Applied and Scalable Optimization of Long-term and Network-aware Service Compositions

Adrian Klein
The University of Tokyo
D3, Honiden Lab
Outline

1. Preliminaries
2. Introduction
3. Approach
   A) Long-term Service Compositions
   B) Network-aware Service Compositions
4. Conclusion
5. Backup Slides
   A) Explanations, B) References,
   C) Other Problem Formalizations, D) Survey, E) Evaluations
1. PRELIMINARIES

General Overview $\rightarrow$ Definition of the QSC Research Problem
Services?

1. A **service** is a **software component** that encapsulates business logic and is accessible over the **network**.

2. **Atomic** services can be **composed** to achieve complex functionality.

3. The **number** of available services keeps **increasing**.

![Graph showing the number of Service APIs increasing from 2000 to 8000 over the years from 2005 to 2012.](www.programmableweb.com)
Example of a Service Composition

15th March 2013
QoS-aware Service Composition (**QSC**) Problem

A) **Services** with different **QoS** (Quality of Service)

B) Workflow Template

C) Service Selection

QoS Optimization

=> NP-hard!
-> DEFINITION
**Service**

**Form:**

input₁, input₂, ..., inputₙ → Service → output₁, output₂, ..., outputₘ

**Definition of the W3C**

- Stateless software component
- Public interface
- Invoked over the network

“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format[...]. Other systems interact with the Web service in a manner prescribed by its description using [...] messages, typically conveyed using HTTP[...].” - W3C
Example Service Descriptions

Inputs: \( \text{int } x \)
Outputs: list of \( \text{ints } \{d_1, ..., d_n\} \)
Precondition: \( x < 10^{10} \)
Postcondition: \( x = \prod d_i \) with all \( d_i \) prime

Price: 1$
Response time: \( \leq 100 \text{ ms} \)
Reliability: 99.5%

Approximate Quality of Service (QoS)

Factorization

\( x \mapsto \{d_1, d_2, ..., d_n\} \)
QSC Problem [ZBD+03]

**Inputs (from the service user 😊)**

- **QoS preferences** (=weights) 
- **QoS requirements** (=constraints) 
- **Workflow template**

\[
\text{util}(q, wf, sel) := \sum_i w_i Q_i
\]

(simplified)

**QSC Problem.**

Given \((wf, (w_T, w_P, w_R), (c_T, c_P, c_R)) \in WF \times \mathbb{R}^3 \times \mathbb{R}^3\)

Optimize \(sel \in SEL\)

Such that \(\text{util}(q, wf, sel) = \max_x \text{util}(q, wf, x)\)

and \(q(wf, sel)(T) \leq c_T\)

and \(q(wf, sel)(P) \leq c_P\)

and \(q(wf, sel)(R) \geq c_R\)

\[=> \text{Search Space} := \#S^\#WF\]

\#S := Services per Task

\#WF := Size of Workflow

Other Formalizations

- Linear Integer Programming Prob.
- Knapsack Problem
- Graph Problem
- Scheduling Problem

\[=> \text{Sufficient to some degree, NP-hard}\]
2. INTRODUCTION

Issues of QSC Research -> PhD Proposal
Top-level Issues

I. Applicability:
Gap of Model vs. Reality
(mostly unchanged since [ZBD+03])

1. Extend the QSC problem
2. Propose effective algorithms
   => better QoS [for users]

II. Scalability:
Fast Near-Optimality
(tackled since [YZL07])

3. Propose efficient algorithms
   => better QoS in time or
   => same QoS faster [for users]
Proposal

Observations

A) Workflow templates are often used more than once.

B) Services are often run by different users over the network.

Challenges of standard QSC

① Static service selection (→ I. App.) (deterministic + time-independent)

② Network-independent QoS (→ I. App.)

③ Network-unaware Optimization (→ II. Scal.)

QSC Extensions:

A) Long-term SC (L’QSC / LoSC ⇒ ①)

B) Network-aware SC (N’QSC / NeSC ⇒ ②+③)

=> Extend QSC problem to L’QSC+N’QSC problems

=> Propose effective and efficient LoSC+NeSC approaches
Approach: Focus

1. Problem

2. Approach

3. Contributions

1. Problem

2. Approach

3. Contributions

1. PRE

2. INTRO

3-A) LONG-TERM SC

3-B) NETWORK-AWARE SERV. COMP.

4. CON

5. BA
3-A) Long-Term Service Compositions

Overview -> Individual Contributions

=> Challenge ①
(static service selection)
Overview

Approach

1. Probabilistic Selection through Linear Programming

2. Time-dependent Selection through a Custom Genetic Algorithm

Benefits

1. Better QoS w/ constraints
2. Optimality + Scalability
3. Better QoS (+high reliability)
4. Scalability (for a harder problem)

Related Work

- A broker provides a long-term “virtual” service with a fixed number of QoS variations (solved with IP), Cardellini et al. [CCGP07] (recently: [CDVG+11]) => no user perspective + IP does not scale vs. our appr. (evaluated in 1.)

- Long-term deployment by a service provider (solved with MOO-GA), Wada et al. [WSO09] (recently: [WSYO12]) => no user perspective + MOO-GA less scalable than our SOO-GA

- Long-term run-time replacement of services (solved w/ Markov decision process), Na et al. [NZG+11] => applied after QSC problem at run-time + shorter “long-term” (run-time perspective)
ProbSel¹: [KIH10a] Long-term QoS with Probabilistic Service Selection

Solve partial L’QSC (no time-dependencies) through Linear Programming

=> Better QoS + Scalability

=> Address part of Challenge ① (static service selection)
Probabilistic Service Selection

**Execution Policy**

- X1: 100%, X2: 0%
- A3: 50%, A4: 50%
- B5: 50%, B6: 50%
- Y7: 50%, Y8: 50%

**Optimal Ratio**

1. X1 + A3 + B5 + Y8
2. X1 + A4 + B6 + Y7
3. X1 + A3 + B6 + Y8
4. X2 + A3 + B5 + Y8
5. X1 + A3 + B6 + Y8
6. X1 + A3 + B5 + Y7

15th March 2013
ImCom²: Improved Time Complexity for the QSC Problem

- [KIH11]

Custom Hill Climbing algorithm (HC* / \sqrt{SF}) with PROBSel¹ as initial bias

=> Observed time complexity vs. HC:

O(SF⁴) -> O(SF¹·₅)

(...linear in problem size = #S * #WF = SF²)

(SF := #S := #WF)

Side Result!
TIMRel³: Time-dependent QoS for Long-term Compositions

[S] [KWIH12] Long-term Compositions

Solve complete L’QSC with a Custom Genetic Algorithm
+ a variable number of time-dependent services (doubling as backup services)

=> Better QoS (incl. high reliability) + Scalability

=> Address Challenge ① (static service selection)

Cooperation with Florian Wagner

15th March 2013
Time-dependent Service Selection

**Probabilistic Patterns (QoS + Use)**

- **(a) Qos Pattern**
  - Response Time [ms]
  - Days of the week: Mo, Tu, We, Th, Fr, Sa, Su

- **(b) Usage Pattern**
  - Service Requests
  - Time intervals: 8-9, 10-11, 12-14, 15-16, 17-18

=> Search Space: $#S#WF \rightarrow (#S#WF)#TD$

- $#S$ := Services per Task
- $#TD$ := Nr. of Time Discretizations
- $#WF$ := Size of Workflow

**Time-dependent Execution Policies**

- **Mo-Fr**
  - Service A1 → Service A5 [in case of failure]

- **Sa/Su**
  - Service A5 → Service A1 [in case of failure]
Custom Genetic Algorithm

Adaptive GA

- Mutate+
- Crossover+
- AdaptGroups

Choose values $\notin$ group

Avoid duplicate/empty group members

Due to reliability constr. and current rel. (per Task, increase or decrease group size)

Probabilistic QoS Model

Probabilistic Patterns (QoS + Use)

Encoding with variable length

(a) Qos Pattern

(b) Usage Pattern

15th March 2013

サービスコンピューティング研究会、Adrian Klein
Evaluation: Better QoS (=Utility)

**Graph:**
- QoS vs. Utility
- Near-optimal!

**Legend:**
- \( GA_i \) Naive GA with static group size \( i \)
- \( TG_6^{UP} \) Custom Adaptive GA 
  (analog, with time-dependency)

15th March 2013

Adrian Klein
3-A) LONG-TERM SCs: Conclusion

1. Solved partial L’QSC
   – Approximated orig. QSC

2. Solved complete L’QSC

=> Better QoS + Scalability!
=> Addressed Challenge ①!
   (static service selection)

15th March 2013
3-B) NETWORK-AWARE SERVICE COMPOSITIONS

Overview -> Motivation -> Individual Contributions

=> Challenge ②
(network-independent QoS)

=> Challenge ③
(network-unaware Optimization)
Overview

Approach

1. **Network QoS**
   with a Distributed Architecture and an Augmented Network Model

2. **Network-aware Optimization**
   through a Genetic Algorithm with custom operators and self-adaptivity

Benefits

1. Reasonable **QoS** in distributed settings (with standard optimization)

1.+2.

− Near-optimal **QoS** in distributed settings
− **Scalability** in distributed settings (for a much harder problem)

Related Work

− Realization of a P2P service system, Schuler et al. [SWSS03]
  => **no QSC formalization** + **no network QoS**

− Distributed service overlay networks (Dijkstra+QoS ratio), Li et al. [LHD+07]
  => **no QoS constraints** + **no global QoS optimization**
  + does not scale vs. our appr. (→ service instances per task, evaluated in 2.)

− Partition a comp. service selection into decentralized processes, Nanda et al. [NCS04] (recently: [FYG09])
  => applied after QSC problem just before run-time

− Distributed execution acc. to chemical paradigm [LMJ10] or by agent model [FPT10] (both not solved)
  => **no QSC formalization** + **no network QoS**

15th March 2013
Motivation (1/4): Distributed Scenario

- Media Tr. ➔ Workflow
- Deployment Situation

Image Generation ➔ BW Transformation
Subtitle Generation

Video Generation ➔ XVGA Transcoding
Subtitle Generation

Network Model

15th March 2013
Motivation (2/4): Network-aware QoS

1. Network Latency

2. Network Bandwidth

=> 80ms (just exec.)

=> 155ms (distributed)

=> 1691ms (100 MB)
Motivation (3/4): **Complexity** -> Network-aware Optimization

**Standard**
- P providers for Task T (e.g. Amazon, Google, ...)

=> [50-100] choices (per Task)

**Network-aware**
- P providers for Task T (e.g. Amazon, Google, ...)
- I_k physical instances per P_k (e.g. in Japan, Germany, ...)

=> \( P \cdot I_k \) choices per Task

=> [50-100] x [20-120] = [1000-12000] choices (per Task)

---

**Search Space:**
\[
#S^{WF} \rightarrow (#I \cdot #S) \cdot #WF
\]
- \(#S\) := Services per Task
- \(#I\) := Instances per Service
- \(#WF\) := Size of Workflow
Motivation (4/4): Dependencies

-> Network-aware Optimization

Change of service for task T

=> Network QoS from preceding tasks and to following tasks change!

New dependencies between each task and its predecessors and successors!

15th March 2013
**NETWORK QoS**

**Distributed Architecture**

+ **Network-aware QoS Algorithm** (compute QoS of workflow)
+ **Network Model**

=> Challenge ②

(network-independent QoS)
SCALABLE DISTRIBUTED SERVICE ARCHITECTURE

[KIH12a]

A distributed architecture [framework] using a flexible number of nodes

=> Near-optimal latency + Scalability in terms of distributed-ness

=> Address part of Challenge ② (network-independent QoS)
Distributed Architecture -> Distr. Exec.
Architecture for a Middleware

Distributed Architecture
- one master node (=user)
- flexible number of slave nodes

Integration with NEQoSA\(^5\) and NEMo\(^6\)

15th March 2013

サービスコンピューティング研究会、Adrian Klein
Evaluation: **Near-Optimality**

**vs.** Perfectly Distributed Arch. (-> [o])

QoS

Centralized architecture!

Near-optimal!

+1 node! /rand.

Latency [ms]

Control Nodes

Dijkstra [o]
NetGA [o]
Dijkstra
NetGA

500

500

15th March 2013
**NETWORK MODEL (NEMO)**

A general model [framework] augmented to find close service nodes

=> Address part of **Challenge ②** (network-independent QoS)
General Model [Framework]

**Challenges**
- Estimate network latency
- Scale with number of network nodes
- Dynamic joining of nodes

**Framework**
- Use a *Euclidean network model*, e.g. Vivaldi [DCKM04]
- **Project** onto 2D plane for optimization algorithms
- **Augment** 2D representation

---

15th March 2013
Augmentation

Use K-D Tree
(⇒ Sanga)

Custom Locality-sensitive Hashing Scheme
(⇒ NetGA)

Find close locations/services
1. Around a network location
2. Between two network locations
NETWORK-AWARE OPTIMIZATION

Network-aware Genetic Algorithm (NETGA\(^7\))

\[ \rightarrow + \text{Self-adaptive network-aware Genetic Algorithm (SANGA}^8) \]

\[ \Rightarrow \text{Challenge 3} \]

(network-unaware Optimization)
**SELF-ADAPTIVE NETWORK-AWARE GA**

[KIH13]

Genetic Algorithm which **self-adaptively** balances **network-aware operators** vs. **standard operators** (extending **NetGA**)

=> **Near-optimal** latency + **Scalability** in distributed settings

=> Address **Challenge 3** (network-unaware Optimization)
**SANGA**

Standard Self-adaptive GA:

- **+ Custom Operators** for:
  - Initial Generation (network-aware + general)
  - **Mutate** (network-aware)
  - **Crossover** (2x: network-aware + other QoS)

- **+ Custom Self-adaptation Rules**

15th March 2013
NetMutation: Mutate Operator

Replace one service of the current selection with one of the close services by using \textbf{NEMO}^6

(consider previous and next service in workflow)

\begin{itemize}
  \item Standard mutate: search space
  \item NetMutation: search space
  \item => most probable choices
\end{itemize}
**NetCrossover:** Crossover Operator

**Combine** close services from parents by **NeMo**

(randomly in proportion to distance from previous + next service)
Self-adaptation

Challenges

- Users have different (low/high) preferences for network QoS
- Network-aware operators are not effective at optimizing other QoS!
  - Do not apply often?
  - Hardwire vs. QoS preferences?

Approach

- Balance net. ops vs. general ops
- Self-adaptive realization

Self-Adapt (unique combination)

1. Record average & maximum QoS improvement ratio (over recent uses)
2. Global mutate vs. crossover ops
3. Local mutate/crossover ops
   - Useful ops? -> 100%
   - Max imp. ratio -> 80%
   - Avg imp. ratio -> 20%

Based on Probability Matching, Adaptive Pursuit, and Power Probability Matching. (see [KIMK12])
Evaluation: Network Dataset [Pro08]

- **Trace dataset** of the Univ. of Minnesota (@ ridge.cs.umn.edu/pltraces.html)
- Collected on **PlanetLab** (@ www.planet-lab.org)
- **10 months** of data from more than **240 nodes**

Generate **100,000 unique locations** by **mutation**
Evaluation: **Near-optimal Network QoS**

- **Quality of Service (QoS)** vs. **Workflow Length (#Tasks)**

- **Standard GA** as bad as random!
  - Randomizer
  - Dijkstra= (real net. data)
  - Dijkstra~ (+ net. model)
  - GA
  - GA*
  - NETGA
  - SANGA

- 3x min!
- 2x min!

- 500 WF

15th March 2013

サービスコンピューティング研究会、Adrian Klein
Evaluation: **Near-optimal Other QoS**

![Graph showing evaluation results](image)

- **QoS**
  - Randomizer
  - Dijkstra=
  - Dijkstra~
  - GA
  - GA*
  - NetGA
  - SANGA

- **Utility**

- **Weight of Latency**

- **Other QoS dominate!**

- **Non-adaptive NetGA is overtaken by GA*!**

15th March 2013
Evaluation: **Scalability**

**Dijkstra does not scale!**

- **Self-adaptation pays off!**

- **Workflow Length [#Tasks]**

- **Service Instances [per Task]**

- **Time**

  - Computation Time [s]

  - Workflow Length [#Tasks]

  - Service Instances [per Task]
3-B) NETWORK-AWARE SCs: Conclusion

1. Solved N’QSC for latency
   (bandwidth ext. possible out-of-the-box)

2. Balanced specialized operators (net.)
   versus general operators (general QoS)

=> Applicability + Scalability!
=> addressed Challenges ②+③
   (network-independent QoS)
   (network-unaware Optimization)
4. CONCLUSION

PhD Overview -> Big Picture -> Conclusion -> Outlook
PhD Overview

1. Problem

15th March 2013

サービスコンピューティング研究会、Adrian Klein
Big Picture

# Executions

Long-term

L’QSC => LoSC

LN’QSC => ?

Short-term

QSC [ZBD+03]

N’QSC => NESC

Distributed-ness

Idea for LN’QSC: Combine TIMREL$^3$ with SANGA$^8$
- Merge GA encodings
- Long-term GA utility + network QoS computation
- Long-term GA ops + network GA ops
- Balance merged ops with Self-Adaptation

=> Caveat is the search space: $S^{#WF}$ -> $(#I\cdot S^{#WF})^{#TD}$
Conclusion

Theory

Two Extensions of the QSC problem, L’QSC and N’QSC.

=> Issue of I. APPLICABILITY

Practice

Effective and efficient custom optimization algorithms addressing the problem extensions’ characteristics and the caused increased search space.

=> Challenges ①+②+③

=> Issues of I. APPLICABILITY + II. SCALABILITY
Applicability and Scalability remain ongoing challenges.

In this PhD I have worked on addressing them through effective and efficient approaches for long-term and network-aware service compositions.

Thank you very much for listening!