Top-k Web Service Composition in the Context of User Preferences

Karim Benouaret $^1$, Djamal Benslimane$^1$, Allel Hadjali $^2$

$^1$LIRIS, University of Lyon  
{karim.benouaret, djamal.benslimane}@liris.cnrs.fr  
$^2$IRISA, Rennes1 University  
allel.hadjali@enssat.fr
Outline

1. Introduction
2. Service composition based preference queries
3. Top-k service composition
4. Experimental evaluation
5. Conclusion
Outline

1. Introduction
2. Service composition based preference queries
3. Top-k service composition
4. Experimental evaluation
5. Conclusion
Problem description

- Data Web services
  - network accessible software entities
  - returning some information to the user (e.g., a weather forecast service or a news service)

- Data Web service composition
  - a combination of primitive Data Web services
  - answering user’s complex queries

- User preferences
  - important to customize the composition process
  - rank-order the Data Web service compositions
  - flexible manner: linguistic terms (e.g., “rather cheap" or "“not expensive")
  - modeled using fuzzy sets

- Objective: find the top-k Data Web service compositions according to user preferences
### Example

<table>
<thead>
<tr>
<th>Service</th>
<th>Functionality</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{11}(x, y)$</td>
<td>Returns the automakers $y$ in a given country $x$</td>
<td>-</td>
</tr>
<tr>
<td>$S_{21}(x, y, z, t)$</td>
<td>Returns the cars $y$ along with their prices $z$ and warranties $t$ for a given automaker $x$</td>
<td>$z$ is cheap, $t$ is short</td>
</tr>
<tr>
<td>$S_{22}(x, y, z, t)$</td>
<td></td>
<td>$z$ is accessible, $t$ is [12, 24]</td>
</tr>
<tr>
<td>$S_{23}(x, y, z, t)$</td>
<td></td>
<td>$z$ is expensive, $t$ is long</td>
</tr>
<tr>
<td>$S_{24}(x, y, z, t)$</td>
<td></td>
<td>$z$ is [9000, 14000], $t$ is [6, 24]</td>
</tr>
<tr>
<td>$S_{31}(x, y, z)$</td>
<td>Returns the power $y$ and the consumption $z$ for a given car $x$</td>
<td>$y$ is weak, $z$ is small</td>
</tr>
<tr>
<td>$S_{32}(x, y, z)$</td>
<td></td>
<td>$y$ is ordinary, $z$ is approximately 4</td>
</tr>
<tr>
<td>$S_{33}(x, y, z)$</td>
<td></td>
<td>$y$ is powerful, $z$ is high</td>
</tr>
<tr>
<td>$S_{34}(x, y, z)$</td>
<td></td>
<td>$y$ is [60, 110], $z$ is [3.5, 5.5]</td>
</tr>
</tbody>
</table>

$Q_1$ : “return the French cars, preferably at an affordable price with a warranty around 18 months and having a normal power with a medium consumption”
Overview of our approach

Challenges
- how to retain the most relevant services
- how to generate the top-k compositions

Contribution
- compute matching degrees between user preferences and services’ constraints
- propose a ranking criteria based on a fuzzification of Pareto dominance to select the most relevant services/compositions
- to avoid returning similar compositions, we also propose a diversified top-k compositions
Outline

1 Introduction

2 Service composition based preference queries

3 Top-k service composition

4 Experimental evaluation

5 Conclusion
Terminology

- \( Q := (q_1, \ldots, q_n) \): a preference query
- \( S = \{S_1, \ldots, S_n\} \): a set of service classes
- \( S_i = \{S_{i1}, \ldots, S_{in_i}\} \): a set functionally similar services
- \( S_i \sqsubseteq q_i \): services of \( S_i \) can be used to answer \( q_i \)
- \( M = \{M_1, \ldots, M_m\} \): a set of matching methods
### Matching degrees between services and query components

<table>
<thead>
<tr>
<th>$S_{i,j}$</th>
<th>$q_i$</th>
<th>CBM</th>
<th>G-IBM</th>
<th>L-IBM</th>
<th>K-IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{11}$</td>
<td>$q_1$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$S_{21}$</td>
<td></td>
<td>(1, 0.57)</td>
<td>(1, 0)</td>
<td>(1, 0)</td>
<td>(0.80, 0)</td>
</tr>
<tr>
<td>$S_{22}$</td>
<td></td>
<td>(0.89, 1)</td>
<td>(0, 1)</td>
<td>(0.90, 1)</td>
<td>(0.50, 1)</td>
</tr>
<tr>
<td>$S_{23}$</td>
<td></td>
<td>(0.20, 0.16)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>$S_{24}$</td>
<td></td>
<td>(0.83, 0.88)</td>
<td>(0.60, 0.50)</td>
<td>(0.60, 0.50)</td>
<td>(0.60, 0.50)</td>
</tr>
<tr>
<td>$S_{31}$</td>
<td></td>
<td>(0.50, 0.36)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>$S_{32}$</td>
<td></td>
<td>(0.79, 0.75)</td>
<td>(0, 0.25)</td>
<td>(0.60, 0.50)</td>
<td>(0.40, 0.50)</td>
</tr>
<tr>
<td>$S_{33}$</td>
<td></td>
<td>(0.21, 0.64)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>$S_{34}$</td>
<td></td>
<td>(0.83, 0.85)</td>
<td>(0.50, 0.50)</td>
<td>(0.50, 0.50)</td>
<td>(0.50, 0.50)</td>
</tr>
</tbody>
</table>
Current approaches

**Scoring function**
- computes a score for each service as an aggregate of the individual matching degrees
- requires users to assign weights to individual matching degrees
- users lose the flexibility to select their desired services
- one matching method

**Skyline**
- compromises the services which are not nominated
- privileges services with a large variance
- one matching method
Pareto dominance vs fuzzy dominance

- **Pareto dominance**: \( u \succ v \iff \forall i \in [1, d], u_i \geq v_i \land \exists k \in [1, d], u_k > v_k \)

- **Fuzzy dominance**: \( \text{deg}(u \succ v) = \frac{\sum_{i=1}^{d} \mu_{\succ}(u_i, v_i)}{d} \), where

  \[
  \mu_{\succ}(x, y) = \begin{cases} 
  0 & \text{if } x - y \leq \varepsilon \\
  1 & \text{if } x - y \geq \lambda + \varepsilon \\
  \frac{x - y - \varepsilon}{\lambda} & \text{otherwise}
  \end{cases} 
  \]

- **Comparison** \((u = (1, 0), v = (0.90, 1))\)
  - neither \( u \succ v \) nor \( v \succ u \)
  - \( \text{deg}(u \succ v) = 0.25 \) and \( \text{deg}(v \succ u) = 0.50 \) (\( \varepsilon = 0, \lambda = 0.2 \))
Associating score with a Service/Composition

- **Service’s score**: \( S_{ij} \in S_i \), indicates the average extent to which \( S_{ij} \) dominates the whole services of its class \( S_i \)

  \[
  FDS(S_{ij}) = \frac{1}{(|S_i|-1)m^2} \sum_{i=1}^{m} \sum_{S_{ik} \in S_i, k \neq i} \sum_{j=1}^{m} \text{deg}(S^n_{ij} \succ S^j_{ik})
  \]

- **Composition’s score**: \( C = \{S_{1j_1}, ..., S_{nj_n}\} \)

  \[
  FDS(C) = \frac{1}{d} \sum_{i=1}^{n} d_i \cdot FDS(S_{ij_i})
  \]
An efficient generation of top-k compositions

- **straightforward method:**
  - generate all possible compositions
  - compute their scores
  - return the top-k ones
  - high computational cost

- **Optimization technique (theorem 1):**

  \[ C = \{S_{1j_1}, \ldots, S_{nj_n}\} \]

  \[ \exists S_{ij_i} \in C; S_{ij_i} \notin top-k.S_i \implies C \notin top-k.C. \]
An efficient generation of top-k compositions (our example)

<table>
<thead>
<tr>
<th>Services</th>
<th>Class</th>
<th>Score</th>
<th>Top-k</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{11}$</td>
<td>$S_1$</td>
<td>-</td>
<td>$S_{11}$</td>
</tr>
<tr>
<td>$S_{21}$</td>
<td></td>
<td>0.487</td>
<td></td>
</tr>
<tr>
<td>$S_{22}$</td>
<td></td>
<td>0.653</td>
<td>$S_{22}$</td>
</tr>
<tr>
<td>$S_{23}$</td>
<td></td>
<td>0.035</td>
<td>$S_{24}$</td>
</tr>
<tr>
<td>$S_{24}$</td>
<td></td>
<td>0.538</td>
<td></td>
</tr>
<tr>
<td>$S_{31}$</td>
<td>$S_2$</td>
<td>0.094</td>
<td></td>
</tr>
<tr>
<td>$S_{32}$</td>
<td></td>
<td>0.593</td>
<td>$S_{32}$</td>
</tr>
<tr>
<td>$S_{33}$</td>
<td></td>
<td>0.130</td>
<td>$S_{34}$</td>
</tr>
<tr>
<td>$S_{34}$</td>
<td></td>
<td>0.743</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compositions</th>
<th>Score</th>
<th>Top-k</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1 = {S_{11}, S_{22}, S_{32}}$</td>
<td>0.623</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$C_2 = {S_{11}, S_{22}, S_{34}}$</td>
<td>0.698</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$C_3 = {S_{11}, S_{24}, S_{32}}$</td>
<td>0.566</td>
<td></td>
</tr>
<tr>
<td>$C_4 = {S_{11}, S_{24}, S_{34}}$</td>
<td>0.640</td>
<td></td>
</tr>
</tbody>
</table>

- Straightforward method: 16 compositions ($\prod_{i=1}^{n_i} |S_i|$)
- Our method: 4 compositions ($\leq k^{n_i}$)
Diversity-aware Top-k Compositions

- Different similar services could exist in each class $S_i$ leading to similar compositions
- Diversification is then needed to improve user satisfaction
- $\text{Quality}(S_{ij}) = FDS(S_{ij}) \times \text{RelDiv}(S_{ij}, dtopk.S_i)$
- $\text{RelDiv}(S_{ij}, dtopk.S_i) = \begin{cases} 
1 & dtopk.S_i = \emptyset \\
\frac{1}{\sum_{S_{ik} \in dtopk.S_i} \text{Dist}(S_{ij}, S_{ik}) / |dtopk.S_i|} & \text{otherwise}
\end{cases}$
Outline

1. Introduction
2. Service composition based preference queries
3. Top-k service composition
4. Experimental evaluation
5. Conclusion
time vs Parameters

(a) Number of candidate Services per class
(b) Service classes (query components)
(c) Max preferences involved in a service class
(d) Number of methods
(e) k
Conclusion & Future work

- **Conclusion**
  A framework that identify and retrieve the most relevant services and return the top-k compositions according to the user preferences

- **Future work**
  - user study to evaluate the quality of the results
  - Combine with QoS
Thank you