ Service providers specify response time through a probability distribution.
➢ Uses Jensen’s in-equality for computing Expected End-to-End Execution time based on the individual service response times.
III. QoS Composition

- **End-to-End Availability computation**

  BPEL files are parsed using SAX parser to find expressions

```
Algorithm 1 Availability Computation of a BPEL process
1: function A(node i)
2: if label(i) = leaf node then
3:   return A_i;
4: else
5:   if label(i) = sequence then
6:     return \prod_{k \in \text{children}(i)} A(k);
7:   else if label(i) = switch then
8:     return \sum_{k \in \text{children}(i)} q_k \times A(k);
9:   else if label(i) = flow then
10:    return \prod_{k \in \text{children}(i)} A(k);
11: end if
12: end if
```

III. QoS Composition

- **End-to-End Throughput computation**
  BPEL files are parsed using SAX parser to find expressions

```plaintext
Algorithm 2 Throughput Computation of a BPEL process
1: function X(node i)
2: if label(i) = leaf node then
3:   return X_i;
4: else
5:   if label(i) = sequence then
6:     return min_{k ∈ children(i)} X(k);
7:   else if label(i) = switch then
8:     return \sum_{k ∈ children(i)} q_k \times X(k);
9:   else if label(i) = flow then
10:    return min_{k ∈ children(i)} X(k);
11: end if
12: end if
```
IV. Utility Functions

**Utility functions**

- response time, availability and throughput

\[
U_i(v(z)) = K_i \frac{e^{\alpha_i(\beta_i-v(z))}}{1 + e^{\alpha_i(\beta_i-v(z))}}
\]

*where*

- \( z \): service selection
- \( v(z) \): end-to-end QoS metric
- \( K_i \): normalization factor
- \( \alpha \): sensitivity parameter
- \( \beta \): QoS objective for the QoS metric
IV. Utility Functions

- **Global Utility function**

  Weighted geometric mean of the utility functions of all three QoS metrics

  \[ U_g(z) = \left( \prod_{i=1}^{3} (U_i(z))^{w_i} \right)^{\frac{1}{\sum_{j}^{3} w_j}} \]

  where

  - \( z \): service selection
  - \( U_i(z) \): utility of a QoS metric
  - \( w_i \): importance weight of metric \( i \) in the range \((0,1)\)
  - Sum of all weights of QoS metrics = 1
V. Optimal Service Selection

- As the number of activities and number of service providers increase, the search space increases exponentially.
  - NP-hard problem
- Efficient Service selection algorithms
  - Extended JOSeS Algorithm
  - The HCB Heuristic Algorithm
V. Optimal Service Selection

1. **Extended JOSeS Algorithm**

   - JOSeS - Jensen-based optimal Service Selection
   - Handles multiple QoS metrics
   - Uses Jensen’s inequality to cut down the search space
     - If a sub-selection exceeds QoS or cost constraints then all selections that start with that sub-selection are not generated.
   - Works well for small to medium complex system.
   - Does not work for large and complex systems – too many points to examine in search space.
V. Optimal Service Selection

2. The HCB Heuristic Algorithm

- HCB – Hill Climbing Based technique

- Steps:
  1. Find a neighborhood of a point currently being visited
  2. Move to the best point in the neighborhood.
  3. Process is repeated until a near-optimal solution is found or stopping criteria is met.

- Uses several random restarts and selects the best allocation among all restarts to overcome local optimum issue
V. Optimal Service Selection

**Neighborhood selection**

- Replace service provider for each activity one at a time with the service provider that provides the best improvement in each of the three QoS metrics – maximum of 3N neighbors.

- The neighbor that meets all the constraints is added to feasible group.
VI. Experimental Evaluation

- 50 BPEL processes
- 6-9 activities, each activity provided by up to 7 service providers
- Response time assumed to be exponentially distributed
- Constraint strength (CS) – 10%, 20%, 30% and 40%
- Complexity levels: Simple, Medium and Complex
- Total = 36,000 runs
VI. Experimental Evaluation

- HCB makes service selections that are at least 99.5% of the optimum.
- The performance of HCB improves as the strength of the constraints (CS) decreases.
VI. Experimental Evaluation

- The number of points examined by HCB is several orders of magnitude lower than JOSeS algorithm.

- As CS increases, the number of points examined decreases significantly for JOSeS.
VI. Experimental Evaluation

➢ HCB took lot less time compared to JOSeS algorithm.

➢ As CS increases, the computation time decreases significantly for JOSeS.
VI. Experimental Evaluation

The HCB makes service selections that are at least 99.65% of the optimal JOSeS algorithm.

As the complexity of the business process increases, the utility of the heuristics algorithm decreases.

Figure 4. Average $U_h/U_o$ (%) vs. $n_{spa}$ for simple, medium, and complex business processes.
VI. Experimental Evaluation

- The number of points visited with HCB are lot less compared to those with JOSeS algorithm.

- Number of points examined increases significantly as the complexity increases for the JOSeS algorithm.
VI. Experimental Evaluation

As the number of service providers per activity increases, the average number of points visited grows linearly.

Figure 6. Average $N_h$ vs. $n_{spa}$
VII. Conclusions

HCB heuristic algorithm

- Performance (time and the number of points examined) is much better than that of the optimal JOSeS algorithm.

- Near optimal service selection – at least 96.5% of the optimal.

- The performance improves as the strength of the constraints (CS) decreases.

- As the complexity of the process increases, the utility decreases slightly.
Questions and Discussion